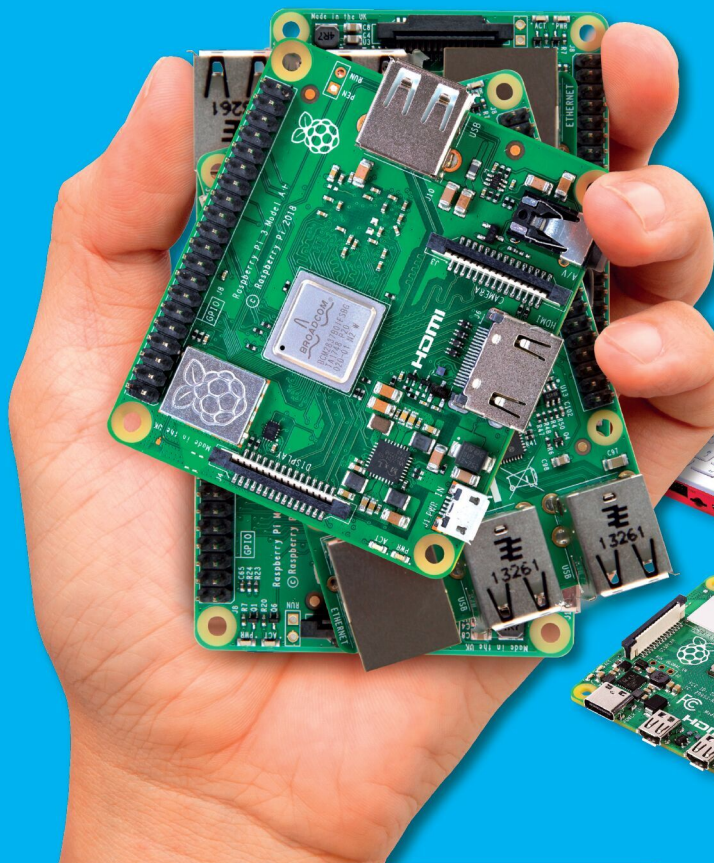


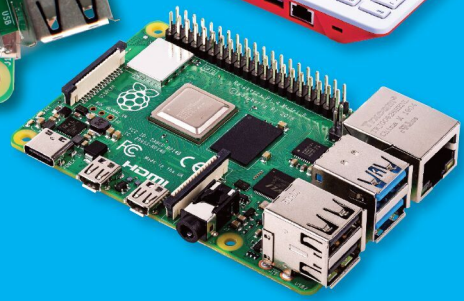
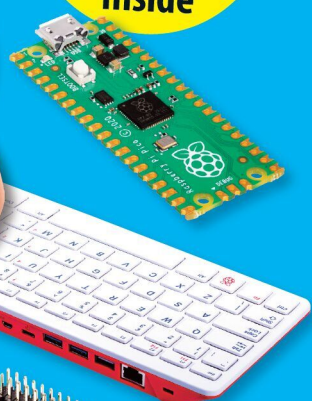
GET STARTED WITH RASPBERRY PI CODING PROJECTS

# Raspberry Pi For Beginners

- ✓ Jargon-free Tips & Advice
- ✓ Step-by-step Tutorials
- ✓ Clear Full Colour Guides



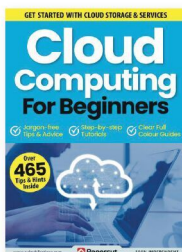
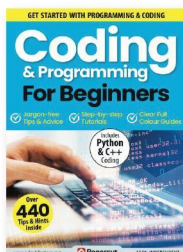
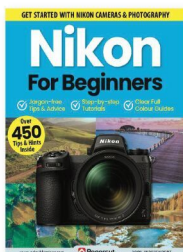
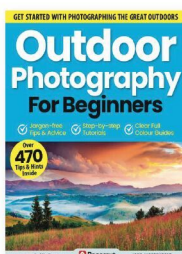
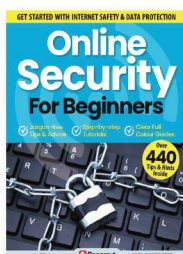
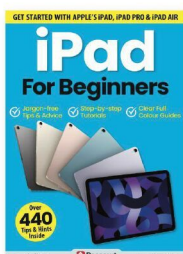
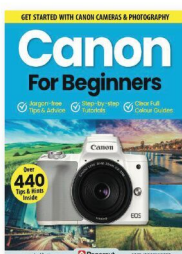
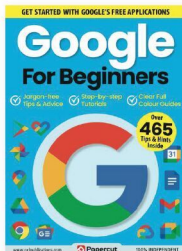
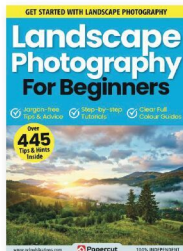
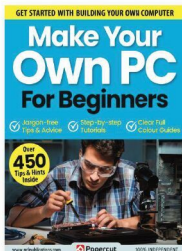
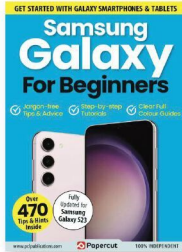
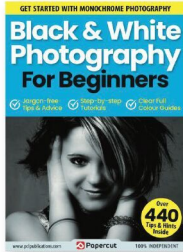
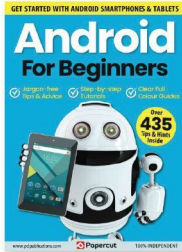
Over  
**480**  
Tips & Hints  
Inside



Read  
More

# For Beginners

Tech Guides Available on  Readly



For a full list of titles available visit:

[www.pcpublications.com](http://www.pcpublications.com)

# Raspberry Pi For Beginners



Raspberry Pi For Beginners is the first and only choice if you want to learn everything about the Pi from the ground up - where the only limitation is your imagination. Learn how to build some amazing projects from a retro gaming console to a home entertainment system. The possibilities of the Raspberry Pi are endless and the best place to start is right here.



[www.pclpublications.com](http://www.pclpublications.com)



# Contents

## 6 Raspberry Pi is Good For You!

- 8 Say Hello to Raspberry Pi
- 10 Get to Know the Raspberry Pi
- 12 Up Close with the Raspberry Pi 4 Model B
- 14 The Pi 400 – Up Close
- 16 Pi 400 – Return of the 80s Home Computer
- 18 Ten Pi Projects and Ideas
- 20 The Pi Pico – Up Close
- 22 Pico Power: the Tiny Microcomputer
- 24 The Pico and MicroPython
- 26 Pico Examples and How to
- 30 Pico Projects & Ideas
- 32 Other SBCs to Use
- 34 Raspbian: The Complete Operating System
- 36 Which Pi is Right for Me?
- 38 Raspberry Pi in Numbers
- 40 Kit You'll Need and How to Set it Up
- 42 Set Up Raspberry Pi Using a Mac
- 44 Set Up Raspberry Pi Using a Windows PC
- 46 The Raspberry Pi Desktop: What You Will Need
- 48 Debian Buster with Raspberry Pi Desktop

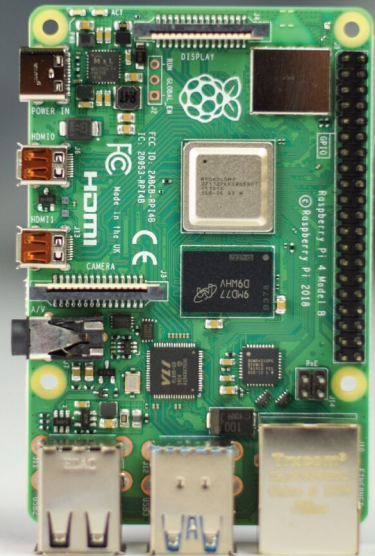
## 50 Explore Raspbian

- 52 Take a Tour of PIXEL
- 54 Exploring the Command Line
- 56 Setting Up a Static IP Address
- 58 Connecting to the Pi Remotely
- 60 Using APT to Install and Remove Programs
- 62 Get More from the Terminal App
- 64 Using the File Manager
- 66 Edit Images with GIMP



## 68 Python on the Pi

- 70 Starting Python for the First Time
- 72 Your First Code
- 74 Saving and Executing Your Code
- 76 Executing Code from the Command Line
- 78 Numbers and Expressions
- 80 Using Comments
- 82 Working with Variables
- 84 User input
- 86 Creating Functions
- 88 Conditions and Loops
- 90 Python Modules
- 92 Python Errors
- 94 Python Graphics
- 96 Glossary of Terms



*“Thanks to the Raspberry Pi’s unique setup, operating system and amazing community of imagineers, this little computer has become one of the most important pieces of educational technology in the world. It’s helped people learn how computers, coding and electronics work, as well as helping science and engineering, astronauts and technicians, young and old, teachers and students.*

*You will learn and discover how the Pi works, what you can do with it and where to take your Pi adventures.”*



Raspberry Pi is Good For You!





# Raspberry Pi is Good For You!

Your journey with this amazing little computer begins here. Say hello to the Raspberry Pi; learn how it works, what kind of operating system it has, what you need to get it up and running and much more. This is your first step into the world of the Raspberry Pi.

In this section, you'll discover what makes the Raspberry Pi such a fabulous little computer and project board. There are in-depth guides to the hardware, operating system, what you'll need to begin with and enjoying the software the Pi has to offer from your computer's desktop. Now let's see what this amazing computer has to offer.



# Say Hello to Raspberry Pi

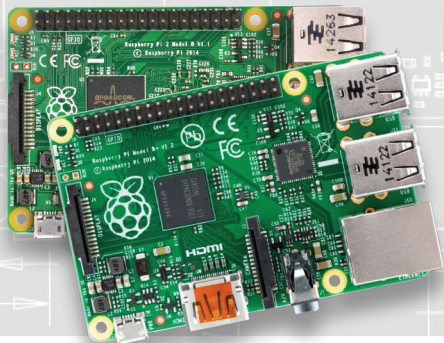
Let's take a look at the best-selling British computer ever: the Raspberry Pi! This bare-bones machine just looks like any other piece of circuitry, until you realise it's a full-blown credit-card sized computer, ideal for learning programming and home hacking.

## Raspberry Pi 2 Model B Raspberry Pi 1 Model B+

The older Raspberry Pi 2 model can still be found on sale. It has the same form factor as the Raspberry Pi 3 with 4 USB Ports, Ethernet connection and Micro USB.

It may look the same but it has a slower 900 MHz quad-core ARM Cortex-A7 CPU and 1GB RAM. This brings it close in line to many low-end desktop computers.

Complicating matters slightly is the existence of a Raspberry Pi Model 1 B+, which is identical in layout to the Raspberry Pi Model 2 but with an even slower ARMv6 700MHz processor. If you are confused which type you own then type `cat /proc/cpuinfo` at the command line. If it lists four ARMv7 processors you are using a Raspberry Pi 2, if you see just one with ARMv6 then you have the older Raspberry Pi Model 1 B+.



## Raspberry Pi 3 Model B+

The Raspberry Pi 3 Model B+ was launched on 14 March - International Pi day 2018. This is an improved version with a faster 1.4GHz, 64-bit quad-core ARM Cortex-A53 processor, a dual-band 802.11ac wireless LAN and Bluetooth 4.2 controller, faster Gigabit Ethernet, improved mass storage, USB booting, improved thermal management and Power-Over-Ethernet support. All other aspects of the new Raspberry Pi are the same as the previous Pi 3 and as such, all the previous content you come across is compatible. If you're new to the wonderful world of the Raspberry Pi, this is the board we recommend you buy.





The Raspberry Pi is a British built low cost computer that enables everybody to learn computing, start programming and explore basic electronics. It's the size of a credit card but capable of running a full operating system and doing everything a desktop does.

More importantly, with the Raspberry Pi you install your own operating system, connect all your devices and create your own programs using languages like Scratch and Python. There's no case so you can hook up electronic circuits to the pins and control them to get input and output, so you can hack together electronic projects at home.

Setting up a Raspberry Pi is pretty simple, and throughout this book we'll take you step-by-step from unboxing your Raspberry Pi to setting it up and getting started.

There are many different models of Raspberry Pi available and each has slightly different features (see opposite).

The fundamentals of each Raspberry Pi are similar though. Each model is a lightweight

computer on a single board that's roughly the size of a credit card. Each Raspberry Pi board features a processor made by ARM, which is similar to the models you find in a mobile phone. The ARM processor is fast and lightweight but it runs a different set of software than you might be used to. There are many different operating systems (OS) available, but for most of this book we'll focus on one called Raspberry Pi OS, which is the OS recommended by the Raspberry Pi Foundation.

There's no hard drive on the Raspberry Pi, instead the operating system is installed on an SD Card (the cards typically used in cameras). The operating system is installed onto the SD Card by copying the files to the SD Card using a computer. We'll show you how to set up the Raspberry Pi OS using a Mac or Windows computer.

Once you have an OS installed on the SD Card you can set up your Raspberry Pi. The Raspberry Pi is connected to a monitor via a HDMI socket whilst a keyboard and mouse is

attached to the USB socket. A smaller Micro USB connection is used to provide power to the device. Most Raspberry Pi models feature an Ethernet connection, and an Ethernet cable is connected from your Raspberry Pi to a socket on your modem router.

Both the Raspberry Pi 3 and 4 models have built in wireless networking and Bluetooth, making it easy to get online. You can attach a USB Wi-Fi dongle or Ethernet adaptor to older models.

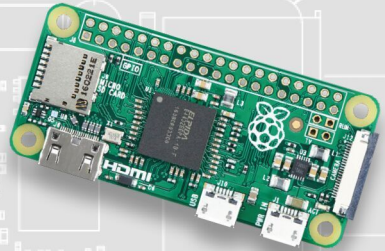
First it will help to know which Raspberry Pi you are using. This can be tricky as there have been several distinct models so far. The layouts opposite will help you determine which Raspberry Pi you have.

This book covers all the different models, and they offer largely similar features, although newer models offer additional extras. Once you know which Raspberry Pi you own, you can get it up and running.

## Raspberry Pi 4 Model B

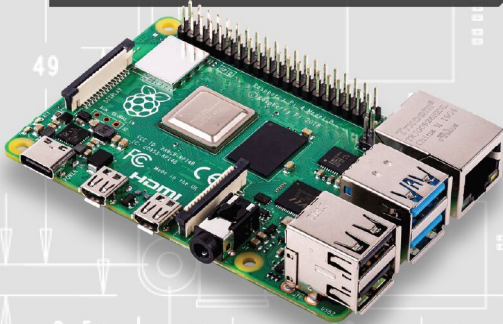
Released on June 24th 2019, the Raspberry Pi 4 Model B introduced a new direction in the layout and hardware specification of the Pi. The Pi 4 Model B now comes in three different memory size versions: 2GB, 4Gb, and 8GB. Obviously, the 8GB version is the most powerful of the three, and the Pi with the highest amount of on-board memory. Subsequently, it was sold out on the day of release and as such, has taken some time to get into the hand of the mainstream Pi users.

There's also an improved CPU, dual-monitor support in the form of a pair of micro-HDMI ports, and improved connectivity. Overall, it's certainly a more capable Pi.



## Raspberry Pi Zero

The new Raspberry Pi Zero is a super small, super cheap computer that costs just £4. It's half the size of the model A+ but has a surprising amount of power, including a 1Ghz single-core CPU and 512MB RAM. It has the full 40-pin GPIO header but you'll need to buy the pins separately and solder them in. It uses the same Micro-SD card as the other Raspberry Pi devices, so you can swap SD cards between them. Last but not least, it draws an incredibly low amount of power (Micro USB) making it ideal for small power devices. The ports are smaller than you'd find on the larger Raspberry Pi though, so you'll need a mini-HDMI adaptor as well as a micro-USB adaptor to connect devices. A four-port USB Hub and Ethernet adaptor will come in handy too.





# Get to Know the Raspberry Pi

While the Raspberry Pi 4 is the newest model, the Pi 3 models are the most popular among the community. They're cheaper, more compatible with current hardware and software, and still pack a significant performance punch. Here's what powers the fantastic Pi 3.

## 40 GPIO Pins

The GPIO (General Purpose Input Output) pins can be accessed directly on the Raspberry Pi. These are used in projects to connect the Raspberry Pi to electronic circuits and control electric devices. Some can be turned on and off while the Raspberry Pi is running.

## ARM-Powered

At the heart of the Raspberry Pi 3 is a Broadcom BCM2837 System on Chip (SoC). It contains a powerful 1.2GHz 64-bit quad-core ARM Cortex-A53 CPU. This is 50-60 percent faster than the Raspberry Pi 2 and ten times as fast as the original Raspberry Pi.

## DSI

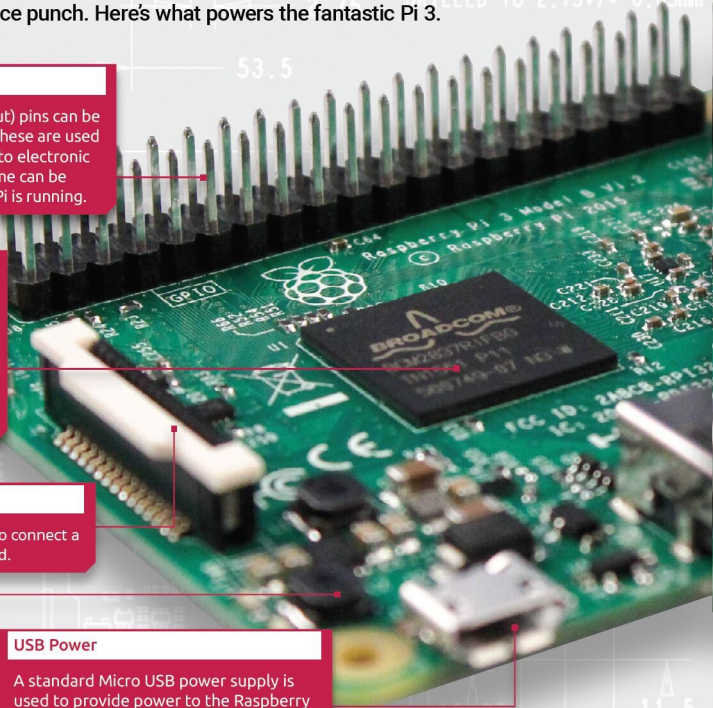
The Display Interface (DSI) can be used to connect a display directly to the Raspberry Pi board.

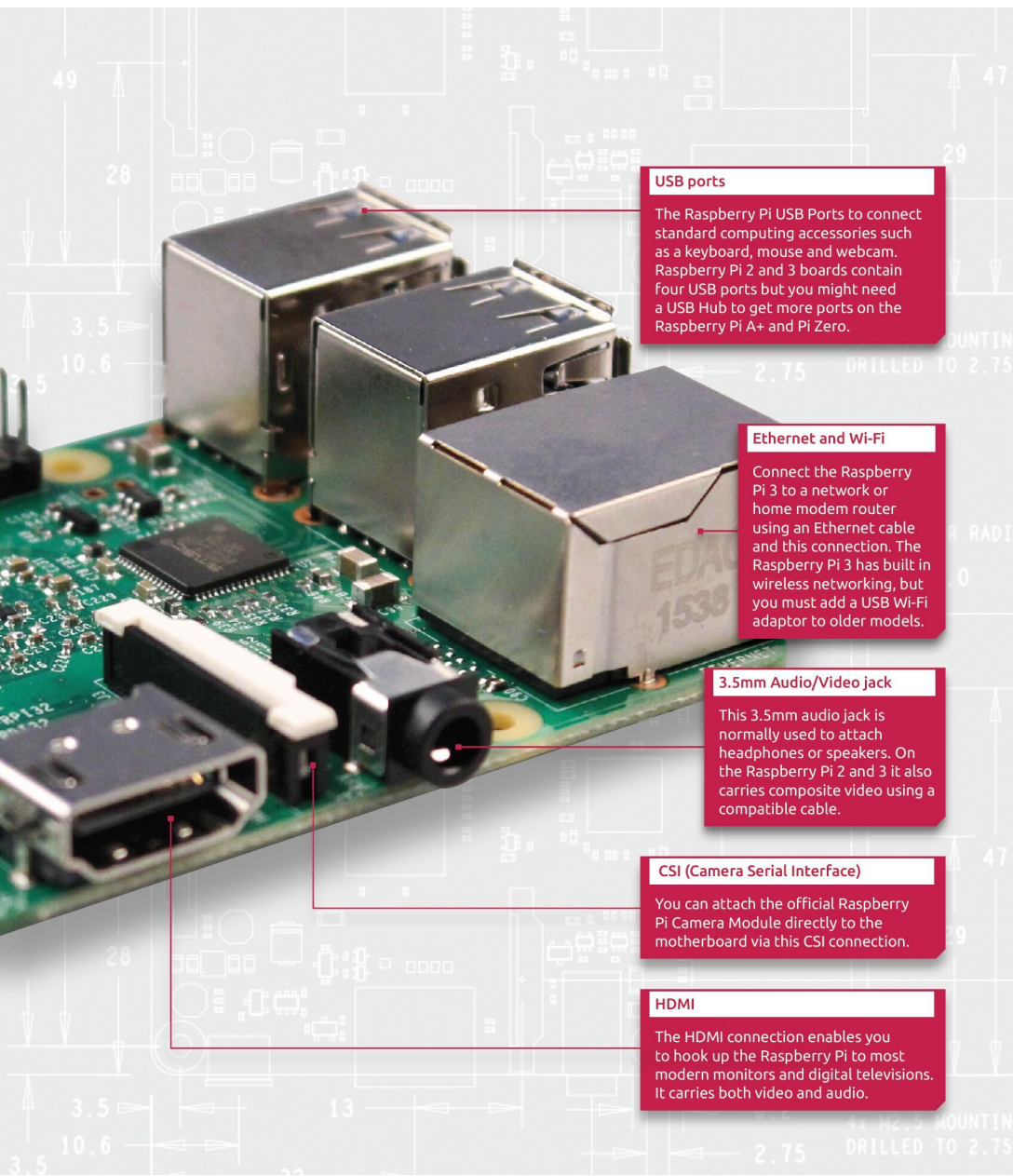
## SD Card Slot

Instead of a hard drive you use an SD Card with the Raspberry Pi. The Raspberry Pi 3 (pictured here) uses a Micro SD Card. Some older Raspberry Pi models use a regular SD Card.

## USB Power

A standard Micro USB power supply is used to provide power to the Raspberry Pi. You don't get a power supply but can use any 5V 2000mA Micro USB power supply and compatible cable, or buy an official power supply separately.





## USB ports

The Raspberry Pi USB Ports to connect standard computing accessories such as a keyboard, mouse and webcam. Raspberry Pi 2 and 3 boards contain four USB ports but you might need a USB Hub to get more ports on the Raspberry Pi A+ and Pi Zero.

## Ethernet and Wi-Fi

Connect the Raspberry Pi 3 to a network or home modem router using an Ethernet cable and this connection. The Raspberry Pi 3 has built in wireless networking, but you must add a USB Wi-Fi adaptor to older models.

## 3.5mm Audio/Video jack

This 3.5mm audio jack is normally used to attach headphones or speakers. On the Raspberry Pi 2 and 3 it also carries composite video using a compatible cable.

## CSI (Camera Serial Interface)

You can attach the official Raspberry Pi Camera Module directly to the motherboard via this CSI connection.

## HDMI

The HDMI connection enables you to hook up the Raspberry Pi to most modern monitors and digital televisions. It carries both video and audio.



# Up Close with the Raspberry Pi 4 Model B

Released at the end of June 2019, the Raspberry Pi 4 Model B is a significant improvement in terms of hardware. Labelled the 'Ultimate' Raspberry Pi, this new generation offers true desktop computing power.

## FASTER AND CONNECTED

There's a lot to like about the Raspberry Pi 4: Up to 8GB of on-board memory, a faster quad-core CPU, support for dual 4K displays via a pair of micro-HDMI ports, and more. Let's take a look at the Pi 4, and see what it's got under-the-hood.

### PRICING

Since the Raspberry Pi 4 now offers three different versions within the new model, it stands to reason that the pricing for it has changed slightly. The 1GB memory version of the Pi 4 is priced at around £34, depending on where you shop. The 2GB memory version is on sale for around £44, while the top-end, 4GB memory version will set you back in the region of £54. Together with the cost of the Pi itself, you will also need to factor in one or two micro-HDMI cables (depending on whether you want to connect one or two monitors), each costing roughly £5-plus.

### Improved GPU

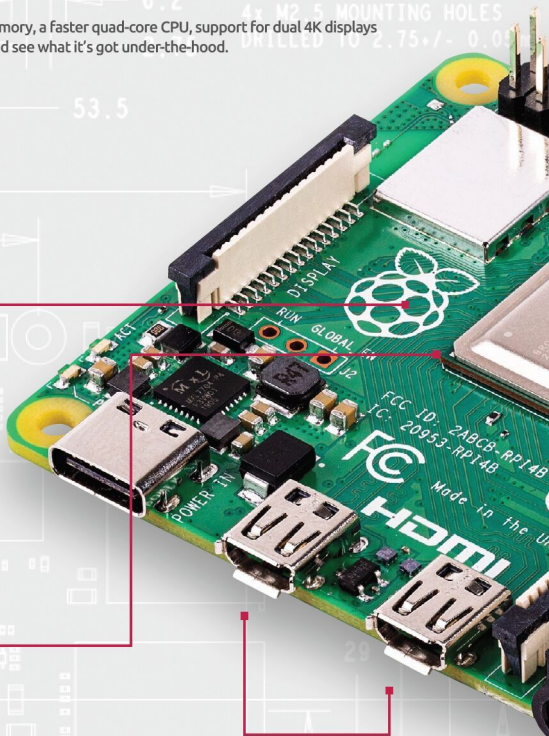
With a VideoCore 6 GPU, the Pi 4 is capable of 4K resolutions at 60FPS (Frames per Second), and thanks to some tweaks to the video codes in both Raspbian and the version of VLC packaged with Raspbian, the Pi 4 is now a pretty decent media device.

### Improved Processing

The 1.5GHz, 64-bit quad-core ARM Cortex-A72 CPU, is the driving force behind the Pi 4's improved performance. Thanks to this CPU, you're able to enjoy faster apps and content.

### Dual Display

In a shock move the team behind the Pi 4 decided to include a pair of micro-HDMI ports. This means you're able to connect two monitors for dual-screen displays.





## PI 4 SPEC SHEET

- 1.5GHz 64-bit quad-core ARM Cortex-A72 CPU (ARM v8, BCM2837)
- 2GB, 4GB or 8GB RAM (LPDDR4)
- On-board wireless LAN - dual band 802.11 b/g/n/ac
- On-board Bluetooth 5.0, low-energy (BLE)
- 2x USB 3.0 ports, 2x USB 2.0 ports
- Gigabit ethernet
- Power-over-Ethernet (requires the Raspberry Pi POE HAT)
- 40-pin GPIO header
- 2x micro-HDMI ports (up to 4Kp60 supported)
- H.265 (4Kp60 decode)
- H.264 (1080p60 decode, 1080p30 encode)
- OpenGL ES, 3.0 graphics
- DSI display port, CSI camera port
- Combined 3.5mm analog audio and composite video jack
- Micro-SD card slot
- USB-C power

## NEW POWER & VIDEO

There are two major differences to the Pi 4 over its older siblings: the use of a USB-C power port, and the newly introduced micro-HDMI ports. Therefore, if you're going to be getting hold of a Pi 4, then you'll also need to source a micro-HDMI cable, and the relevant USB-C power adapter – since the power and HDMI from the older Pi models won't work with this fourth generation Pi.

### USB 3.0

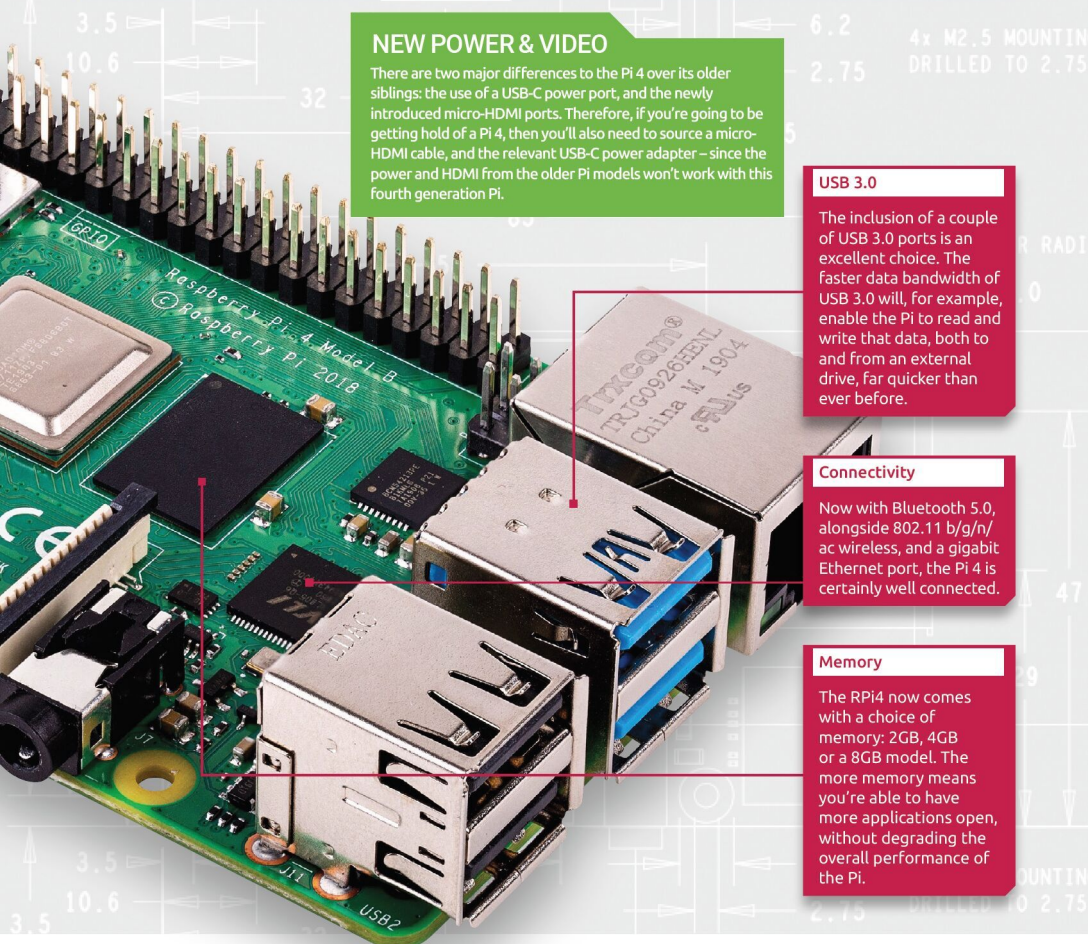
The inclusion of a couple of USB 3.0 ports is an excellent choice. The faster data bandwidth of USB 3.0 will, for example, enable the Pi to read and write that data, both to and from an external drive, far quicker than ever before.

### Connectivity

Now with Bluetooth 5.0, alongside 802.11 b/g/n/ac wireless, and a gigabit Ethernet port, the Pi 4 is certainly well connected.

### Memory

The RPi4 now comes with a choice of memory: 2GB, 4GB or a 8GB model. The more memory means you're able to have more applications open, without degrading the overall performance of the Pi.





# The Pi 400 – Up Close

The Raspberry Pi 400 is an all-in-one keyboard Pi that was released in November 2020, offering the user a different approach to their Pi lifestyle. This compact and powerful Pi is the flagship of a new kind of Pi, and one we'll hope to see more of in the future.

Inside the Pi 400 is a modified Raspberry Pi 4. It's longer, spanning the width of the keyboard, with a large heatsink to dissipate enough heat for the overclocked processor. The Pi 4's communication ports are spaced along the back of the Pi 400 keyboard, minus a second USB 2.0 port and a 3.5mm audio jack.

## 40-Pin GPIO

The slightly recessed GPIO port is directly connected to the Pi 400 board inside the keyboard. This means you can still attach the usual Raspberry Pi HATs and other devices as you would a bare-metal Raspberry Pi.

## microSD Card Slot

The microSD Card Slot is where you'll insert the card that contains the Raspberry Pi OS – or any other operating system you prefer to run that's compatible with the Raspberry Pi 4.

## micro-HDMI Ports

As with the Raspberry Pi 4 bare-metal device, you're able to connect to a dual-monitor setup with the Pi 400 via the two micro-HDMI ports. Both are capable of up to 4K at 60 frames-per-second.





## TECH SPECS

The Pi 400 has a slight advantage over the Raspberry Pi 4 in terms of its processing power, but it loses out in the overall memory department. However, the specs are an impressive Broadcom BCM2711 quad-core Cortex-A72 (ARM v8) 64-bit SoC, overlocked to 1.8GHz – and tests have shown that the Pi 400 can be further overlocked to a stable 2.2GHz. And there's 4GB of LPDDR4-3200 memory available too.

Not bad for a mere £94, which includes the Pi 400, micro-HDMI cable, power supply, and a 16GB microSD card pre-programmed with the latest Raspberry Pi OS.

### USB-C Power

The Pi 400 is powered by the same USB-C connector power adapter as the Raspberry Pi 4, offering a minimum of 3A.

### 2x USB 3.0 Ports

There are two fast USB 3.0 ports available on the rear of the Pi 400, so you can attach projects, as well as game controllers and other such devices.

### Regional Keyboard

The Pi 400's keyboard is available in a number of regional variations: UK, US, German, French, Italian and Spanish.

### USB 2.0

There's also a legacy USB 2.0 port available. In case you're wondering, where the Pi 4 had two USB 2.0 ports, on the Pi 400 the other USB 2.0 port is hardwired as an internal ribbon connected to, and powering, the Pi 400's keyboard.

### Gigabit Ethernet

Gigabit Ethernet offers a fast connection for the Pi 400 to your home network and peripherals.

### Wi-Fi

The Pi 400, like the Raspberry Pi 4, comes with dual-band, 2.4GHz and 5GHz, IEEE 802.11b/g/n/ac wireless connectivity, along with Bluetooth 5.0.





# Pi 400: Return of the 80s Home Computer

The 80s is considered the golden era of the home computer, a time when computing components were small enough to fit inside the living room of a house, and versatile enough to work on as well as play games. It was a wonderful time to experience, and the Pi Foundation has kept that feeling alive with the Pi 400.

Released in November 2020, and while the UK and most of the world was still in lockdown, the Raspberry Pi 400 came at a time when we were beginning to feel like there was no end to the pandemic and the restrictions it brought. Initial thoughts were that the Pi 400 was a new small-board computer, in a similar vein to the Pi 4 and its predecessors. However, we were quite wrong.

The Pi 400 is a very different beast to that of its siblings. Rather than the naked board we've seen in the past, the Pi 400 came as a complete keyboard package, with the Pi's familiar ports lined up along the backplate of the keyboard.

This in itself was remarkably reminiscent of the home microcomputers of the 80s, the prime examples being the ZX Spectrum and the Commodore 64. Indeed, the nostalgia was strong with this one, and it brought a smile to more than one retro fan's face.



## THE REINVENTION OF THE MICROCOMPUTER

It's more than sheer aesthetics that makes the Pi 400 hark back to a simpler age of computing. Inside the keyboard is a Raspberry Pi 4, albeit one that's had a facelift and some hardware alterations. The Pi 4's ports, which are the twin micro HDMI and USB 3.0 ports, USB 2.0, Gigabit Ethernet and microSD card slot, take up the rear of the keyboard together with a port that connects directly to the 40 GPIO pins – and resembling an older style communications port from the rear of an XT or earlier machine.

In terms of hardware, the Pi 4 inside the Pi 400 has been expanded to fit most of the width of the keyboard. The processor is the same ARM v8 Cortex-A72 as found in the bare Raspberry Pi board, but in this instance it's been overclocked to 1.8GHz; 300Mhz faster than the bare board. There's also 4GB of LPDDR4 RAM installed, but sadly no sign of an 8GB version – which the Pi 4 bare model has on offer.

The keyboard itself is available in different language layouts: UK, US, Germany, France, Italy and Spain. Essentially, it's the same keyboard that the Pi Foundation has been selling for some time, except this time there's an actual Raspberry Pi inside of it as opposed to it simply being a multi-port USB hub.

The overlock is thanks to the inclusion of a large, internal aluminium heatsink, that spreads across the entire internal space of the keyboard. This, and thanks to a couple of air vents on the bottom of the keyboard, are what allows the Pi 400 to be overclocked to the default 1.8GHz. In fact, we even took the Pi 400 to an impressive 2.2GHz, and it remained stable and cool enough for continual operation.

The heat dissipation from the large heatsink is a great idea – and one that was previously seen in the Commodore 64. There's also a thermal sticky pad on the top of the heatsink, that's attached to the Pi 400's processor and helps to lift as much heat as possible from the CPU. If you get inside, you'll also see why the Pi 400 is missing a USB 2.0 port; since it only features a single 2.0 port over the Pi 4's dual USB 2.0 ports. One of the Pi 400's USB 2.0 ports has been used as a hard-wired ribbon connector to the keyboard section of the unit. Again, those of you familiar with the internals of an 80s home computer will recall ribbons being attached to keyboard membranes.







## PI 400 DESKTOP COMPUTING

One of the prime benefits of having the Pi 400 keyboard setup is the fact that it can be quickly plugged into a monitor and working from the moment you hit the power key. The Pi 4 bare board, for example, requires a separate keyboard and mouse, and, let's be honest here, not many folk want to see a bare circuit board on their desk. True, there are some great design Pi 4 cases around these days, but the Pi 400 already comes in a neat package.

The extra overclock brings a decent amount of oomph to the Pi 400, and as such it's perfectly reasonable to use the Pi 400 as a standard desktop computer. However, there comes with it some negatives in this respect.

While a great device, the Pi 400 is a little too small for comfort, specifically a typical working day's use. It's roughly the same size as a laptop keyboard, but the keys don't feel as good – a bit too squishy for our tastes. But, as with most things, we'll probably get used to it over time.

Another caveat that's worth mentioning, is that the Pi 400 lacks a 3.5mm audio port. Where the Pi 4 model had one, this doesn't, instead opting to output sound via the HDMI port(s). While this isn't a deal-breaker for most, it does mean that for personal listening you'll need to hook up a set of headphones to your monitor, or find another solution. The built-in Bluetooth will connect to headphones with that technology,

but if you're limited to wired models, then you could be looking to buy upgrades.

However, with all that in mind, the Pi 400 is another step closer to becoming the only computer you'll ever need.

## THE ZX PI 64

There's a good feeling of nostalgia when plugging in a Pi 400 to a monitor or TV. It's like you're back to being 10 years old again and plugging in the Spectrum or the C64 for a spot of gaming or homework, before having to unplug it all when your Mum calls through that dinner is ready.

Thankfully, these days, it's easy to save where you were – and the entire thing doesn't crash when the kettle is switched on!

Overall, the Pi 400 is a great addition to the Foundation's stable, and one that we're sure you'll come to love and use daily.





# Ten Pi 400 Projects and Ideas

You may think that a Raspberry Pi 4 inside a keyboard would create some limitations, however that's not the case. Indeed, you won't be able to utilise the Pi 400 in quite the same way as a standard Pi 4, but there's a lot you can get to grips with.

## EVERYONE LOVES PI

We've put together ten possible projects that you could use your Pi 400 with; that will entertain and keep you busy while helping you learn more about coding, the Pi and computing.

### RETRO COMPUTING

With the Pi 400 being very similar to that of the great home computers of the 80s, we thought we'd start these ideas with a simple retro-themed project. Fuse is a ZX Spectrum emulator that's available for the Raspberry Pi. Install it, and you can enjoy some of the greatest games ever created.



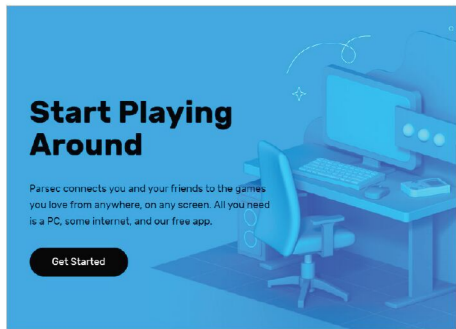
### C64 LOVE

Since we've mentioned the ZX Spectrum, it's only fair that we include the other great 8-bit home computer of the 80s, the Commodore 64. The Pi uses a program called c64-raspi for C64 emulation, and can be installed via [https://c64emulator.111mb.de/index.php?site=pp\\_raspi&lang=en&group=c64](https://c64emulator.111mb.de/index.php?site=pp_raspi&lang=en&group=c64).



### STREAM PC GAMES

It's possible to stream any game installed on your powerful gaming PC to a Raspberry Pi. All you need is to ensure that your home network is up to scratch – use Ethernet connections between the two if you can – and to install Parsec from <https://parsec.app/downloads>.



### GOOGLE STADIA

While we're on the subject of streaming games, if you own a Google Stadia, then you're able to stream games to the Pi via the Chromium browser. You'll need a good network connection, and some games may well be a bit laggy, but it's worth a try.





## OVERCLOCKING

The Pi 400's 1.8GHz processor is adequate for most tasks, but thanks to the large, metal heatsink inside the keyboard, it's possible to get that clock speed even higher. There's a great YouTube tutorial on how to overclock the Pi 400 to 2.2GHz at <https://www.youtube.com/watch?v=DqZ99mGbSR0>.



## RUN WINDOWS 10

Yes, it's possible to run a version of Windows 10 on the Pi 400. There are going to be some limitations; there's no sound, Wi-Fi or Bluetooth, but it's a fun project that's definitely going places. Check out the install video courtesy of Lepspvideo at <https://www.youtube.com/watch?v=xbRBovkmZvU&t=0s>.



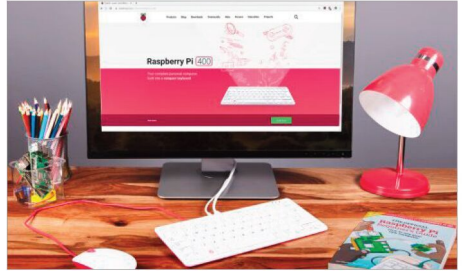
## CODING BASE

The Pi 4 made for a great platform on which to learn how to code, but thanks to the neater approach of the Pi 400, it's now an even better base on which to code. You can learn how to code with Python, MicroPython, C and C++ and many more languages. Check out our coding guides at <https://bdmpublications.com>.



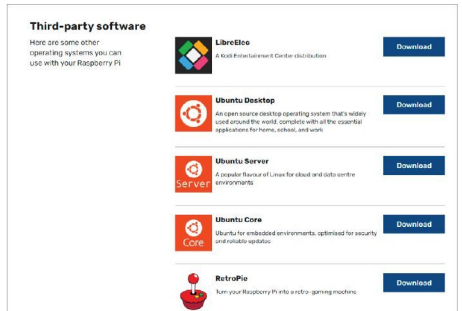
## DESKTOP PC

Since lockdown started in 2020, folk have been coming up with novel ways in which they can work and educate at home on a budget. The arrival of the Pi 400 was a great boost for home working, and it makes for a great desktop PC. Just add a mouse, dual monitors and you can even use Teams via Chromium.



## INSTALL A DIFFERENT OS

The default Raspberry Pi OS isn't the only operating system available for the Raspberry Pi. We've already seen RetroPi, but there's also a version of Ubuntu, an entertainment OS called LibreElec, RISCOS, the original ARM OS and many more to discover.



## MEDIA CENTRE

The Pi 400 makes for a great under-the-TV media centre. You don't specifically need a media-centric operating system, the Raspberry Pi OS will suffice. With it you can watch network-stored movies, browse the Internet, watch YouTube content and much more.

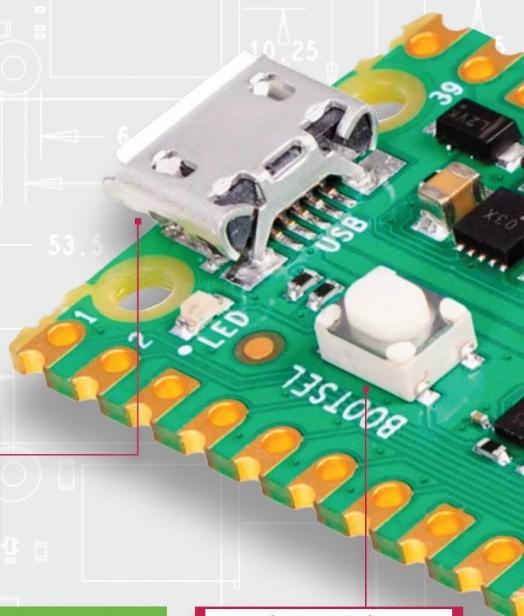




# The Pi Pico – Up Close

The Pi Pico isn't your normal Raspberry Pi. Instead, this is a tiny microcontroller, designed to help enthusiasts control and utilise connected appliances and electronics projects. From displaying an LCD to controlling LEDs, using MicroPython and C++.

The Pi Pico may be small, but it packs a decent punch. With it you can expand your electronics projects, and incorporate a whole new level of functionality that you weren't able to before with the traditional Raspberry Pi units.



## micro-USB Port

The Pi Pico's micro-USB provides both power, as well as communications between the Pico and a Raspberry Pi or other computer. Here you'll be able to upload programs and projects on to your Pico.

## PICO SPECIFICATIONS

- 21mm × 51mm Form Factor
- RP2040 microcontroller chip designed by Raspberry Pi in the UK
- Dual-core Arm Cortex-M0+ processor, Flexible clock running up to 133 MHz
- 264KB on-chip SRAM
- 2MB on-board QSPI Flash
- 26 multifunction GPIO pins, including 3 analogue inputs
- 2 × UART, 2 × SPI controllers, 2 × I2C controllers, 16 × PWM channels
- 1 × USB 1.1 controller and PHY, with host and device support
- 8 × Programmable I/O (PIO) state machines for custom peripheral support
- Supported input power 1.8–5.5V DC
- Operating temperature -20°C to +85°C
- Castellated module allows soldering direct to carrier boards
- Drag-and-drop programming using mass storage over USB
- Low-power sleep and dormant modes
- Accurate on-chip clock
- Temperature sensor
- Accelerated integer and floating-point libraries on-chip

The Pico is a great little project board, with plenty of potential for those who love to tinker with electronics and stretch their knowledge of everything that's connected.

## Boot Selection Switch

The small button labelled BOOTSEL, Boot Selection, will switch the Pico between two start-up states enabling you to access the Pico as a storage device on your computer or Raspberry Pi.

4 × M2.5 MOUNTING HOLES  
DRILLED TO 2.75 ± 0.05mm



### Memory & Storage

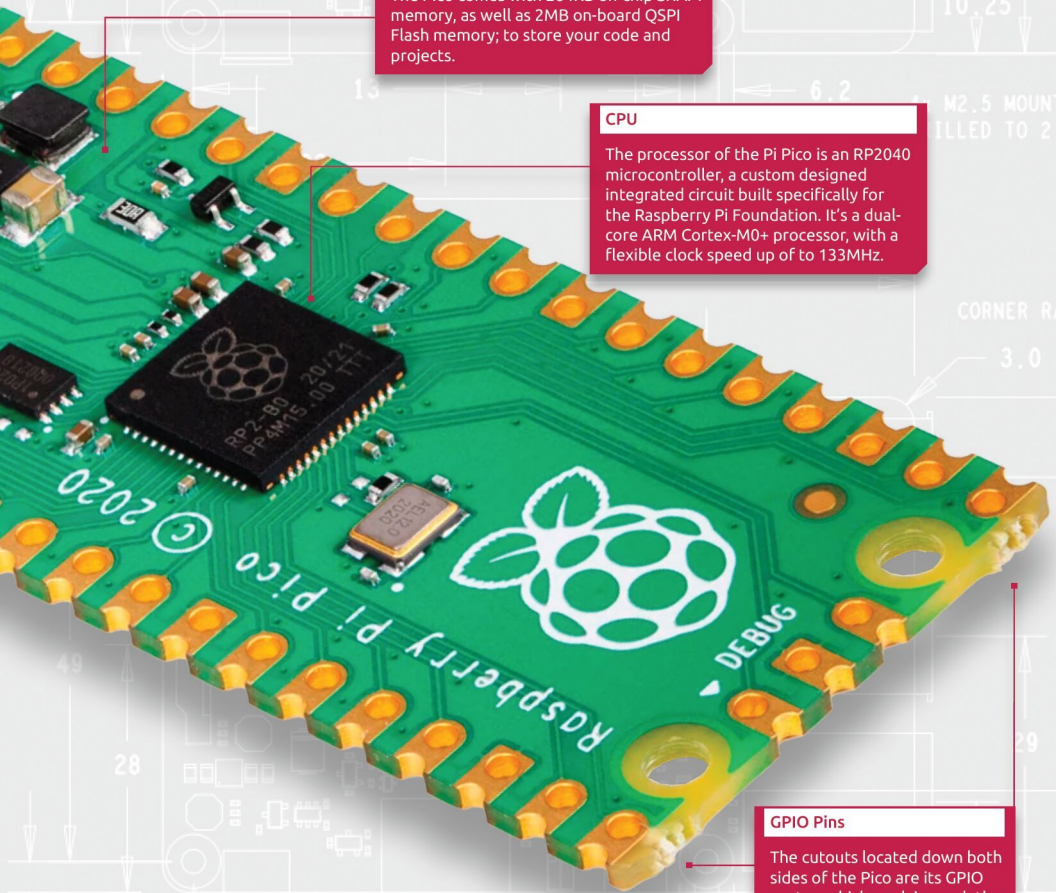
The Pico comes with 264KB on-chip SRAM memory, as well as 2MB on-board QSPI Flash memory, to store your code and projects.

### CPU

The processor of the Pi Pico is an RP2040 microcontroller, a custom designed integrated circuit built specifically for the Raspberry Pi Foundation. It's a dual-core ARM Cortex-M0+ processor, with a flexible clock speed up of to 133MHz.

### GPIO Pins

The cutouts located down both sides of the Pico are its GPIO ports – which work in much the same way as the Raspberry Pi's GPIO ports. There are 26 of them, including three analogue inputs.





# Pico Power: The Tiny Microcontroller

When is a computer not a computer: when it's a microcontroller. To be fair, a computer, by its true definition, is an electronic device for storing and processing data, which is what the Pico does. But for most users, a computer is defined as a device that sits on a desk, something you can work and play on. This is where the Pico differs.

The Pico is something entirely different from the Raspberry Pi Foundation's usual releases. Whereas the Raspberry Pi, now on version 4, is a small computer, and even the Compute Module and the Pi 400 are still just computers, the Pico is actually a microcontroller.

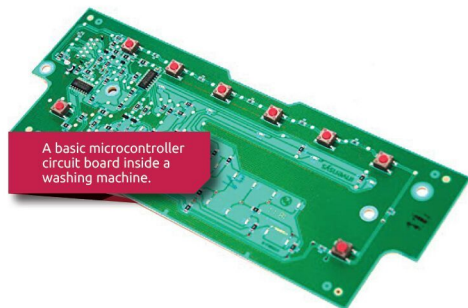
A microcontroller is a processing unit that's designed to work with programmable peripherals, and has input and output modules alongside a small amount of memory and storage capabilities. The processors are often scaled down, compared to the processor on a Raspberry Pi, or even inside your desktop computer, but powerful enough to complete basic tasks.

Examples of a microcontroller in action can be found inside a washing machine, or traffic lights. A washing machine has no need for a quad-core processor, 8GB of memory, and the ability to output to a 4K monitor.

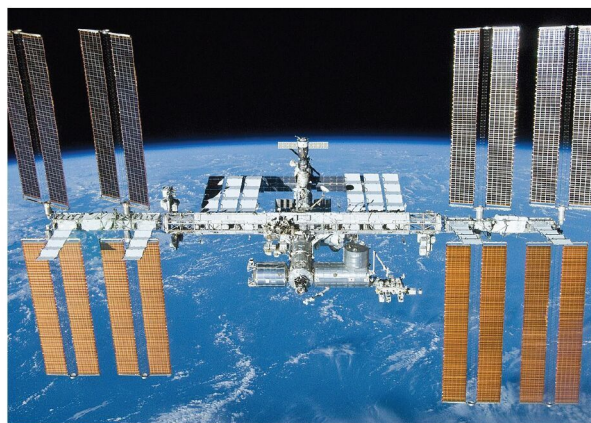
Instead, it has a set of microcontrollers, that are connected to various parts of the washing machine – such as the pump, or motor. When a signal is sent to an input on the washing machine's microcontroller, it knows – through some clever programming stored in its limited memory – that it needs to send a signal output to the valves to allow in a certain amount of water, then activate the motor for the drum and so on.

The same can be applied in the traffic light example. A powerful computer isn't necessary, so a microcontroller will activate the red, amber and green lights in sequence, or switch to red to stop traffic when someone presses the Walk button.

While it would probably be interesting to have a powerful computer inside your washing machine, it's overkill. A microcontroller takes up less space, uses less energy, and isn't prone to the kind of crashes or in need of the



A basic microcontroller circuit board inside a washing machine.



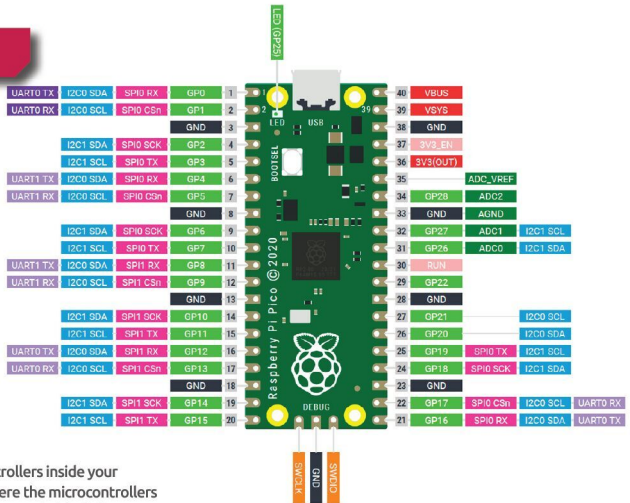
frequent updates that a traditional computer requires. They simply sit there, act on inputs that come their way, process that data, and output whatever is required depending on the input actions.

Power use is the main reason why microcontrollers are in action everywhere we look. There's often limited supply to devices, or situations. A microcontroller can work effectively with the limited amount of energy drawn from a small solar cell – even in the most overcast of northern climes. For example, on-board the International Space Station, there are thousands of microcontrollers in action: monitoring air quality, monitoring radiation levels, distributing power and keeping track on the crew. If each of these units were a more powerful computing device, the power draw on the ISS would be immense; so microcontrollers are the way forward – at least until we work out how to build a warp core with near unlimited energy, that is.



## The Raspberry Pico GPIO mapping.

- Power
- Ground
- UART / UART (default)
- GPIO, I2C, and PWM
- ADC
- SPI
- I2C
- System Control
- Debugging



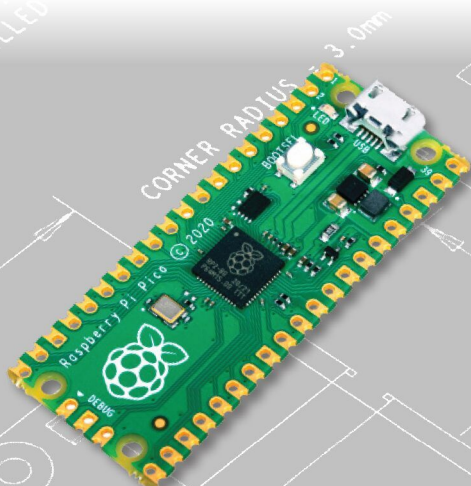
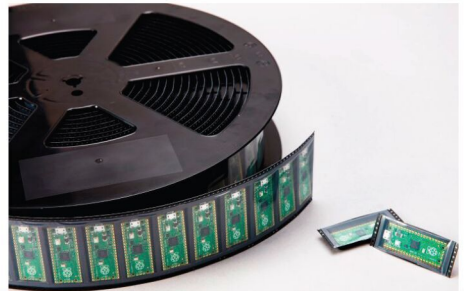
## PROGRAMMABLE PICO

The main difference between the microcontrollers inside your everyday appliances and the Pico, is that where the microcontrollers within your washing machine are pre-programmed with their instructions, the Pico isn't and can be programmed by you.

This means that you can use the Pico's USB port to connect to a computer, write some code to control something, upload the code to the Pico's RP2040 microcontroller, and watch as it does what you've asked it do.

The 26 multi-function GPIO pins, located down the long sides of the Pico, can be used to solder electronics projects to. And with some clever coding, you're able to control the input and output of devices depending on their state via the Pico microcontroller.

Each of the Pico's GPIO pins have specific uses, as you can see from the pinout chart. This means that certain electronic components soldered to the pins, can be programmed using the Pico's primary microcontroller language, MicroPython (or C++).



## REMEMBER, IT'S NOT A PI

It's worth mentioning that the Raspberry Pico isn't designed the same as, or can replace, a Raspberry Pi. This is a different type of device altogether. Where you'll use your Pi to browse the Internet, play a game, do some coding and so on, the Pico can't do that. It's designed for use with physical components and projects, such as lighting LEDs or responding to a button push, controlling motors and the like.

If you want more versatility for your projects, then the Raspberry Pi is still the perfect device to use. If you've got something specific in mind, and you want to expand into electronics and microcontrollers, then the Pico will fit the bill. But, you'll also need a computer, or Raspberry Pi, in order to program the Pico to begin with.



# The Pico and MicroPython

The beauty of the Pico is that it's a blank microcontroller, meaning that it's free from any pre-installed code or routines. This allows you to program the Pico to whatever tasks or projects you have planned for it. But first, you'll need to set it up and get it ready to use MicroPython.

The Pico uses MicroPython or C/C++ code in order to communicate with the devices it will be attached to. The code will, when uploaded to the Pico, run a specific set of instructions depending on the electronics that are soldered to the Pico. For example, in MicroPython, with an LED attached to the Pico, you would initialise and import the Pin and Timer modules, tell the Pico which GPIO the LED is attached to, then setup a routine that will use the Timer function to send a signal to the LED – let's say every couple of seconds. The end result would be a blinking LED, which goes on and off every two seconds.

Naturally, the more complex the electronics attached to the Pico are, then the more complex the MicroPython code will be to make them work in the way you want.

MicroPython is probably the best language to begin with when it comes to programming the Pico. Like Python, it's a bit easier to get your head around – when compared to C or C++ – and it's probably slightly easier to get up and running. There's also the added advantage of there being more examples of it online, so you can dip into the various sites around the Internet and take the code snippets you need to make certain devices and peripherals work.

**Getting started with MicroPython**


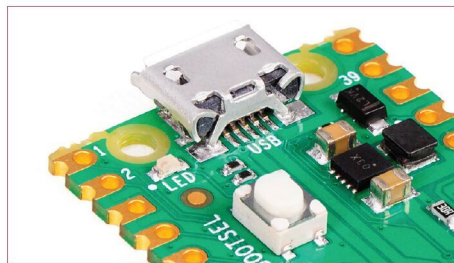
**Drag and drop MicroPython**

You can program your Pico by connecting it to a computer via USB, then dragging and dropping a file onto it, so we've put together a downloadable PDF file to help you install MicroPython more easily.

1. Download the MicroPython UF2 file by clicking the button below.
2. Push and hold the BOOTSEL button and plug your Pico into the USB port of your Raspberry Pi or other computer. Release the BOOTSEL button after your Pico is connected.
3. It will mount as a Mass Storage Device called RP2-RP2.
4. Drag and drop the MicroPython UF2 file onto the RP2-RP2 volume. Your Pico will reboot. You are now running MicroPython.

You can access the REPL via USB Serial. Our MicroPython documentation contains step-by-step instructions for connecting to your Pico and programming it in MicroPython.

[Download UF2 file](#)

## MICROPYTHON

MicroPython is a Python 3-based coding language that's designed to be executed on the RP2040 microcontroller – the one that's powering the Pi Pico. It's a highly efficient language, and thanks to the Pico's larger than average memory and capacity – for a microcontroller – it's the ideal language to start learning. If you already have some experience with Python 3, then MicroPython should be relatively easy to follow; since they're essentially the same. If you've never used Python before, then we'd recommend you take a look at one of our coding titles at, <https://bdmpublications.com>.

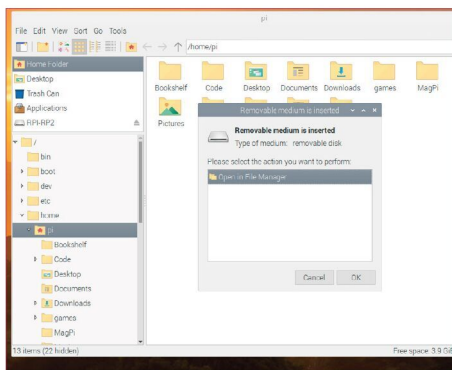
Before you can use MicroPython, though, you'll need to flash the Pico with the latest version of MicroPython. This means downloading the latest version of MicroPython and uploading it onto the Pico. The process is remarkably easy, however.

Start by connecting one end of a USB cable to the Pico. Then press and hold the BOOTSEL button on the Pico, while at the same time plugging the other end of the USB cable into your computer, or a Raspberry Pi. Count to five, then let go of the BOOTSEL button.

Within a few seconds the computer, or Raspberry Pi, will display a notification that an external, or removable, drive has been attached – in the same fashion as when you plug in a USB flash drive.

If you open your File Manager or Explorer window, and navigate to the newly installed Pico storage, you'll see two files present: INDEX.HTM and INFO\_UF2.TXT.

The INFO\_UF2.TXT file contains information regarding your Pico, and the INDEX.HTM is a web-based page that contains all the information you'll need to get the Pico connected and installed with MicroPython.







Alternatively, you can open a browser and navigate to <https://www.raspberrypi.org/documentation/pico/getting-started/>. Scroll down until you get to the **Getting started with MicroPython** section, and follow the on-screen instructions. Click on the **Download UF2 File** button – at the bottom of the Getting Started info box.

Once the UF2 file has been downloaded, locate it in the Downloads folder on your computer, and click and drag it into the Pico storage drive; as you would drag and drop a file from your computer to a standard USB Flash drive.

After a few seconds, the Pico storage drive will disappear – you may get a warning stating that the device wasn't unplugged correctly, but ignore that. This process has now 'flashed' the Pico with the MicroPython firmware. The Pico disappearing from the File Manager means that it has rebooted and is now in MicroPython mode. This is essentially all you need to do for the moment. The Pico is now ready to start coding on to using MicroPython.

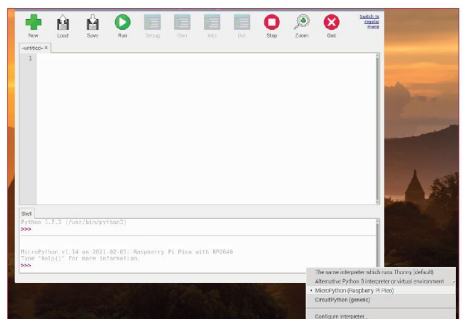
## GET READY TO PICO CODE

While the Pico is in MicroPython mode, it doesn't do much. You need to get access to the MicroPython layer through an Integrated Development Environment (IDE). The Raspberry Pi Foundation has adopted Thonny as its default IDE for use with MicroPython, since it's already built-in to the Raspberry Pi OS; as found on the Pi 4, Pi 400 and so on.

What you need to do is open Thonny – make sure the Pico is still connected to your Raspberry Pi – then when it's loaded, click on the Python label in the bottom-right of the Thonny window; it's going to say Python followed by a version number, such as 'Python 3.7.3'.

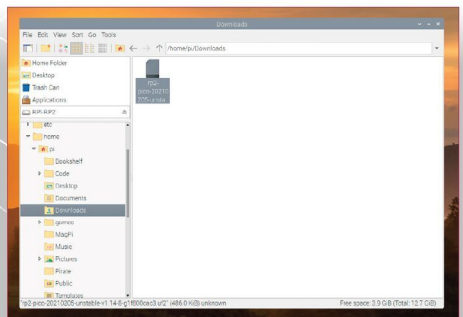
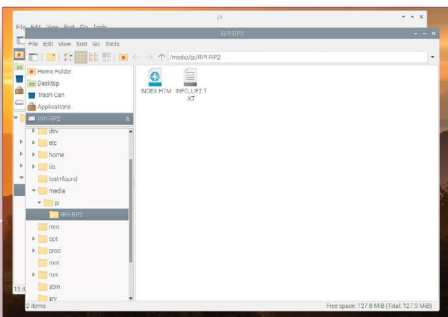
This is the current interpreter that Thonny is using for the process of taking the code you enter and converting it into instructions that the computer can understand and execute. When you click the Python label, you'll notice a small menu appear, in which will be listed 'MicroPython (Raspberry Pi Pico)'. Click the MicroPython entry. This will change the Shell window, the lower-portion of the Thonny app, to read MicroPython, followed by the version number.

Thonny and the Pico are now ready to be coded on to. Since MicroPython is based on Python 3 – with some scaled-down



elements due to the fact that it's designed to work on a much-lesser processor – most of the usual Python syntax command will work, such as `print("Hello, world!")` and so on. You can also run loops and other such repetitive routines, as well as store variables.

The Shell portion of Thonny can be used to 'talk' directly to the Pico's MicroPython interpreter, whereas the blank upper-section of Thonny can be used to write multiple lines of code which can then be saved to the Pico and executed later.



## GET DISCOVERING

The world of the Pico is now open to you. From here, you can run through many of the examples that are available online, connect electronic elements to the Pico and have them function through MicroPython, or even stretch your learning and discover how the Pico works with the C and C++ programming languages.



# Pico Examples and How To

With the Pico now ready to accept MicroPython code, it's useful to understand some of the basics and how to get started. We've put together some foundation examples with MicroPython, to help you begin to get to grips with this wonderful microcontroller.

When you flash the Pico with the latest MicroPython the Pico will enter a wait state, waiting for instructions from you through MicroPython. You can easily start to upload a pre-created Python file, a .py file, on to the Pico's storage, however, you can start to 'talk' to the Pico immediately via REPL.

REPL, pronounced 'ripple', stands for Read, Evaluate, Print, Loop. Read means that MicroPython is waiting for you to enter code; Evaluate will interpret the code and begin to execute it; Print displays the results of the last executed line of code; Loop returns to the start, prompting you for more code.

You can access the Pico via the command line, through a tool called Minicom, but since we've already got Thonny up and running, it makes sense to stick with the path of least resistance and code the Pico (communicate with it) using Thonny's interface.

## HELLO, WORLD!

The most popular first lines of code is the good old "Hello, world!". In Thonny, click into the lower Shell section, to the right of the three right-facing greater-than signs '>>>'. Now, enter the following:

```
print("Hello, world!")
```

Users of Python will undoubtedly recognise the syntax, but if you've never coded with Python before, then let's explain what's going on in this simple line.

The print command, as you probably assume, is used to print, or display, something to the screen for the user to read. If you want to say something to the user, then you'll need to enclose your text inside a pair of brackets and quotation marks. In this example, we're displaying the words **hello**, **world!**. When the MicroPython interpreter reads the Print command it will naturally assume that something is being displayed, and look for a number of acceptable outcomes. The first bracket and opening quotation mark means there's going to be some text, so the interpreter will look for a closing set of quotation marks and bracket.

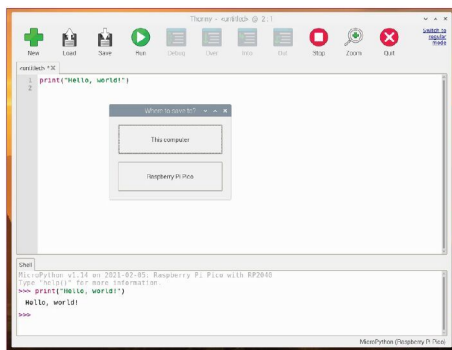
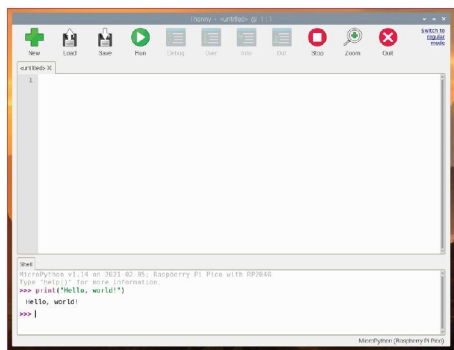
Press Enter, and the words **hello, world!** will appear in the Shell section of Thonny.

What you've just done is communicate directly with the Pico, and through MicroPython on the Pico, it's recognised the command you've entered and acted on it.

Just for fun, enter the command again, but this time intentionally miss out the last bracket. You'll notice that the Shell doesn't want to continue without the bracket, so therefore it's an error. While using MicroPython, or most other programming languages, you'll need to be careful what you type. The slightest mistake can have dire consequences to your code – or, more than likely, it'll just stop them from working.

For the next example, click the upper part of Thonny, where the number '1' is. Now enter the Hello, world code again. When you hit Enter this time, you'll notice that instead of running the code through the Pico, Thonny has created a second line and is waiting for more input.

While using the Shell is great, it's not so good for creating multi-line code. This is where the upper section comes in. To execute the code in the upper section you'll need to do the following: Click the Run button; this opens a new window asking for a save location (This computer or the Raspberry Pi Pico). This is because any code that's been entered in the upper section of Thonny needs to be saved first, as a .py file, before it can be executed.





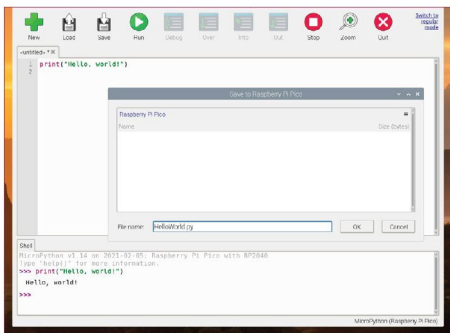
Select the Pico as the save location, then when asked, create a name for the .py file – in this case call it HelloWorld.py – and click the OK button.

As soon as you've saved the file, it will automatically run and you'll see the output from the code displayed in the Shell box.

Alternatively, you can open a browser and navigate to <https://www.raspberrypi.org/documentation/pico/getting-started/>. Scroll down until you get to the Getting started with MicroPython section, and follow the on-screen instructions. Click on the Download UF2 File button – at the bottom of the Getting Started info box.

Once the UF2 file has been downloaded, locate it in the Downloads folder on your computer, and click and drag it into the Pico storage drive; as you would drag and drop a file from your computer to a standard USB flash drive.

After a few seconds, the Pico storage drive will disappear – you may get a warning stating that the device wasn't unplugged correctly, but ignore that. This process has now 'flashed' the Pico with the MicroPython firmware. The Pico disappearing from the File Manager means that it has rebooted and is now in MicroPython mode. This is essentially all you need to do for the moment. The Pico is now ready to start coding on to using MicroPython.



## BLINKING LEDS!

MicroPython, as with Python, uses modules to enhance the code you're writing. A module is simply further code, but when called upon by your code it will operate to specific parameters. For example, there are modules available that are designed to handle graphics, while others handle complex mathematical operations. Using these modules saves you from having to invent the code and place it among the many lines you're typing in. All you need to do is call up the module, and pass any variables and parameters through the module to get the result you want.

In MicroPython, and in the Pico's case, there are modules that can be called upon that handle the physical hardware side of things. The Pico-specific modules will handle input and output through the GPIO ports, or utilise some of the Pico's inherent sensors. For example, here's some code that will light up the built-in LED on the Pico:

```
from machine import Pin
led = Pin(25, Pin.OUT)
led.on()
```



Type it into the upper section of Thonny, save it as LED.py (for example) and run the code. The LED will light up. But, let's explain what's going on here.

To begin with we have, `from machine import Pin`. This means that the code is going to open the built-in module called Machine, of which a certain class within it is called Pin. This particular set of external code is designed to 'talk' to and handle the input and output of the Pico's GPIO pins.

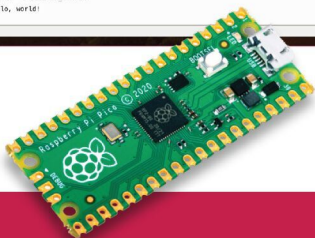
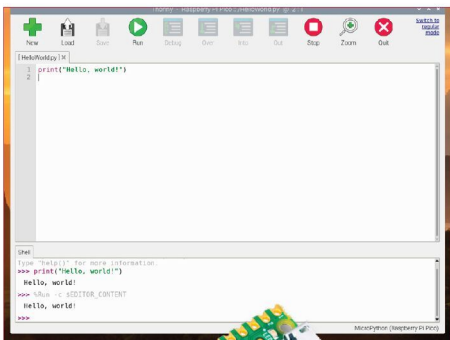
The next line, `led = Pin(25, Pin.OUT)`. What we've done here is create a variable called `led`, which we've told to hold the information regarding the Pin class, in particular the number 25, as an Output. The GPIO pin on the Pico is hardwired to the LED, so essentially we've told MicroPython that Pin 25 is going to have Output data sent to it, and it's going to be called led.

The final line, `led.on()`, is a trigger that sets Pin 25 Output to a high value – in this case, On. The result is that the LED on the Pico will be turned on. Try and alter the code, so the last line reads:

```
led.off()
```

As you would expect, this turns the LED off. Now try:

```
led.toggle()
```





# Raspberry Pi is Good For You!

What `toggle()` does is every time the code is executed, if the LED is on it will switch off, and if it's off it'll switch on. Interesting stuff, isn't it.

Let's expand our LED code further and include a loop with another kind of module:

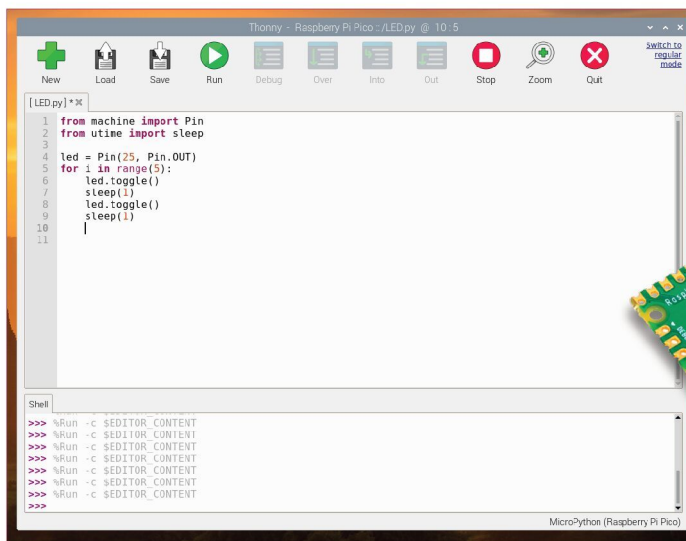
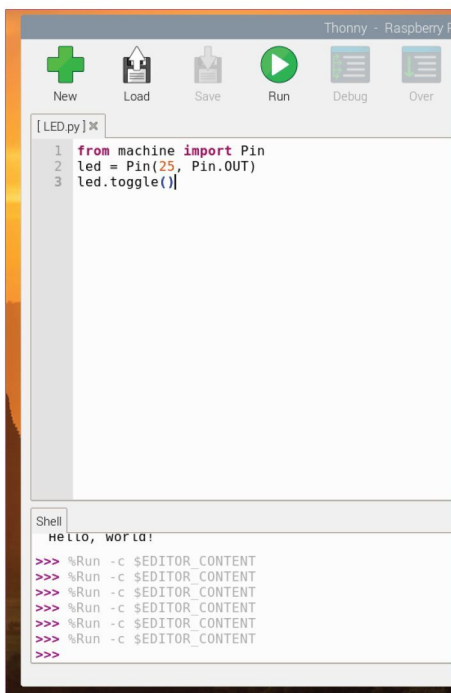
```
from machine import Pin
from utime import sleep
led = Pin(25, Pin.OUT)
for i in range(5):
    led.toggle()
    sleep(1)
    led.toggle()
    sleep(1)
```

When you execute this new code, it will blink the LED five times – based on whether the LED was previously on or off using the `led.toggle()` syntax we looked at a moment ago. Let's take a moment to break down the new elements of the code, though.

**From `utime import sleep`**, as you probably suspect, this is using a new module called `utime`, from which we want to use the `sleep` class. `utime` holds code that can be used for timing, and the `sleep` class enables the code to pause for a specific number of seconds.

**For `i in range(5)`**, is a new line which is the beginning of a loop. Basically, we've asked MicroPython to count to 5, and anything after the colon is repeated five times. Providing what code we've entered in the loop is okay, it'll do it five times before stopping. Note: The indent after the colon. This means that the code in the indent is directly controlled by the for loop. Indents in Python are as important as spelling and getting the correct characters in order.

The final new element is `sleep(1)`, which pauses the code for the count of one, or one second. Feel free to mess around with the code you've created. Extend the number of seconds, use `led.on` and `led.off` instead of `toggle`, and extend the for loop to more than five.





## TEMPERATURE READINGS

The Pico also comes with a built-in temperature sensor, and is connected to one of the Pico's analogue pins. The analogue pins on the Pico are actually called ADC's – or Analogue to Digital Converter – whereas the GPIOs are digital. The difference is that a digital pin reads only two states: High or Low (or One and Off), and an analogue pin can read a range of values from zero 65535 (through the use of MicroPython, the Pico's ADC really reads from zero to 4096). This means that anything that's read through the analogue pins can be from 0 to the value of 65536, which makes it perfect for a temperature sensor.

The temperature on the Pico is hardwired into ADC4 – or analogue pin 4. So we can extract the data from the temperature sensor using the following code:

```
from machine import ADC
tempensor = ADC(4)
temperature = tempensor.read_u16()

print (temperature)
```

There's a lot going on here, and the displayed number that appears in the Shell probably won't make much sense, so let's have a look at the code.

First we're using the machine module and from it the ADC class. Then we've created a tempensor variable which will pull the data from ADC pin 4 – which is the Pico's temperature sensor. The third line of the code creates a temperature variable, which stores the value of ADC4 and applies a 16-bit data range to it so we can have a value from 0 to 65535. And Finally, we're printing the currently held value of temperature in the Shell.

Now here's the part that will make your head hurt.

The Pico's microcontroller runs at 3.3 volts. So when there's a full 3.3V being applied, the ADC will read a value of 65535, and when there's no voltage it will read zero. The temperature sensor on ADC4 works by sending voltage to ADC4 that is proportional to the environmental temperature. So if the environment the Pico is in is 27 degree Celsius, the Pico sends a voltage of 0.706V. So, using the 3.3V maximum at the value of 65535 and the operational specifications of 27 degrees Celsius as being a voltage of 0.706V, we can work out code that will convert the original value you got from the temperature variable into degrees Celsius.

Before our brains dribble out, here's the rest of the code:

```
from machine import ADC

tempensor = ADC(4)
temperature = tempensor.read_u16()

volts = 3.3 / 65535
temperature = temperature * volts

celsius = 27 - (temperature - 0.706) / 0.001721

print (celsius)
```

Let's run through the code. We've created a volts variable which takes the maximum 3.3V and divides it by 65535, then we've taken

```
from machine import ADC
tempensor = ADC(4)
temperature = tempensor.read_u16()
print (temperature)
```

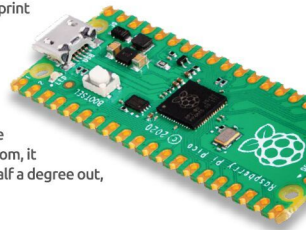
```
Shell
42424
>>> from machine import ADC
14212
```

```
from machine import ADC
tempensor = ADC(4)
temperature = tempensor.read_u16()
volts = 3.3 / 65535
temperature = temperature * volts
celsius = 27 - (temperature - 0.706) / 0.001721
print (celsius)
```

```
Shell
42424
>>> from machine import ADC
14241
>>> from machine import ADC
20.4908
```

the already read temperature variable and multiplied that by the volts value. We've then created a Celsius variable, which we've used the above numbers to store the value of 27, minus the temperature value, minus 0.706 and divided by 0.001721. This will give us a reasonably accurate conversion of volts to Celsius, which the code will then print into the Shell section of Thonny.

As you can see from the screenshot, the room we ran this code in was a balmy 20.49 degrees Celsius. When we looked at the temperature on the digital thermostat in the same room, it read 21 degree Celsius, so only half a degree out, which isn't too bad.



## THE WORLD OF THE PICO AWAITS...

This is only a small example of the type of projects and coding that can be achieved with the Pico and MicroPython. There are numerous extras, such as motors, sensors, LEDs and even LED screens that can be attached to the Pico's GPIO ports with some neat soldering and the use of a handy electronics kit.

What we've looked at here is a taster, and the foundations on how to call up the Pico's GPIO pins and how to use MicroPython in Thonny. From here, we'd recommend you learn more about how the Raspberry Pi functions as well as some more Python coding, which will give you a better insight into how the Pico and MicroPython will work.

Have fun, and let us know what Pico projects you've created.



# Pico Projects & Ideas

There are some interesting projects you can apply your Pico to. Thanks to its combination of processing power, GPIO pins, and MicroPython language, the only real barrier is your own imagination.

## PICO POWER!

Here are ten ideas and projects that you could look into achieving with your Raspberry Pi Pico. You may need to enhance your electronics skills, but there's plenty of help online if you hunt around for it.

### ROBOTICS

Can you create a Pico-powered microcontroller to work different aspects of a robot? You could use the

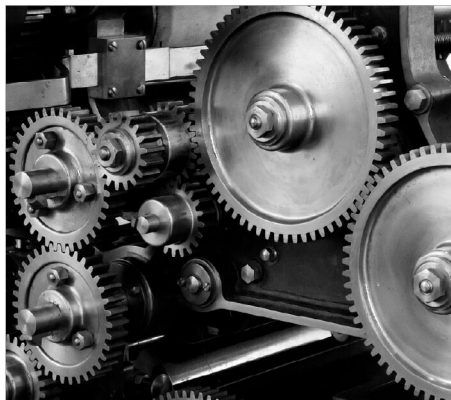
Pico to drive the robot, or control an element on it depending on where it is in your home.



### MOTORS

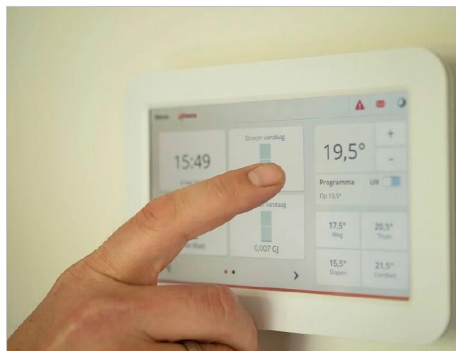
In addition to the robotics idea, a Pico can be used to control motors. Again, depending on the input to the

Pico, the output can spin up a motor and move an object, stop it, or even power something else.



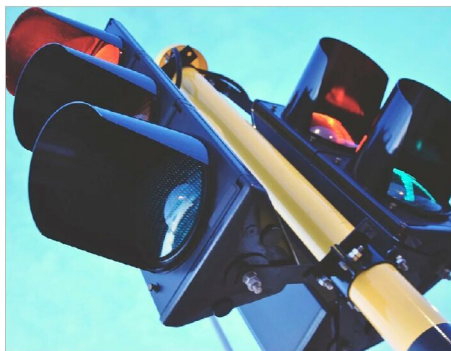
### TEMPERATURE GAUGE

The Pico's RP2040 processor has a built-in ADC to convert analogue to digital signals. It also has a built-in temperature sensor, so it's possible to utilise the Pico as a temperature gauge.



### TRAFFIC LIGHTS

Why not hook up some coloured LEDs, and use the Pico microcontroller as a set of traffic lights. You can even incorporate a button, that when pushed will act as a Walk button and stop the traffic for pedestrians.





### MOTION SENSORS

Using an infrared sensor, you can wire up and solder your own motion detection sensor. Which means you can create a burglar alarm, or include it with the robot idea to allow some form of autonomy.



### BUTTON PUSH GAME

Using buttons and LEDs, you can create a simple button push reaction game, for multiple players. Perhaps you could attach a buzzer that signifies the winner or loser.



### HOME AUTOMATION

Most home automation tasks are run through microcontrollers, so the Pico is suitably placed for those who have setup some advanced home automation devices. Elements such as turning the heating or lights on, or opening the garage door, for instance.



### MOON TRACKING TELESCOPE

If you've got the know-how, then the Pico, together with a phototransistor sensor, and a telescope, could be used to create a scope that can track the passage of the Moon through the night sky.



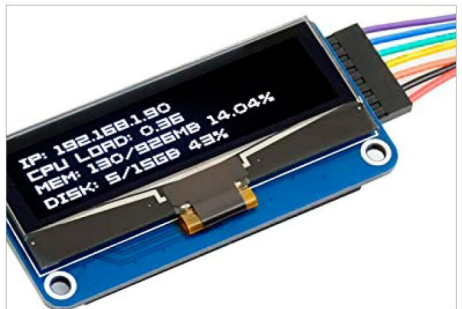
### NATURE PHOTO CAPTURE

With the infrared motion sensor, you could attach the Pico to an outdoor camera – or camera within a weather-proof container – to activate when something passes the sensor; capturing that neighbour's pesky cat in the act!



### ATTACH A DISPLAY

There are a number of Pico-built RGB LED displays available online. You can use one to display messages, create some form of interactive game, or display data that's coming from one of the input GPIOs attached to a sensor.









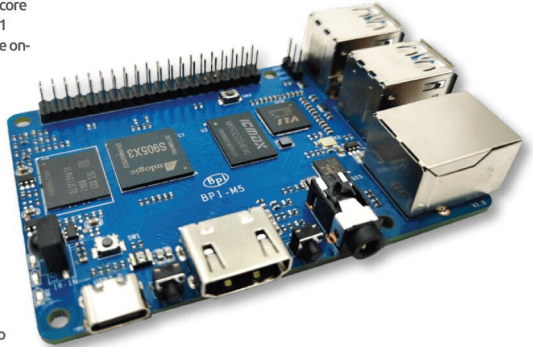
## BANANA PI

Launched in 2014, the Banana Pi is a Raspberry Pi-compatible board that has seen many releases since its initial conception. These are ARM-based SBCs developed in China and sold via a number of international distributors.

The latest version of the Banana Pi, the BPI-M5, looks remarkably similar to the Raspberry Pi 3B. It features an Amlogic S905X3 quad-core Cortex-A55 CPU running at a decent 2GHz, together with a Mali G31 GPU. There's 4GB of LPDDR4 memory, 16GB of eMMC flash storage on-board and a MicroSD slot with support for up to 256GB cards.

Connectivity comes in the form of a gigabit Ethernet port, full-sized HDMI, 3.5mm audio jack, four USB 3.0 ports and a USB-C port for power input.

The I/O mimics the Raspberry Pi's 40-pin GPIO, but there's no extra connectivity for cameras and such; as found on the Raspberry Pi.



The Banana Pi M5 is a tad more powerful than the Raspberry Pi 4, and the 4GB memory option copies the more common Pi available. Although you will have extra RAM if you were lucky enough to get hold of the 8GB memory Pi. You can install both Linux or Android on the M5, and it's even Raspberry Pi OS compatible.

It costs \$53 – at the time of writing – which equates to around £38; so not too far from the cost of a Raspberry Pi.

## ROCK PI

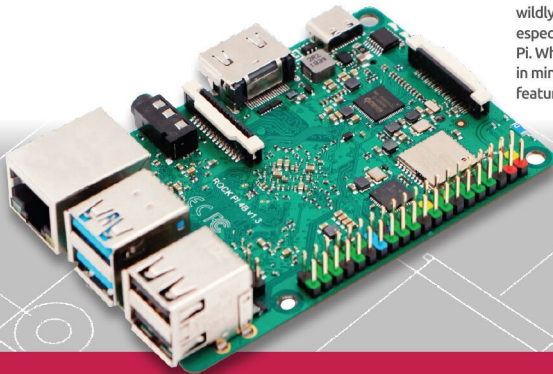
Developed by Radxa, the Rock Pi has been a direct competitor to the Raspberry Pi for some years now; and it's certainly not looking like it's slowing down any time soon.

The latest version is the Rock Pi 4 Model C, which sports a hexa-core CPU with a dual-core Cortex-A72 at 1.8GHz and a quad core Cortex-A53 at 1.4GHz together with a Mali T860MP4 GPU. Models A and B feature 1GB, 2GB and 4GB memory options; but the Model C comes with only 4GB of LPDDR4 memory. There's an eMMC flash module available, in 8GB, 16GB, 32GB, 64GB and 128GB options, as well as a MicroSD card slot with support for 128GB cards, and an M.2 connector for M.2 SSDs.

As for connectivity, there's HDMI, gigabit Ethernet, two USB 3.0 ports, two USB 2.0 ports, 3.5mm audio jack, USB-C for power, and Bluetooth and Wi-Fi.

For project work, there's a 40-pin GPIO, as well as a MIPI CSI slot for camera attachments. It's also interesting to note that the Ethernet port supports Power-over-Ethernet (with an additional HAT). Not bad for a board that measures 85mm x 54mm.

The Rock Pi Model C costs \$59 or £42, and while a little more expensive than the Raspberry Pi, it out-performs the Pi in nearly every benchmark.



## ARE THEY WORTH IT?

The question of whether to use a different board other than the Raspberry Pi is purely up to you as an individual and what you're planning on using the SBC for. For most folk, who buy a Raspberry Pi for emulation or as a media server – which are the two most popular projects used by users – then they'd probably get more mileage out of the aforementioned boards. True, the expense is considerable in the case of the Udo0, but it'll do a far better job of processing than the Raspberry Pi will ever do.

The real argument comes down to cost. If you're willing to pay a little more for the extra performance, then the alternative SBCs will suit you fine. But when you compare them to the Raspberry Pi, the Pi does seem to come on top in terms of cost-versus-performance.

Needless to say there are countless more SBCs available. Some are smaller, even more powerful than the Pi; and cost does vary wildly from one product to the next. For most users, though – and especially for the beginner – the best bet is to stick to the Raspberry Pi. When your skills improve, and you have more specific projects in mind, then look to the other boards for the specifications and features you need.



# Raspbian: The Complete Operating System

The main Raspberry Pi operating system is Raspbian, which is a Linux-based OS. While the Pi is primarily an educational and project board, with Raspbian, it does become a fully-functional desktop computer.

## POWERING THE PI

The Raspbian OS has been powering the Raspberry Pi since its release back in 2013. At its core, Raspbian is a Linux operating system, based on the hugely popular Debian flavour of Linux.

Mike Thompson and Peter Green created Raspbian in 2012, to work on the low-performance ARM CPUs found in the early Pi models. Toward the end of 2014, Raspbian was further improved to work on the then new model Pi, the Raspberry Pi 2; utilising the 3.18 Linux kernel while still having Debian 7 (codenamed Wheezy) as the base.

Mid-way through 2015, Raspbian started using Debian 8 (Jessie) as the base, with kernels 4.1, 4.4, and 4.9.

Kernel 4.9 was also used from 2017's Raspbian release, which has since used Debian 9 (Stretch) as the foundation distribution.

More recently, from November 2018, the Raspberry Pi Foundation has split the Raspbian OS project into three separate distributions. Although these are essentially the same core operating system, those available now are a Lite, Minimal and Full desktop set of images.

One of the primary reasons for the three-way split is due to the overall size of the continually improving Raspbian. In its early years, the Raspbian desktop image size was a little under 1GB. This image contained everything the Pi user of the time needed to get their projects up and running, as well as enjoying third-party applications such as LibreOffice. Slowly, as the Pi itself evolved, the Raspbian image increased. These days the image weighs in at around 1.8GB and it's likely to continue to expand over the coming years as new software is improved on and added, alongside changing hardware. However, the three-way split addresses the issue of an ever-expanding OS image, while still enabling the user to get the most from their Raspberry Pi.

The Lite version image comes in at roughly 350MB in size. While it may seem like a logical choice to opt for the Lite version, it's not recommended for every user. The Lite version basically installs without any of the software you expect from the previous versions of Raspbian; it doesn't even come with the graphical desktop. This makes it an incredibly streamlined and efficient version of the OS that's ideal for those who run headless Pi servers (a Pi without a monitor, keyboard or mouse attached, that the user remotely connects to in order to input

commands). You will need to know your Linux commands in order to get the most from the Lite version, so for most beginners it's not a good place to start. The more advanced users will, however, be able to setup their Pi projects without the unnecessary baggage of all the other applications clogging up the Pi's precious system resources. It's worth noting that the Lite edition can be made up to a fully functional desktop version by installing all the relevant software manually.

The Minimal edition is now the default version recommended for most new Raspberry Pi users.

This version is around 1GB in size and includes the PIXEL desktop, Chromium web browser, new hardware accelerated VLC player and Python. It does not include LibreOffice, Scratch, Sonic Pi and many of the other tools, programs and applications that previous versions of Raspbian included. It's a perfect start for most users and those who are more familiar with the workings of Raspbian and Linux can easily add more apps later, as they require.

The Full desktop version is 1.8GB in size (at the time of writing) and as you imagine, contains pretty much everything you can squeeze into Raspbian. You get the PIXEL desktop, LibreOffice, VLC, browser, games, programming resources and much more.

So, which version should you download and use on your Pi? Normally, as recommended by the Raspberry Pi Foundation, most new Pi users are best opting for the Minimal desktop; but that depends greatly on what your plans are for using your Raspberry Pi. We recommend new users begin with the Full desktop version, as this is the version that gives you the complete Pi experience along with all the associated software; even if you only use a small percentage of it. In fact, this entire book was written using the Full desktop version on a Raspberry Pi 3 Model B+.

As you learn more about the Raspberry Pi, Raspbian and Linux and begin to form your Pi into a working project, you can easily downgrade the version of Raspbian to reflect your use. One day, you may even opt for the Lite, command line only version, occasionally connecting to it remotely and issuing a few commands. As with most things Pi-related, it's purely up to you.

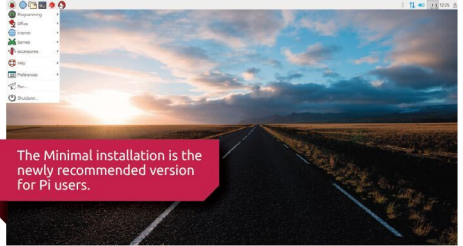




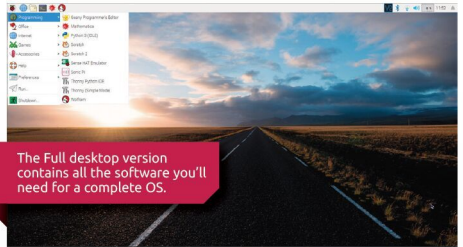
```
Starting Login Service...
[ 1.00 ] Started Regular background program processing daemon.
[ 1.00 ] Started daily apt upgrade and clean activities.
[ 1.00 ] Reached target Timers.
Starting SaneBackend Sane Card State...
Starting triggerhappy global hotkey daemon...
[ 1.00 ] Started System Logging Service.
[ 1.00 ] Started triggerhappy global hotkey daemon.
[ 1.00 ] Started Bluetooth RFCOMM socket state.
[ 1.00 ] Started SaneBackend Sane Card State.
[ 1.00 ] Started Pulse network interface.
Starting Load/Save IF Kill Switch Status...
[ 1.00 ] Started Login Service.
[ 1.00 ] Started Load/Save IF Kill Switch Status.
[ 1.00 ] Started LSB: mountpoint and run a swap file.
[ 1.00 ] Started LSB: Switch to manual cpe mode when (unless shift key is pressed).
[ 1.00 ] Started Confuser Bluetooth Pdaless connected by UART.
Starting Bluetooth services...
[ 1.00 ] Started Bluetooth service.
[ 1.00 ] Reached target Bluetooth.
Starting Hostname Service...
[ 1.00 ] Started hostname service.
[ 1.00 ] Starting dhcpcd on all interfaces.
[ 1.00 ] Reached target Network.
Starting rfc226c Local Compatibility...
Starting Networkd User Session...
[ 1.00 ] Started rfc226c Local Compatibility.
[ 1.00 ] Started networkd Local Compatibility.
[ 1.00 ] Started Networkd User Sessions.
Starting systemd-logind finished up...
Starting Terminate Plymouth Boot Screen.
Starting Terminate Plymouth Boot Screen.

Raspbian (GNU/Linux) raspberrypi login
raspberrypi login
```

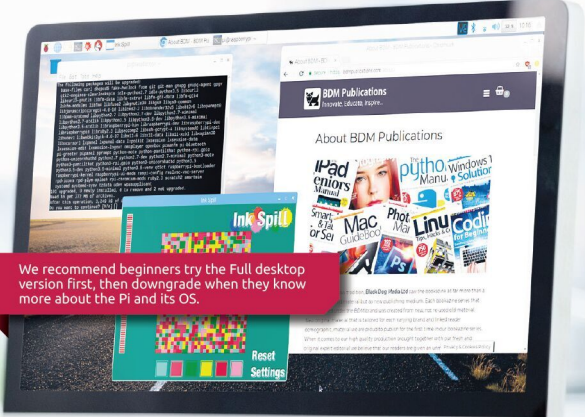
The Lite version doesn't even have a graphical desktop; it's command line only.



The Minimal installation is the newly recommended version for Pi users.



The Full desktop version contains all the software you'll need for a complete OS.



We recommend beginners try the Full desktop version first, then downgrade when they know more about the Pi and its OS.

# Which Pi is Right for Me?

With several models of Raspberry Pi available to purchase, you can be forgiven for any confusion over which model you should buy. So, to help save you time and money, let's see which Pi is best for your needs.

## THE POWER OF PI

With many different versions across four generations of models, the Raspberry Pi is certainly a busy little board. But while each Pi offers something slightly different, for the newcomer it's a confusing medley of hardware specifications and model numbers. Which, Pi, then is best for you?

Naturally, that questions depends greatly on what it is you want to do with your Raspberry Pi. If you have a particular project in mind, such as a home media centre server, then you'd probably opt for the more powerful and hardware-capable model of Raspberry Pi. If you want to setup a wireless security camera project, using the Raspberry Pi as the core hardware, then perhaps the Pi Zero W would be a better fit. To begin with, let's have a brief look at the models available.

## FIRST GENERATION:

Although now quite old, in computing terms, the first generation Raspberry Pi models are still available to purchase. These are the Raspberry Pi 1 Model A+ and Pi 1 Model B+.

The Pi 1 Model A+ was released in November 2014, and replaced the original Model A. It features the now standard 40-pin GPIO, Micro SD Card, lower power consumption, and better audio circuitry. It's also a smaller package than the original Pi, while having a 700MHz processor and 512MB of memory.

While a worthy Pi, it does lack the built-in networking of its newer model counterparts; both Ethernet and Wi-Fi. The processor is somewhat lacking in the performance department, which can be an important factor if you are thinking of using this model for any serious CPU-related projects. In short, it's probably worth avoiding the Pi 1 Model A+.

The Pi 1 Model B+ is a far more capable design than its sibling Model A+. Released in July 2014, the Pi 1 Model B+ boasts built-in 100Mb Ethernet and four USB 2.0 ports. It does, however, still feature the same lacklustre processor as the Pi 1 Model A+, but it's extra USB and Ethernet ports make up for any lack in available hardware.

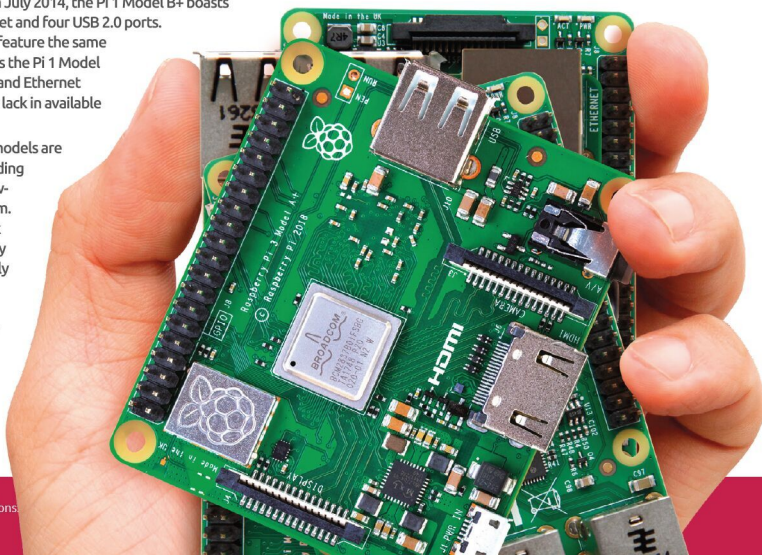
The first generation models are decent enough, providing you're doing some low-level project with them. Although you can pick them up from a variety of online stores slightly cheaper than the current models, they probably aren't worth the savings.

## SECOND GENERATION:

There are two Pi 2, second generation models available: the Model B and the Model B version 1.2. Although, you may be hard pressed to find an original Pi 2 Model B (without the 1.2 version), so we will solely refer to the Pi 2 Model B as the 1.2 version that was released mid-2016 instead.

The Pi 2 Model B offered the user a more powerful Pi experience. With 1GB of memory, an upgraded 900MHz quad-core ARM Cortex-A7 processor, and an improved HDMI port. As with the previous first generation Model B, this version also comes with a built-in 100Mb Ethernet port and four 2.0 USB port hub.

Overall, this is a better choice of Raspberry Pi over the first generation models. The extra processing power, even though it's only 200MHz, does make a difference, alongside the extra 512MB of memory, too. If you find one that's significantly cheaper than a more recent Pi, then it's worth considering.





## THIRD GENERATION:

The third generation Pi models are significantly more capable than the previous, but they may cost slightly more than the second generation model. However, you do get more Pi for your money.

The first third generation model to be released was the Pi 3 Model B, in February 2016. A newer quad core 1.2GHz Broadcom BCM2837 64-Bit processor, 1GB of memory, and a 4-pole stereo output and composite video port meant that this was the power-Pi to have. It's still a very good model to use, and can do most of the tasks its younger sibling, the Pi 3 Model B+ can do.

The second of the third generation models released was the upgrade to the Model B, the Pi 3 Model B+, released in March 2018. With an impressive 1.4GHz quad core processor, 1GB of faster LPDDR2 memory, dual frequency built-in Wi-Fi and a gigabit Ethernet port, the Pi 3 Model B+ is the most powerful Raspberry Pi to date, and will be capable of running all your projects without any problems.

The newest member of the third generation Pi models is the replacement for the second generation A+. Released in November 2018, the Pi 3 Model A+ enjoys the same processor as the more powerful Pi 3 Model B+, but has half the available memory, at 512MB. It also loses the Ethernet port, and three of the USB ports, but it does boast dual band Wi-Fi, and a far smaller footprint than previous models.

Out of the third generation models, the ones to look out for are the Model B+ and Model A+. The B+ will give you a more powerful Pi experience, but it costs slightly more. Whereas the Model A+ is much smaller, but lacks the extra memory and additional USB ports.

## FOURTH GENERATION:

Introduced on 24th June 2019, the Raspberry Pi 4 Model B is a significant leap in terms of Pi performance, and hardware specifications. It was also one of the quickest models, aside from the original Pi, to sell out.

With a new 1.5GHz, 64-bit, quad-core ARM Cortex-A72 processor and a choice of either 2GB, 4GB or 8GB memory versions, the Pi 4 is one-step closer to becoming a true desktop computer. In addition, the Pi 4 was launched with the startling decision to include dual-monitor support, in the form of a pair of two micro-HDMI ports. You'll also find a pair of USB 3.0 ports, Bluetooth 5.0, and a GPU that's capable of handling 4K resolutions and OpenGL ES 3.0 graphics.

In short, the Pi 4 is the most powerful of the current Raspberry Pi models. However, the different memory versions have an increased cost. The 1GB version costs £34, 4GB is £54, and the 8GB version will set you back £74. Remember to also factor in one or two micro-HDMI cables with your order.

## ZEROS:

In between the first and second generation Raspberry Pi models, the foundation launched the Pi Zero and Zero W.

The Raspberry Pi Zero was a significant release, as the extremely popular Pi was now even smaller. Measuring at just 65 x 30 x 5mm, the Zero still managed to pack in a single core 1GHz processor, 512MB of memory, a mini-HDMI port, micro USB port, 40-pin GPIO and a micro-SD card slot. However, it lacked wireless and other networking capabilities, so you would need to factor in a USB hub and network hardware.

The Raspberry Pi Zero W, on the other hand, is a far better choice. The processor and memory are the same, as are the other hardware items, but, as the W indicates, this model comes with wireless networking built-in. With a 2.4GHz single-band Wi-Fi module as well as Bluetooth 4.1, the Pi Zero W is an impressive slim bit of hardware.

While the Pi Zeros may sound like a logical choice, considering their far smaller footprint, they do lack the performance power of the newer third generation models. We'd recommend you opt for a Pi Zero W over the older Pi Zero, as networking is available out of the box.

In conclusion, the Pi 4 Model B is the main Raspberry Pi worth considering if you want the full Pi experience; use it for programming, gaming, projects, connectivity and so on. The Pi 3 Model A+ can be used for projects that require more power, but where a smaller size is needed. And the Pi Zero W for projects where a much smaller footprint and lower power draw are needed, and CPU performance isn't too important.

The Pi 3 Model A+ was released in November 2018 and greatly improves over the previous Model A.

The Raspberry Pi 3 Model B+ is one of the best Pi models available.

The Raspberry Pi Zero W, with built-in Wi-Fi and Bluetooth, is a great project Pi.

Of the Pi models available, we'd recommend the Pi 4 Model B, Pi 3 Model B+ and Pi Zero W.



# Raspberry Pi in Numbers

The Raspberry Pi was one of the most successful launches of a computer in decades. With the perfect mix of hardware, cost, connectivity, and development, plus a good, stable Linux OS to back it up, the Pi has proved itself to be a near-perfect educational platform. Here's some facts and figures for our favourite flavour of Pi.

## 3.14159265358979...



There's a Raspberry in Antarctica, where it's as cold as  $-42\text{C}$  ( $-45\text{F}$ )



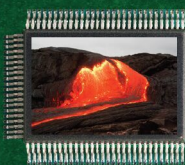
It's estimated that over **250,000** young people every week are learning how to code with a Pi

There are over **2,500** Raspberry Pi Certified Educators



The Pi 4 is between three to four times more powerful than the Pi 3 Model B+

It is estimated that there are over 50,000 code clubs across the world using the Raspberry Pi



During the eruption of the Kilauea volcano in Hawaii in 2018, a Raspberry Pi was used as a seismometer

It was originally going to be called Raspberry Py, as it was only designed to run Python

Sources: The Raspberry Pi Foundation, The Guardian, readwrite.com, Popular Mechanics, CNN, BBC.



It was originally designed for school children to learn electronics and coding

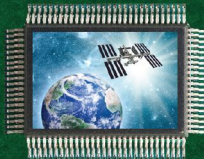


Pi Zeros are used as small cameras, placed on the backs of Green Sea Turtles, to monitor habitation

February 2019 marked the point where the **25 millionth** Raspberry Pi was sold



There are several Raspberry Pi's in Borneo's Rainforests, monitoring biodiversity



In 2017 two Pi's went onboard the ISS and ran code developed by school children

The Pi 4 is powerful enough to run Windows 10 as a Thin Client



The two most used Pi projects are retro emulation and as a media centre

It is estimated that there are over 50,000 code clubs across the world using the Raspberry Pi.



Stacked end to end, all the Raspberry Pi's would be taller than the Empire State Building



# Kit You'll Need and How to Set it Up

The Raspberry Pi is as bare bones as a computer can get. You get a motherboard, but not much else. So you'll need a few extras to get it up and running but they're things you're likely to have or at least find it easy to get your hands on.

## ASK AROUND

The kit list required to set up a Raspberry Pi is pretty basic: keyboard, mouse, HDMI monitor, SD Card and an optional case. Many of these items you'll already have, but don't rush out and buy those you don't. Ask around to see if anybody you know has spares.

### STEP 1

The most important thing you need is an SD Card. These are the storage cards commonly found in digital cameras. All modern Raspberry Pi boards, including the Raspberry Pi 3 and Pi Zero, use Micro SD cards but the older boards may use a larger SD Card.



### STEP 2

The SD Card plugs into the SD Card socket. On the Raspberry Pi 4, 3, and Pi Zero, you push the card in and pull it out. On some older models, you push the SD Card in again to eject it. All SD Cards have a notch on one corner that ensures it only goes in one way round. Unless you bought a card with NOOBS preinstalled, you'll need to install the software on it first.



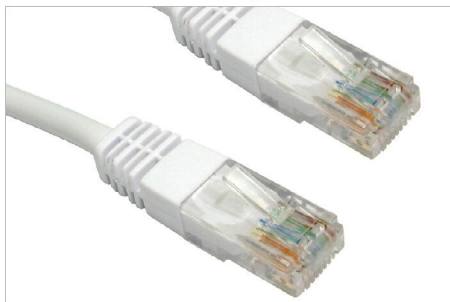
### STEP 3

You'll need a HDMI cable to connect your Raspberry Pi to a monitor or television set; these cables are used with modern televisions and video game consoles, so you shouldn't find it too hard to pick up a spare. If you own a Pi Zero, or you've got a Pi 4, you'll need a micro-HDMI cable in order to connect to a monitor or TV.



### STEP 4

You'll want to connect your Raspberry Pi to the Internet. The Raspberry Pi 4 and Pi 3 Model B+, have built in wireless networking and Ethernet. The Model A+ and Raspberry Pi 2 include an Ethernet socket, but the Pi Zero and older boards will require either an Ethernet adaptor or a Wi-Fi dongle to get online.





**STEP 5**

Both the Pi 4 and 3 come with four USB sockets but if you own a Pi Zero, you may need to purchase a USB Hub. These connect into the USB socket, and provide four (or more) USB connections. Typically, you need to connect a keyboard and a mouse, so at least two USB ports are required.

**STEP 6**

Two things that should be easy to find are a USB Keyboard and USB Mouse. We routinely prefer USB devices that plug directly into the Raspberry Pi, but devices with wireless dongles generally work just as well. Only the Raspberry Pi 4 and 3 Model B+ come with Bluetooth as standard, so you're best bet is to opt for USB, if possible.



You can also choose to house your Raspberry Pi in one of many enclosures like the official one pictured here.

**GETTING EVERYTHING CONNECTED**

Now that you've got all the basics you'll need to get everything set up. If you haven't installed the NOOBS files on your SD Card, then it's best to do that first (see the next few pages).

**STEP 1**

Connect your Raspberry Pi to the monitor using the HDMI cable. Now attach the keyboard and mouse to separate USB ports. If your Raspberry Pi only has one USB port, connect the mouse and keyboard to the USB hub and attach it.

**STEP 2**

Now, if required, connect the Ethernet cable to the Raspberry Pi. Connect the other end of the cable directly into your modem/router or into a network socket.

**STEP 3**

Finally connect a Micro USB cable to the power supply and attach the other end to a 5V USB power adapter. This is the type used to power most modern smartphones, so you should be able to source a spare. Press the On button to power up your Raspberry Pi.

**STEP 4**

Consider a case to hold your Raspberry Pi together. Raspberry now makes an official case that you can use to hold all the components and protect it from knocks and drops.



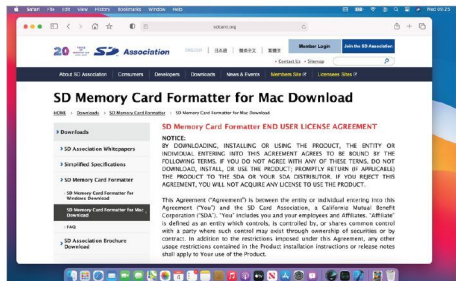
# Set Up Raspberry Pi Using a Mac

You should install NOOBS (New Out Of Box Software) on your SD Card before you go any further. This feature looks at how to format an SD Card and copy the software using an Apple Mac computer.

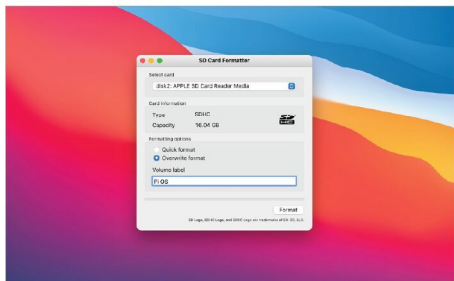
## GETTING TO KNOW NOOBS

The easiest way to get up and running is to use NOOBS, a software program created by the Raspberry Pi Foundation. You can buy a NOOBS SD Card from Raspberry Pi but it's easy to make your own with an old unused SD Card (8GB recommended).

**STEP 1** We're going to use a utility called SD Card Formatter 5 to erase the SD Card with correct formatting. You can download this app from [https://www.sdcard.org/downloads/formatter\\_4/](https://www.sdcard.org/downloads/formatter_4/). Click Download SD Formatter for Mac and Accept. Click the SDFormatter.pkg file in your Downloads folder and follow the instructions to install the app.

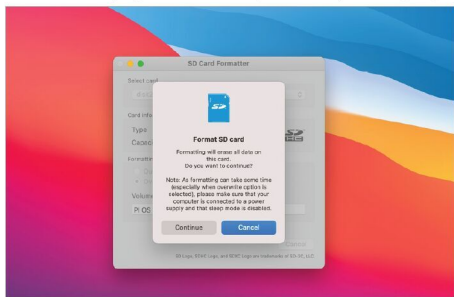


**STEP 3** Make sure the card is present in the Select Card area; you should only have one SD Card inserted into your Mac. Open SDFormatter and choose the Overwrite Format option to ensure that all the old data is removed from the card. Enter a name for the card in the Name field to help you identify it; although this isn't necessary for the installation process.



**STEP 2** Use an SD Card of at least 8GB capacity to install the operating system on your Raspberry Pi. The card needs to be formatted, which will erase all the data on it, so make sure you have copied any files from it you want to save. Insert the SD Card into your Mac, either directly into the SD Card slot or with an SD Card adaptor.

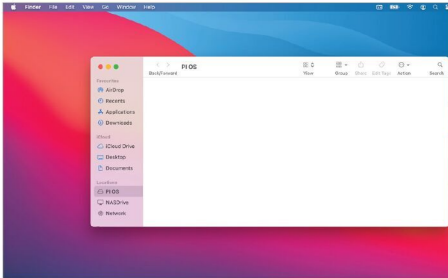
**STEP 4** Click the Format button when you are ready to wipe the card. There are other ways to format SD Cards in macOS (in particular Disk Utility) but the advantage of SD Card formatter is that it wipes SD Cards with the correct FAT32 format and doesn't affect the protected partition. It can take a while to format the SD Card, but the progress bar will show you how the process is going.





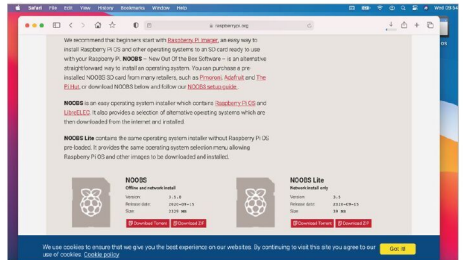
## STEP 5

When SD Card formatter has finished formatting the SD Card it will be mounted so you can access it. By default mounted volumes appear as an icon on the desktop. If not open a new Finder window and check for it under Devices. Check that the SD Card is accessible and click Close in SDFormatter.



## STEP 6

Now it's time to download the NOOBS software from the Raspberry Pi website. Open Safari and enter [www.raspberrypi.org/downloads/](http://www.raspberrypi.org/downloads/) into the Smart Search Field. Scroll down to find the NOOBS section (not NOOBS LITE) and click Download Zip. A zip file containing the NOOBS files will be placed in your Downloads folder.

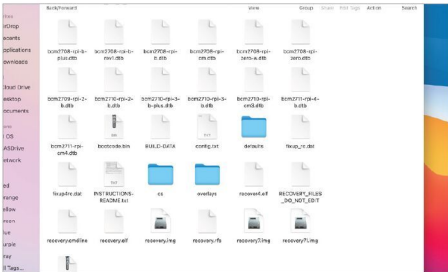


## COPY THE SOFTWARE

Transfer the NOOBS files to your SD Card to continue the setup process.

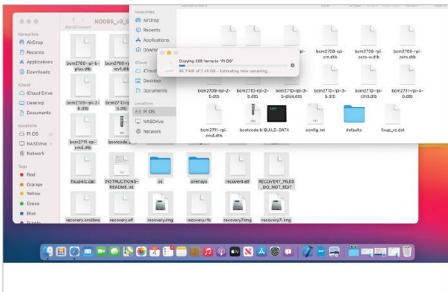
### STEP 1

Open the Downloads folder and click on the NOOBS zip file to unzip it. A NOOBS folder should appear in your downloads; it will be marked with the version number you downloaded from the Raspberry Pi Downloads page. Click on the folder to open it and view all the files contained inside.



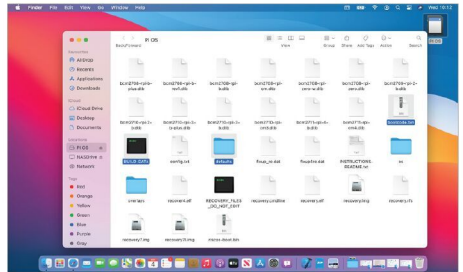
### STEP 2

Press Command-A to select all the files inside the NOOBS folder. Now drag and drop all the files from the NOOBS folder to the SD Card. This will copy all of these files to the root (the base) of the SD Card. Make sure you copy the files and not the NOOBS folder containing them.



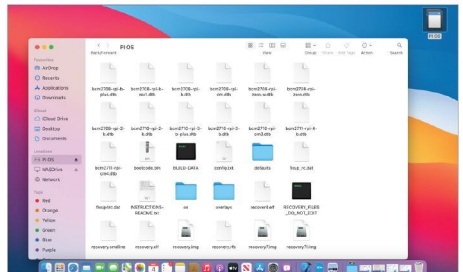
### STEP 3

Wait for all of the files to be copied from your Downloads folder to the SD Card. After the files have finished copying open the SD Card and check that all of the files are in the root. You should see "bootcode.bin" and "BUILD-DATA" files, and a "defaults" folder among other files.



### STEP 4

Make sure you eject the SD Card properly. Do not just remove it from the Mac. Instead, drag the SD Card icon to the Trash to eject it from your system. Or open a new Finder window and locate the SD Card under Devices, click the small Eject icon next to it. Now remove the SD Card from your Mac. It is ready to be inserted into your Raspberry Pi.





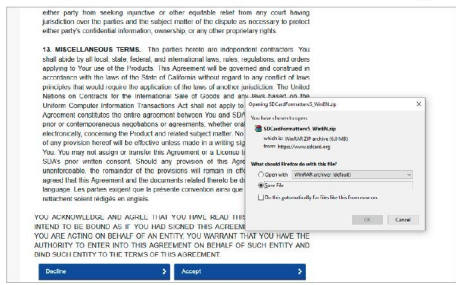
# Set Up Raspberry Pi Using a Windows PC

It's easy to set up your Raspberry Pi with a Windows PC by downloading and copying NOOBS. This installs a selection of OS's onto the Raspberry Pi. So let's format your SD Card and install NOOBS using a Windows PC.

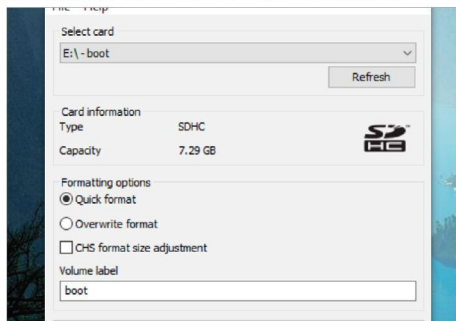
## SETTING UP WITH NOOBS

NOOBS (New Out Of Box Software) is a program used to make setting up a Raspberry Pi simple. You can buy SD Cards with NOOBS preinstalled, but here we'll look at how to format your SD Card and install NOOBS using a Windows computer.

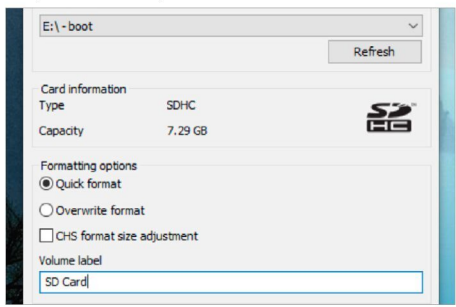
**STEP 1** We're going to use a utility called SD Card Formatter to erase the SD Card with correct formatting. You can download this app from [https://www.sdcard.org/downloads/formatter/eula\\_windows/index.html](https://www.sdcard.org/downloads/formatter/eula_windows/index.html). Click the Accept button to start the download of the latest version of the software. Extract the software, and double-click the executable to install and run the app.



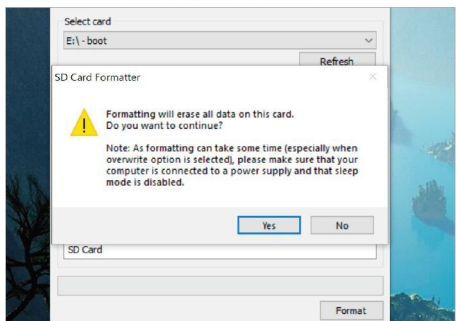
**STEP 2** We're going to use an 8GB SD Card to install the operating system on our Raspberry Pi. The card needs to be formatted, which will erase all the data on it, so make sure you have copied any files from it you want to save. Insert the SD Card into your PC, either directly into the SD Card slot or using an SD Card adaptor.



**STEP 3** The Drive letter will be automatically assigned depending on the drive(s) in your PC. Click Refresh if it can't find your SD Card. Ensure the Quick Format option is selected. Enter a name in the Volume Label field to make it easier to identify the card (we used "SD Card").



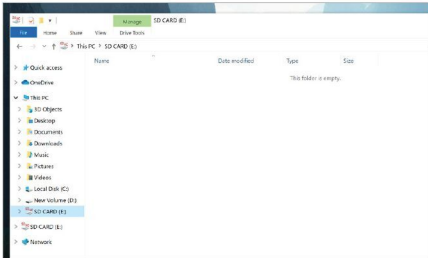
**STEP 4** Click the Format button and Yes when you are ready to wipe the card. There are other ways to format SD Cards in Windows, but the advantage of SD Card formatter is that it wipes SD Cards with the correct FAT32 format, and doesn't affect the protected partition. Click OK again, when the format is complete.





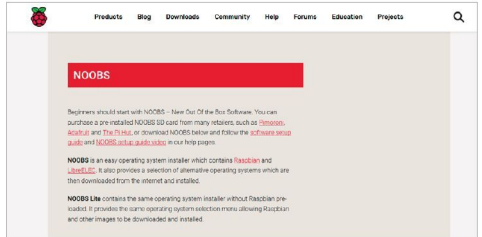
## STEP 5

Click Exit to close down the SDFormatter program. Click Start > File Explorer and choose the SD Card in the sidebar. The drive should be empty, but we will copy the files to it in the next steps. For now just make sure that you can access the empty root of the SD Card.



## STEP 6

Now it's time to download the NOOBS software from the Raspberry Pi website. Open a web browser and enter <https://www.raspberrypi.org/downloads/noobs/> into the URL field. Scroll down to find the NOOBS section (not NOOBS LITE) and click Download Zip and Save. A zip file containing the NOOBS files is copied to your Downloads folder. Click Open Folder when it is done.

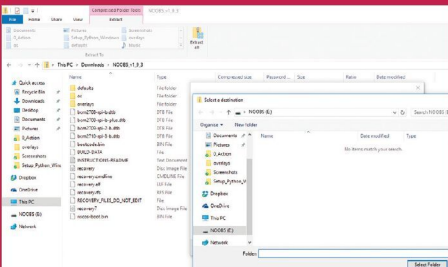


## COPY THE SOFTWARE

Continue making your SD card by copying the files across.

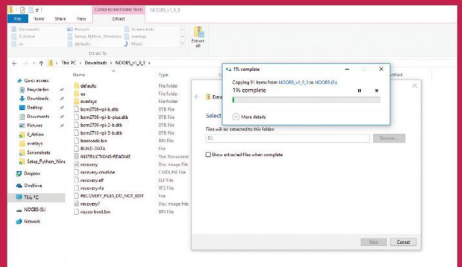
## STEP 1

A NOOBS folder should appear in your downloads. It will be marked with the version number that you downloaded from the Raspberry Pi Downloads web site. Double-click the NOOBS zip file in your Downloads folder to view the contents. These are the files you need to copy to the root (base) of your SD Card.



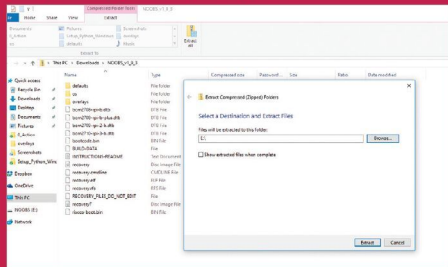
## STEP 3

Wait for all of the files to be copied from your Downloads folder to the SD Card. The files that are copied will install an operating system of your choice on the Raspberry Pi when you first boot it up. It is important that the files are copied to the root (base) of the SD Card and are not inside another folder such as the NOOBS folder.



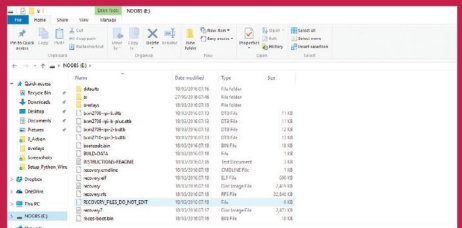
## STEP 2

Click Extract All to expand the zipped file and extract its contents. Click Browse in the Select a Destination and Extract Files folder and choose "SD Card(F:)" in the Select a Destination window. Click Select Folder and Extract. The files will now be copied directly from the zip file to your SD Card.



## STEP 4

Click Start > File Explorer and choose "SD CARD (F:)" in the sidebar to display the contents. It should now contain all the files NOOBS needs to setup an operating system on your Raspberry Pi. Check that you can see a Defaults folder, bootcode and BUILD-DATA files along with the other files shown here. Remove the SD Card from your Windows PC. It's now ready to be inserted into the Raspberry Pi.





# The Raspberry Pi Desktop: What You Will Need

Did you know that there is a way in which you can run the full Raspbian OS desktop without even needing a Raspberry Pi? The Raspberry Pi Desktop edition is a fully working, Debian-based installation of Raspbian that can run on a standard PC.

In 2016, the Raspberry Pi Foundation started work on an x86 version of its popular Raspbian operating system. By the end of the following year, there was a link on the Downloads page and non-Pi owners could now install and use Raspbian in almost the exact same way as if they were in front of a Raspberry Pi.

Since then, the Raspberry Pi Desktop, or as it's correctly called: Debian Stretch with Raspberry Pi Desktop, has followed the same release path as its Pi-based sibling. However, there are some differences you need to be aware of prior to throwing your Pi away and opting for this version instead.



The Raspberry Pi Desktop (x86 version) allows you to view the same desktop and pre-installed software as the Full Desktop version available for the Raspberry Pi. You can install it on any decent PC, or laptop, one that's at least six or seven years old would be perfect and as a PC that age usually struggles with the latest version of Windows 10, you can now install Raspbian instead of boxing that old kit and resigning it to a life of gathering dust in the loft.



The major difference is that this version has been compiled to run on x86 processors, that's Intel and AMD PC processors. Whereas, the version of Raspbian that you will find on a Raspberry Pi has been compiled and created exclusively for ARM processors; the type of CPU a Raspberry Pi uses. These CPUs use a different architecture to that of an Intel or AMD CPU, so the software required to run on one won't run on the other unless the developer has specifically made either an x86 or ARM version. In short, if you've found a piece of software that you enjoy using on your Raspberry Pi, you won't be able to use it on the Raspberry Pi Desktop x86 version – UNLESS, there's a dedicated x86 version of the software.



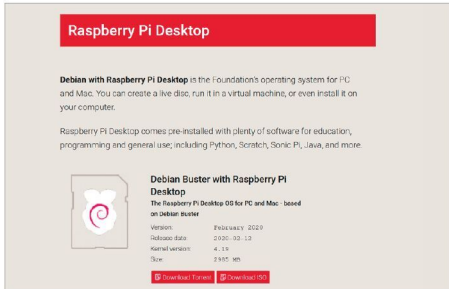
Another caveat worth mentioning is that you won't have access to the 40-pin GPIO that the Raspberry Pi features, as this isn't a standard piece of hardware on a normal PC. There is, however, a way around this. If you own a Raspberry Pi Zero (either with or without the W), then you're able to connect the Zero to a spare USB port on the PC, via the Zero's micro USB port and the Raspberry Pi Desktop OS will recognise the Zero as extended hardware and allow you access to the Zero's 40-pin GPIO, which, you have to admit, is pretty clever stuff.



## WHAT YOU WILL NEED

Here's a list of items needed if you want to test out the latest version of the Debian Stretch with Raspberry Pi Desktop OS on your PC.

**THE OS** Naturally you'll need the Raspberry Pi Desktop OS, just as you would with Raspbian or the Pi. Open a browser and download the ISO file that contains the OS from: <https://www.raspberrypi.org/downloads/raspberry-pi-desktop/>.



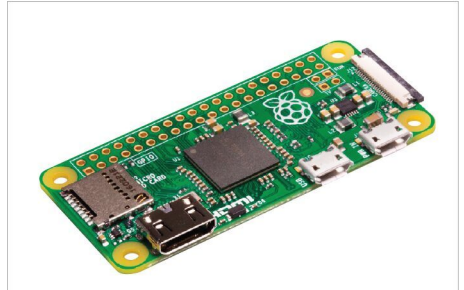
**USB STICK** The best method of transferring the ISO to a PC for installation is to use a combination of a tool such as Rufus (found at: [https://rufus.ie/en\\_IE.html](https://rufus.ie/en_IE.html)), which is a piece of software that can create a bootable USB driver from an ISO file and, at least, an 8GB USB stick.



**A COMPUTER** An older desktop PC or laptop is an ideal candidate for running the Raspberry Pi Desktop OS. While you can easily use the latest PC/laptop, it's a bit overkill for this particular operating system.

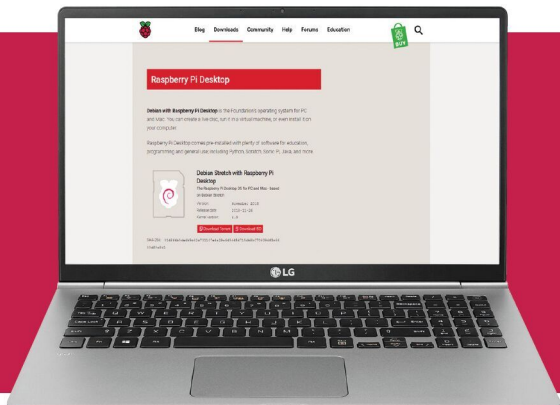


**RASPBERRY PI ZERO** Although not strictly a necessity, a spare Raspberry Pi Zero will allow you to access the 40-pin GPIO from within the Raspberry Pi Desktop OS. If, however, you don't have a spare Zero, it's not a problem as there's still plenty you can do with Raspbian on a PC.



## HOW TO SET UP THE RASPBERRY PI DESKTOP

Setting up the Raspberry Pi Desktop version on a PC or laptop is very simple, and follows roughly the same method as the previous Mac and Windows setups for the Pi. Begin by downloading the ISO image from <https://www.raspberrypi.org/downloads/raspberry-pi-desktop/>, then download Rufus from [https://rufus.ie/en\\_IE.html](https://rufus.ie/en_IE.html). Insert an 8GB USB stick and run Rufus, then follow the on-screen instructions to locate the downloaded Desktop ISO. Once the image has transferred to the USB stick, remove it from the computer and insert it in the spare laptop/PC. Select **Boot from USB** from your system's BIOS/UEFI then follow the instructions to use and install the Raspberry Pi Desktop OS.





# Debian Buster with Raspberry Pi Desktop

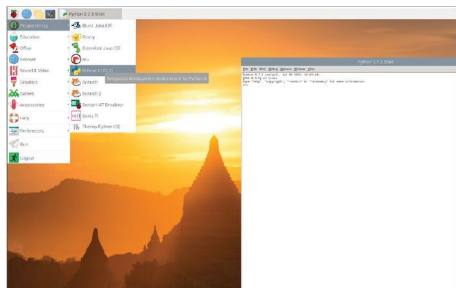
You can be forgiven for thinking that a version of the Pi OS that isn't on a Raspberry Pi is somehow lacking, and therefore a bit pointless in its use. However, there's a lot that you can do with the Debian Buster with Raspberry Pi Desktop version.

## 10 THINGS TO DO WITH X86 RASPBERRY PI OS

There's just as much you can do with this version of Raspberry Pi OS as with the dedicated Pi version. Here's ten great projects, and things to do with Debian Buster with Raspberry Pi Desktop.

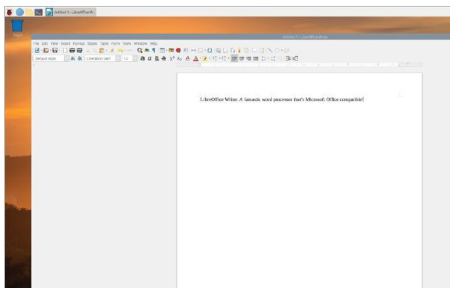
### CODING

This version of Raspbian apes the Full Desktop version for the Raspberry Pi, so that means it comes with all the necessary programming languages out of the box. Learn Python, C++, use the HAT emulator, and even learn JAVA. It's all there under the Programming menu option.



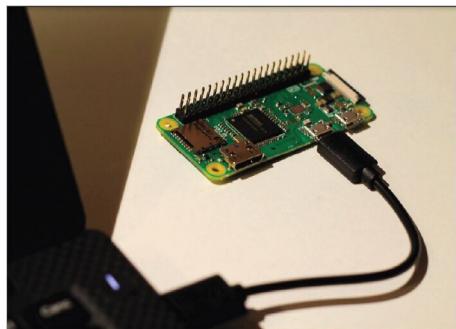
### FULL DESKTOP

As the x86 version of Raspbian follows the Full Desktop Pi version, there's an entire office suite pre-installed. That means you can use that old laptop or PC as a fully functional desktop replacement for Windows or macOS. LibreOffice is also Microsoft Office compatible.



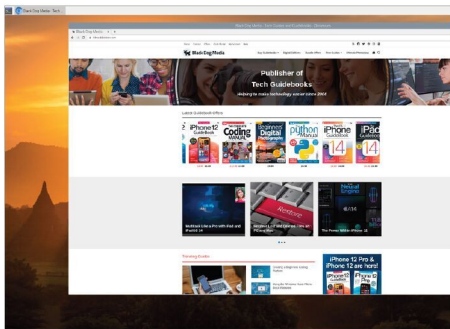
### HARDWARE

Hook up a Pi Zero without an SD installed, and select GPIO Expansion Board from the options, and you will be able to talk to the Zero's 40-pin GPIO via the likes of Python or Scratch. By doing so, you'll have all the benefits of a Raspberry Pi, with the power of your laptop.



### WEB BROWSING

As with most modern operating systems, you can also browse the Internet exactly the same as you would with the likes of Windows 10 or the latest macOS. As this is a Linux-based OS, there's also an added element of security, as Windows-targeted viruses won't affect it.

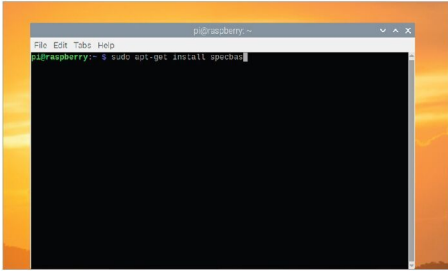






## LINUX X86-BASED SOFTWARE

As this is a Debian based version of Linux, you can naturally install any of the tens of millions of Linux x86-based software currently available. Remember though, only software that has both an ARM and an x86 version will work on both the Pi version and this version of Raspberry Pi OS.



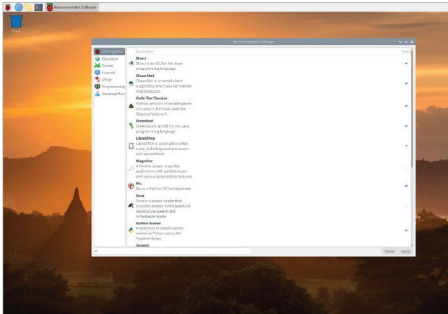
## MINECRAFT

The Raspberry Pi, even the Pi 3 Model B+, isn't the most powerful computing device available. Your older laptop is probably significantly more powerful, and as such, you will be able to play the likes of Minecraft without too much trouble.



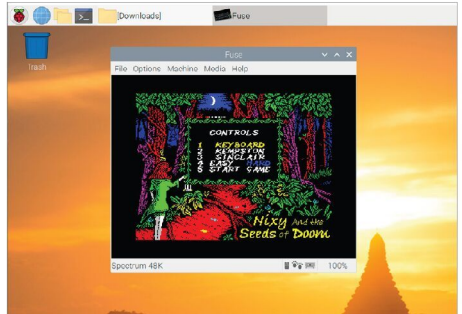
## RECOMMENDED SOFTWARE

The Recommended Software option is also available with the version of the PI OS. This tool allows you to pick and choose which of the Pi-favoured software you can add or remove. Simply tick the box next to the name of the app.



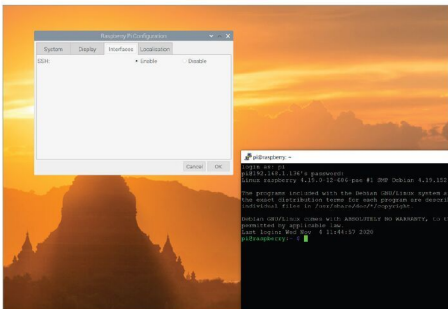
## RETRO GAMING

If retro gaming is more to your tastes, then Debian has a huge number of retro emulators available covering consoles and home computers. You will need legal ROMs, however, to play the games, but these are widely available from reputable sites.



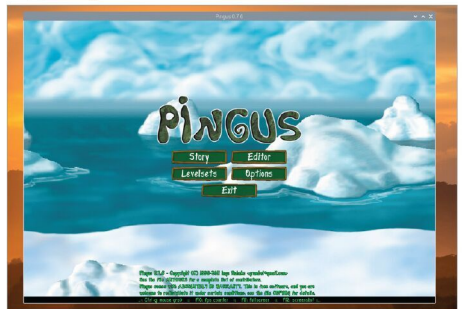
## SSH

It's easy to use SSH from within the Raspberry Pi Configuration tool. Select the Enabled option, then use a client such PuTTY to SSH into the OS. Here we've enabled SSH and have connected from another Windows PC, using PuTTY as the SSH client.



## GAMING

Of course, there's also an equally large selection of modern gaming available for Debian; most of which will run perfectly well under the PI OS in an x86 environment. You will need to Google what's available, and try out a few examples, the list is simply too big to mention.







# Explore Raspbian

The Raspbian operating system is a Linux distribution that comes pre-packed with a fantastic collection of programs and apps to help you get the most from your Pi. Raspbian enables the Raspberry Pi to become a fully functioning desktop computer, and with it, you can code in the latest version of Python, C++ and more. There's even an entire office suite for free, called LibreOffice, that's Microsoft Office compatible if you install the Full Desktop version of Raspbian.

Raspbian allows you to connect to your existing home network, surf the Internet, play music and videos, edit documents, send emails and much more. With a few simple tricks you can easily install more free software that, for instance enables you to edit images and videos or even connect to other computers and control the desktop.

This section of the book introduces many of the skills, tricks and hacks you need to take full command of the Raspberry Pi. Skills that you will use time and time again when creating your own Pi projects.



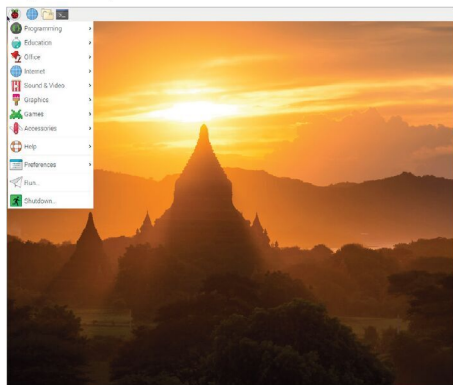
# Take a Tour of PIXEL

PIXEL is the desktop environment for your Raspberry Pi. It's built on top of Raspbian, the core OS, which is itself designed around the Linux distribution - Debian, specifically, Debian Buster. PIXEL is fine-tuned to run on the Raspberry Pi, taking advantage of its unique hardware.

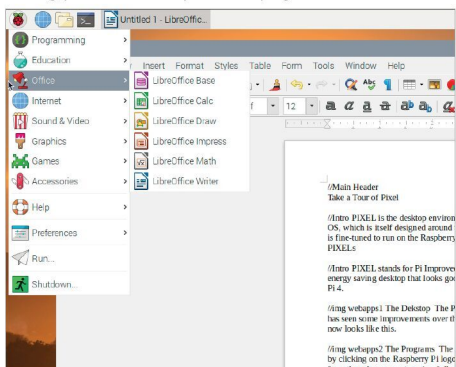
## PIXEL

PIXEL stands for "Pi Improved Xwindow Environment, Lightweight". It's a lightning fast and energy saving desktop that looks good too. It contains almost everything you'll ever need to work with the Pi 4.

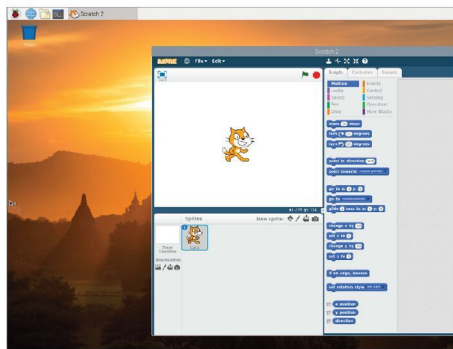
**THE DESKTOP** The PIXEL interface is a module that's installed on top of the core OS. It has seen some improvements over the years, and with the release of the Pi 4 and Raspbian Buster, it now looks like this.



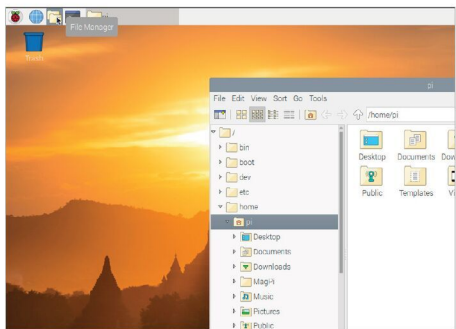
**PRODUCTIVITY** You'll notice that within the Main menu, there's a category named Office. In here, you'll find a pre-installed suite of programs called LibreOffice, that are Microsoft Office compatible; there's a word processor, database, drawing, presentation and spreadsheet program.



**THE PROGRAMS** The programs that come pre-installed with Raspbian are easily located by clicking on the Raspberry Pi logo in the upper left corner of the desktop. Once there, you can select from the sub-menu categories, followed by the program name.



**FILE MANAGER** To view the files stored on your Raspbian OS, click on the File Manager icon in the Application Launch Bar (the one that is shaped like a pair of files). This opens a window displaying all the files in your home directory. The Directory Tree gives quick access to common folders, such as Desktop and Documents.





## MENU EDITOR

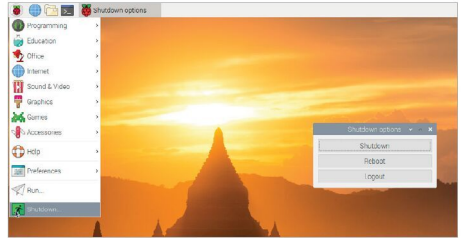
It is possible to remove and add items to the Application Menu using the Main Menu Editor.

Choose **Menu > Preferences > Main Menu Editor**, then add or remove ticks next to programs and sections you want to include/exclude. Use **Move Up** and **Move Down** to rearrange items in the Menu.



## SHUTDOWN

It's important that you switch off your Raspberry Pi safely. When you are finished with your Raspberry Pi session click **Menu > Shutdown**. Three options appear: **Shutdown**, **Reboot**, and **Logout**. Click **Shutdown**. Always wait for the screen to go blank before removing the USB power.

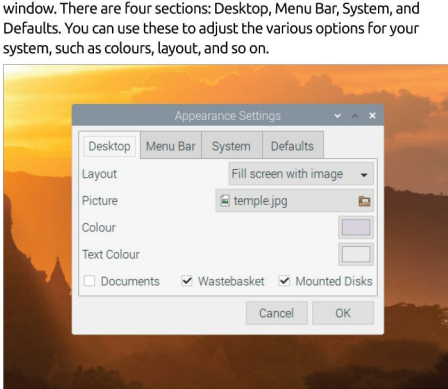


## CUSTOMISE THE DESKTOP

As with any other operating system desktop, you're able to customise the Raspberry Pi's PIXEL interface, making it more personal and more you.

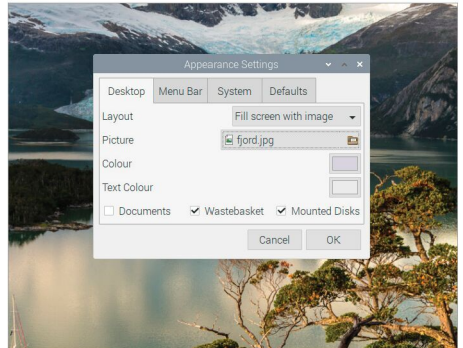
### YOUR PIXEL

Choose **Menu > Preferences > Appearance Settings** to open the Appearance Settings window. There are four sections: Desktop, Menu Bar, System, and Defaults. You can use these to adjust the various options for your system, such as colours, layout, and so on.



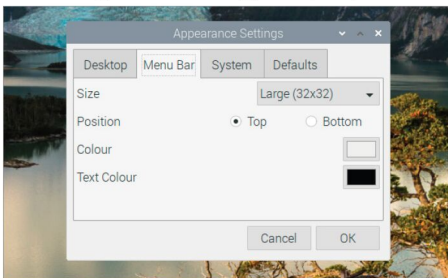
### MENU BAR

The second tab in the Appearance Settings window allows you to adjust the menu bar that runs along the top of the desktop. You can change its size, colour, and even its location from the top to the bottom of the screen.



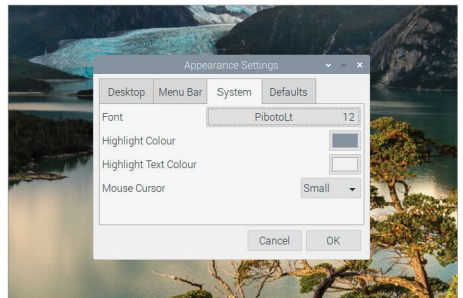
### DESKTOP WALLPAPER

The most common change is that of the desktop wallpaper, or desktop background. The default is called 'Temple', however, by clicking on **temple.jpg** in the **Picture** section of the **Appearance Settings**, you have other images available. Naturally, you can opt for your own.



### FONTS

The **System** tab in the Appearance Settings window will enable you to change the core system font, as well as the mouse cursor size. There are numerous fonts to choose from and, if you want, you can even install your own.





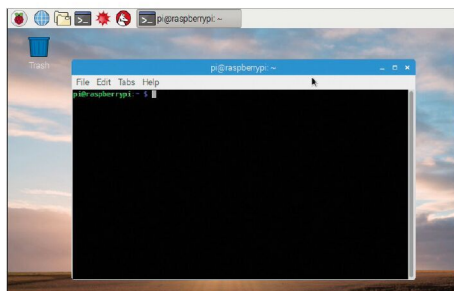
# Exploring the Command Line

If you've grown up with Windows or the Mac OS then you might never have encountered the command line. This part of the operating system sits beneath the desktop and is used to control a computer using text commands. You'll need to get familiar with it to use your Raspberry Pi.

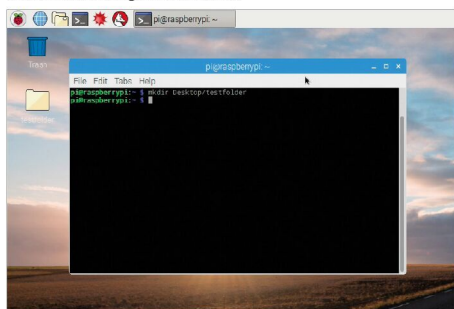
## USING TERMINAL

Despite its graphical PIXEL interface, Raspbian is a Linux operating system and this means you'll spend a lot of time working with text commands. Using the command line is an important part of learning to use a Raspberry Pi computer.

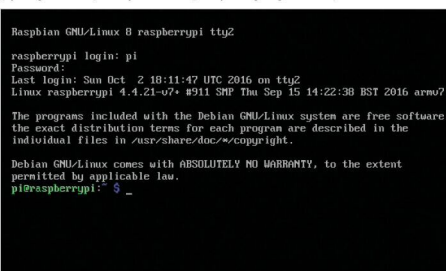
**STEP 1** By default the Raspberry Pi boots directly into the PIXEL interface. Here you use a visual metaphor, files, buttons and so on, to make changes to your computer. The command line is where you control a computer by entering text commands. Click Terminal in the Menu bar to view the console.



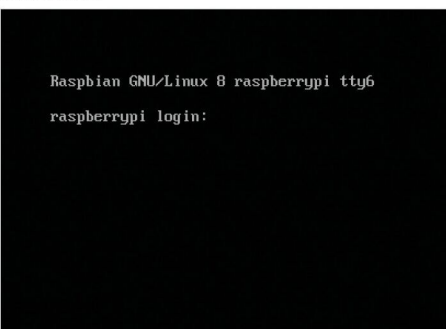
**STEP 2** You can enter commands into the Terminal to make changes to the computer. This works alongside the PIXEL interface. Enter: `mkdir Desktop/testfolder` and press return. Notice a new folder appears below the Trash on the desktop. Drag it to trash to get rid of it. You can use the Terminal and PIXEL environments alongside each other.



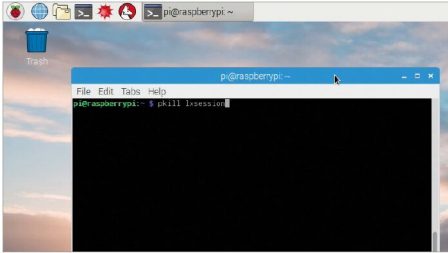
**STEP 3** Another way to switch to a command line environment is to press Control+Alt and F2. This gets rid of the desktop completely and you see just a black screen with text. This is known as a virtual environment and you're now in tty2 ("tty" is a throwback to teletext writers). You'll need to enter your login name ("pi" by default) and password ("raspberrypi" by default).



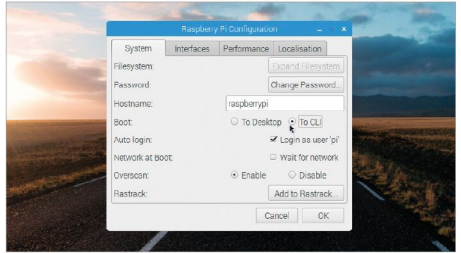
**STEP 4** There are six different virtual environments available. Press Control+Alt-F3 to switch to the third, and Control+Alt-F4 to go to the fourth, and so on. You'll need to log on to each one in the first instance but can then jump back and forth between them.



**STEP 5** Press Control-Alt-F1 to return to the PIXEL interface. One thing to realise is that PIXEL is just a program running on top of the tty1 session. Open Terminal and enter: `pkill lxsession`. Ouch, there goes your desktop. Don't worry, enter: `startx` to get it back again. Most of the time you just enter commands into the Terminal window on top of the PIXEL environment.



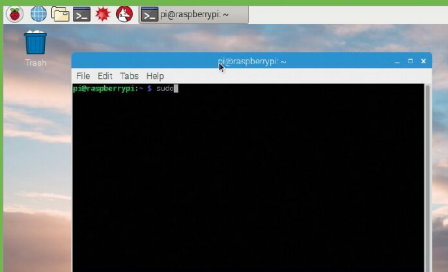
**STEP 6** Some users prefer to use just the command line. Open Menu > Preferences > Raspberry Pi configuration and choose To CLI and click OK and Yes. The Raspberry Pi will now boot into the command line interface. Enter: `startx` to get back to PIXEL, open Raspberry Pi Configuration and choose To Desktop to get back to normal.



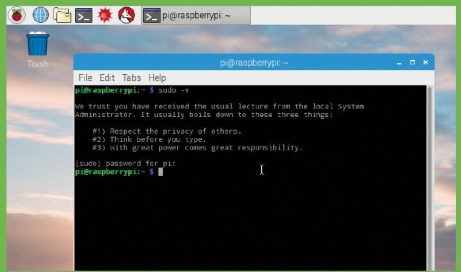
## USING SUDO

Discovering the power of the super user.

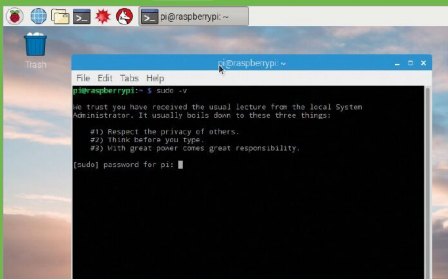
**STEP 1** One of the most important command line instructions to know is `sudo`. This command stands for "substitute user do", sometimes incorrectly called "super user do" and allows you to run a command as another user. Typically this is the root user account, which has more access privileges than your user account.



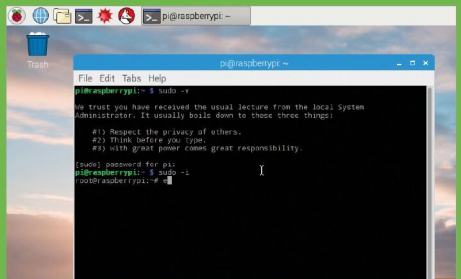
**STEP 3** The first time you enter `sudo` you'll be given a warning message. Commands to make, edit and delete files prefaced with `sudo` can be used to change any file on the system. Sometimes with powerful, or disastrous, results. Be careful. Enter your password and press return.



**STEP 2** When you start a command with `sudo` it is run as the root user and is often used when changing files outside of your user account, such as installing new programs. When you enter `sudo` you will be asked to enter your user password. Enter: `sudo -v` and press enter.



**STEP 4** Be especially careful of any set of commands that use `sudo -i`. This takes you to root mode, where every command is run as `sudo`. Your name (normally "pi") is replaced with "root". Root mode is generally considered a bad place to be, so enter exit to get out of it as soon as possible.



# Setting Up a Static IP Address

Setting up a static IP address comes with several advantages, the main one being you'll always know what your Pi's network address is – should you decide to operate it without a keyboard, mouse, or monitor. It's easy to set up too, but it's not a critical process.

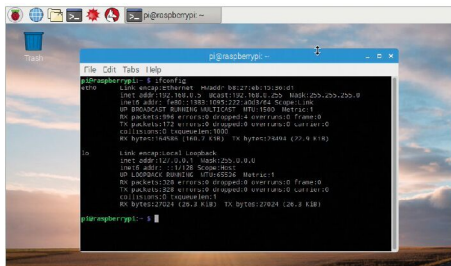
## DHCP RESERVATION

Now that you've set up your Raspberry Pi and connected it to the network, you should take time to fix its IP address. This will make it a lot easier for you to connect to the Raspberry Pi from a Mac or Windows computer and share files.

**STEP 1** When you connect your Raspberry Pi to a network, the router (or modem/router) assigns it a number, known as its IP (Internet Protocol) address. This is a block of four digits and the first three are usually 192.168.0. After that is a digit for each electronic device.

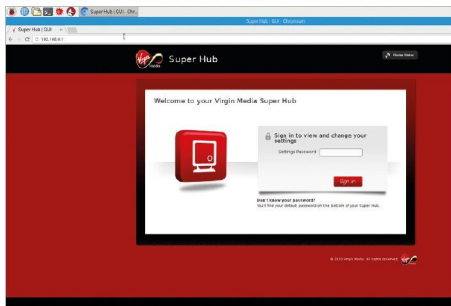


**STEP 3** To find out what number your Raspberry Pi is using click on the Terminal icon and enter: `ifconfig` and press Return. You should find it next to "inet addr". The challenge is that the router assigns this number using DHCP (Dynamic Host Configuration Protocol). When the Raspberry Pi is unplugged it reuses the number and your Raspberry Pi may get a different number next time.



**STEP 2** The router typically takes the first address, so it is usually found at 192.168.0.1. That number is reserved for the router. Often you'll find this number on your router marked "web address". The router then assigns similar IP address numbers to the other devices you own as they are added to the network, so 192.168.0.2 might be your computer, 192.168.0.3 your smartphone and so on.

**STEP 4** It's better all-round if you can get the Raspberry Pi to use the same IP address every time you connect it to the network. You do this using DHCP Reservation. This is where you tell your router to remember the Raspberry Pi address, and only use that address from now on. Start by opening the Chromium Browser and entering 192.168.0.1 to connect to your router's web interface.

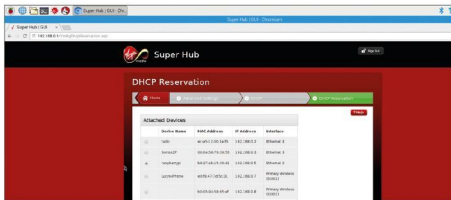






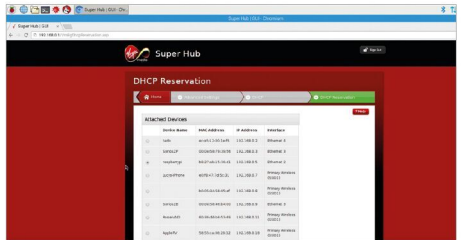
## STEP 5

We're using a Virgin Broadband router but the process is similar on most routers. Google the name of your router and "DHCP Reservation" to find the router you're looking for. Enter your admin password (our default password was listed on the back of the router). Click Advanced Settings > DHCP Reservation. In the DHCP Settings section you will find all of the devices connected to your network. Scroll down and find the one with the same IP Address listed in Step 3.



## STEP 6

Select the device using the check box on the right and scroll down to the Add Reservation section. It should have filled in the Device Name, MAC Address and IP Address fields. If the Device name is "unknown" change it to "raspberrypi". Click Add Reservation to ensure it always uses that IP Address. Click Apply to enforce the changes.

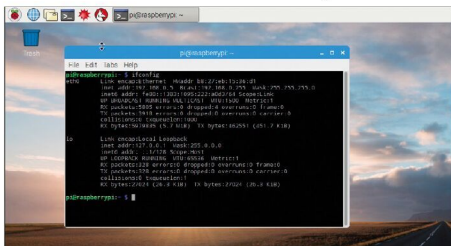


## MANAGING YOUR DHCP DEVICES

Now that you have set up a DHCP Reservation, you need to understand how to manage the different devices that may need to use it and connect to the IP.

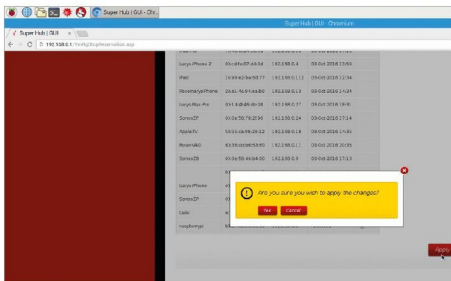
## STEP 1

Now that you have set up your Raspberry Pi with a DHCP Reservation it will always connect to the router using that IP Address. You can check this at any time by entering: `ifconfig` into terminal. The router always knows that it is your Raspberry Pi by checking its MAC address. This is the set of six hexadecimal numbers next to "HWaddr" in ifconfig.



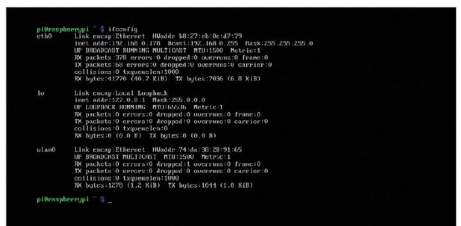
## STEP 2

If you decide to free up that IP Address so other devices can use it, you can remove it from the router. Open the router web interface (192.178.0.1) and click Advanced Settings > DHCP Reservation. Scroll down to the IP Lease Table to find the Raspberry Pi. Select the check-box next to it and click Apply and Yes.



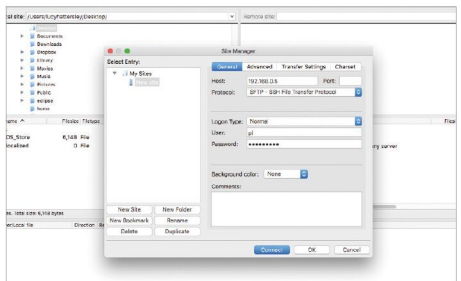
## STEP 3

Your Raspberry Pi will still have the same IP Address but it isn't guaranteed to keep it. Restart your Raspberry Pi by entering: `sudo shutdown -r` now into a Terminal window. When it has restarted enter: `ifconfig` and you'll find you still use the same IP Address. Over time the router will use other slots for new devices until it reaches its maximum (255 by default). Then it will recycle disconnected devices.



## STEP 4

Setting up DHCP Reservation is good practice because it ensures you Raspberry Pi is always going to use the same IP Address. It's also a good idea to write it down or use a label printer to create a label for the Raspberry Pi. You will need this number to connect to the Raspberry Pi via SSH, VPN or FPT (techniques we will use throughout this book).





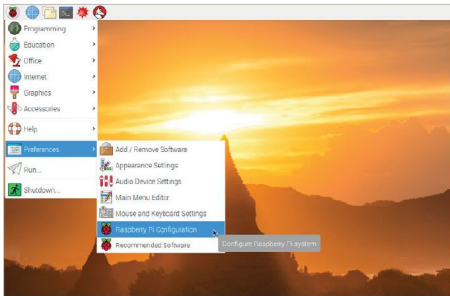
# Connecting to the Pi Remotely

A 'Headless' Pi – a Raspberry Pi that doesn't have a keyboard, mouse, or monitor connected – can easily be controlled remotely using VNC (Virtual Network Computing). Everything is built-in to the Pi, so it's an easy process to set up and use.

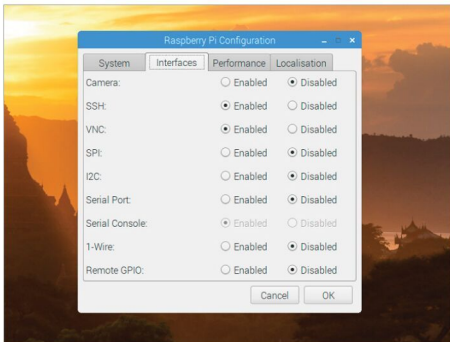
## REMOTE CONTROL

The Raspberry Pi is easy to set up and use in projects around the home and, surprisingly, many of these projects don't need a keyboard, mouse or a monitor; these are known as Headless devices. VNC lets you control a Headless Pi from another computer.

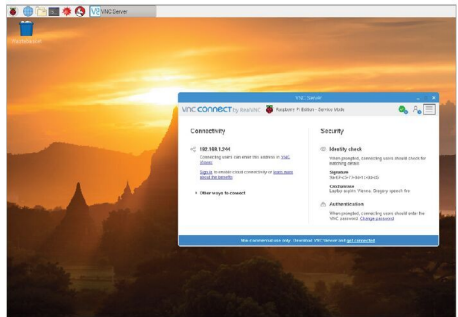
**STEP 1** You will first need to ensure that your Pi is connected to a keyboard, mouse and monitor, and is connected to your Wi-Fi router, before you attempt to connect to it remotely. Start by clicking on the Raspberry Pi menu, then scroll down to **Preferences > Raspberry Pi Configuration**.



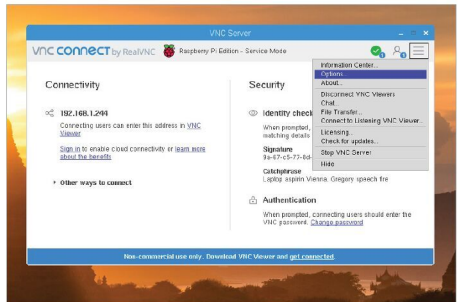
**STEP 2** With the Raspberry Pi Configuration window open, click on the Interfaces tab, and ensure that the **VNC** option is **Enabled**. Click on the radio button next to Enabled to activate, then click on the OK button to close the window.



**STEP 3** In the upper right corner of the menu bar you'll notice a VNC icon (next to the Bluetooth icon). This is the built-in VNC server, click the icon in the menu bar once to open the VNC Server window.

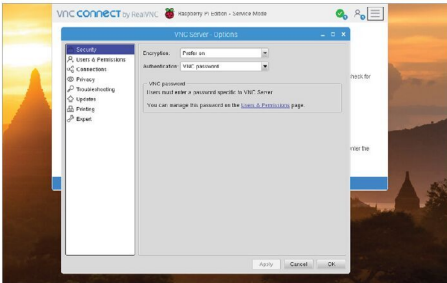


**STEP 4** Make a note of the IP Address of the Raspberry Pi as detailed in the left-hand section of the VNC Server window; our example is 192.168.1.244. Now click on the three horizontal lines in the top right of the VNC Server window. From the sub-menu that appears, click on the **Options** entry. This will open a new window pop-up.

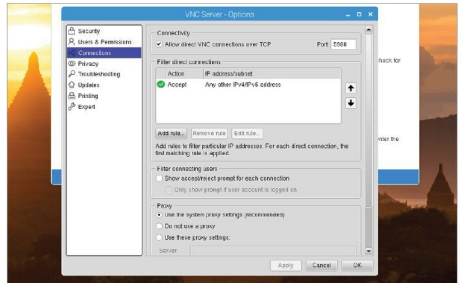




**STEP 5** With the Security option highlighted on the left, ensure that the Encryption option, to the right, is set to **Prefer On**, and that Authentication is **VNC Password**. This may ask you to enter a password, which you will use when connecting remotely from another computer. Enter a password you'll remember.



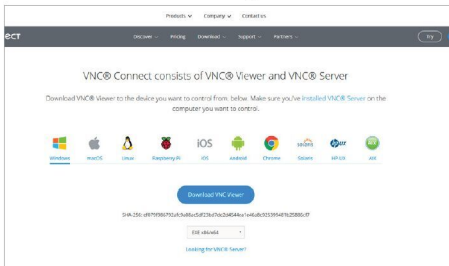
**STEP 6** While still in the Options window, click on Connections in the left-hand pane. In the right hand pane, ensure that the **Allow Direct VNC Connections Over TCP** is ticked, and that the Port is set to **5900**. It usually is, but sometimes it can be unticked if an update to software has been applied.



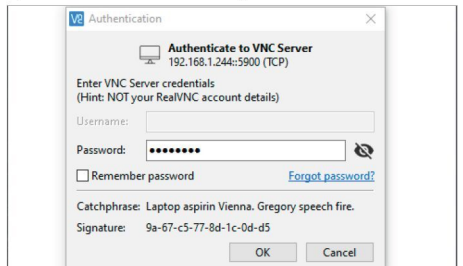
## REMOTE VIEWING

The Pi-side of things has now been set up. Just remember the Pi's IP address, and get to your PC or Mac for this next part.

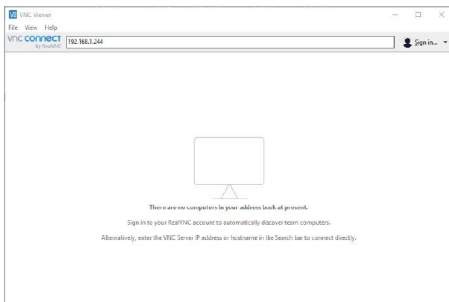
**STEP 1** Reboot the Pi, then disconnect the keyboard, mouse and monitor; but keep the power connected. From your PC or Mac, open a browser and navigate to: <https://www.realvnc.com/en/connect/download/viewer/>. Choose your operating system and click the Download VNC Viewer app.



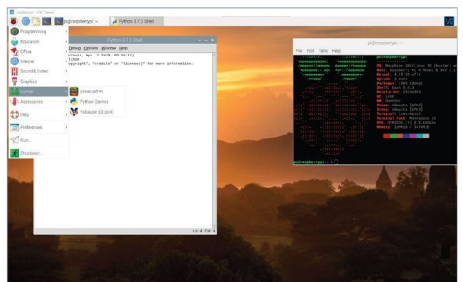
**STEP 3** In the new pop-up window, enter the password you created through the Raspberry Pi's VNC Server options page. You can choose to Remember Password if you want, but if the computer you're using is shared, you can always opt to leave the option unticked. Click OK when ready.



**STEP 2** Follow the on-screen instruction to install VNC Viewer on your system, when it's finished, launch the app and choose whether you want to send anonymous data. In the main VNC Viewer window, enter the IP Address of the Raspberry Pi; ours was 192.168.1.244. Hit Enter when you've typed in the IP address.



**STEP 4** You may be asked to confirm connection to a remote computer, click Continue to make the connection. You are now connected remotely to your Raspberry Pi. As long as the Pi is powered up, and has access to the Wi-Fi signal of the router, then you can place it anywhere and get access without using a mouse, keyboard, or monitor.



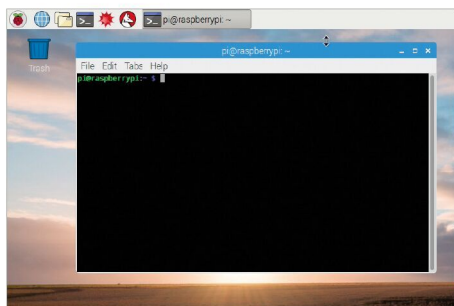
# Using APT to Install and Remove Programs

When you start to get serious with your Raspberry Pi, you'll need to install software that is available for Linux, but not part of the Pi Store. These programs are installed from the command line using a service called APT. Learning how to use APT is a vital part of using your Pi.

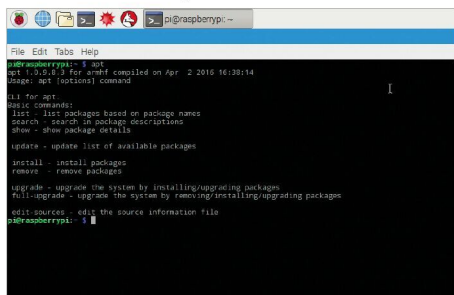
## ADVANCED PACKAGING TOOL

Add / Remove Software is great for finding and installing software but most of the time you manage apps on a Raspberry Pi using a program called APT (Advanced Packaging Tool) using the `apt` command.

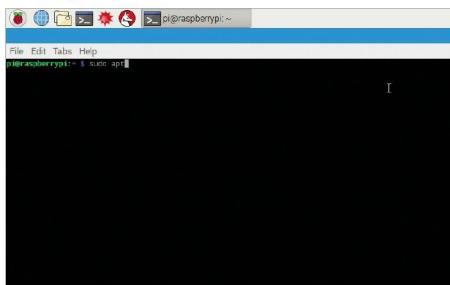
**STEP 1** To manage the apps on your Raspberry Pi you'll need to use a tool included with Raspbian called APT (Advanced Packaging Tool). APT is a command line tool so you need to either start your Raspberry Pi in the command line or click on the Terminal icon.



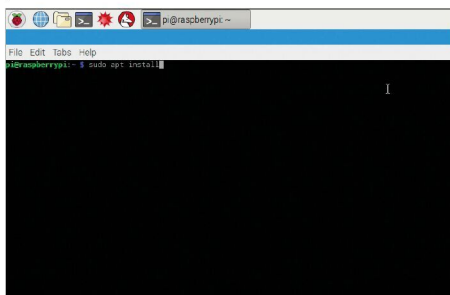
**STEP 2** The command used to control packages in Raspbian is `apt`. Type `apt` into the terminal and press Return to get a description of what the app does. The instructions tell us that the "most frequently used commands are update and install". You may also hear about "apt-get" which is an older version of the same tool. It works in much the same way.



**STEP 3** The command used to add new software to your Raspberry Pi is `apt install` followed by the name of the package you want. However, because `apt` adds (or removes) files outside of your home directory you need to preface `apt` with the word `sudo`. Almost all of the time you will need to type `sudo apt` when using this tool.



**STEP 4** Beginning a command with "sudo" runs the command as a super user, "super" meaning the one above your Pi account, rather than one with super-hero properties. However, if it helps you to think of it that way that's fine. So to install a new program you type: `sudo apt install` followed by the name of the app you want.





## STEP 5

You're going to install a chess game called **Dreamchess**. Enter: **sudo apt install dreamchess** and press Return. Descriptive text will run up the screen. You may see "Do you want to continue [Y/n]?" Enter: **y** and press Return to install the application. You'll find it by choosing: Menu > Games > Dreamchess.

```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~$ sudo apt install dreamchess
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
dreamchess-data libsm1
The following extra packages will be installed:
dreamchess-dreamchess-data libsm1
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 5,200 kB of archives.
After this operation, 6,852 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

## MORE APT COMMANDS

The **apt** command is also used to list, and delete, any apps you have installed.

## STEP 1

Sometimes you will try to install an app only to get an error that the file cannot be found. In this case you need to update apt so it has the latest links. Enter: **sudo apt update**. This doesn't update the apps you've installed, it just gets a newer listing of apps. To update your apps to the latest versions enter: **sudo apt upgrade**. You'll often find both commands rolled into one: **sudo apt update && sudo apt upgrade**.

```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~$ sudo apt update && sudo apt upgrade
Get:1 http://archive.raspberrypi.org/jessie InRelease [13.2 kB]
Get:2 http://mirrors.director.raspbian.org/jessie InRelease [14.9 kB]
Get:3 http://archive.raspberrypi.org/jessie/main armhf Packages [144 kB]
Get:4 http://mirrors.director.raspbian.org/jessie/main armhf Packages [9,981 kB]
Get:5 http://archive.raspberrypi.org/jessie/ui armhf Packages [14.1 kB]
Ign http://archive.raspberrypi.org/jessie/main Translation-en_US
Ign http://archive.raspberrypi.org/jessie/main Translation-en
Ign http://archive.raspberrypi.org/jessie/ui Translation-en_US
Ign http://archive.raspberrypi.org/jessie/ui Translation-en
104 [4 Packages 1,303 kB/6,961 kB 15%]
```

## STEP 2

To view the packages you've installed you use a different command called **dpkg**. Enter: **dpkg --get-selections | grep -v deinstall** to view all of the packages on your Raspberry Pi. Place **| less** after it to view one page at a time. Or **dpkg --get-selections | grep -v deinstall > ~/Desktop/packages.txt** to save the list as a text file on your desktop.

```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~$ dpkg --get-selections | grep -v deinstall
ac1 install
adduser install
administrative-theme install
alacarte install
alsa-base install
alsa-utils install
apt install
apt-listchanges install
apt-utils install
aptitude install
aptitude-common install
aspell install
aspell-en install
avahi-daemon install
case-files install
case-passed install
bash install
bash-completion install
bind9-host install
binutils install
bit install
```

## STEP 6

You can use **apt-cache** to search for apps you can install and remove. Enter: **apt-cache pkgnames | less** to view all the packages available. Typing **apt-cache pkgnames | less** enables you to view a page at a time (press any key to scroll). You can use **apt-cache search** to find packages, enter: **apt-cache search pong** to view a list of Pong games you can install.

```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~$ apt-cache pkgnames | less
libraspberrypi-1 apt-cache search pong
childplay Suite of educational games for young children
efp - Escape from Pong NES game
furl-2-inputpad-1.0 - On-Screen Input Pad to Send Characters with Mouse
input-pad - On-Screen Input Pad to Send Characters with Mouse
libinput-pad-dev - On-Screen Input Pad to Send Characters with Mouse - dev
libinput-pad-ptest - On-Screen Input Pad to Send Characters with Mouse - test
libinput-pad - On-Screen Input Pad to Send Characters with Mouse - libs
libmmonop-0 - massively multiplayer pong game library (shared libraries)
libmmonop-0-dev - massively multiplayer pong game library (development headers)
libmoe-mingmp1 - module for CPU latency measurement
mmonop-caca - massively multiplayer pong game client (caca version)
mmonop-gi - massively multiplayer pong game client (OpenGL version)
mmonop-gi-data - massively multiplayer pong game client data (OpenGL version)
mmonop - massively multiplayer pong game server
```

## STEP 3

Packages are uninstalled from your Raspberry Pi using the remove command. Enter: **sudo apt remove dreamchess** to start deleting the chess game that you installed. An alert saying "Do you want to continue [Y/n]?" appears, enter: **y** and press Return to delete the program.

```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~$ sudo apt remove dreamchess
Reading package lists... Done
Building dependency tree
Reading state information... Done
pi@raspberrypi:~$
pi@raspberrypi:~$ sudo apt remove dreamchess
Reading package lists... Done
Building dependency tree
Reading state information... Done
Use 'apt-get autoremove' to remove them.
The following packages will be REMOVED:
dreamchess
0 upgraded, 0 newly installed, 1 to remove and 21 not upgraded.
After this operation, 459 kB disk space will be freed.
Do you want to continue? [Y/n] y
The following files and directories currently installed:
Removing dreamchess (0.2.0-3) ...
Processing triggers for man-db (2.7.0-2.5) ...
Processing triggers for gnome-menus (3.13.2-3) ...
Processing triggers for desktop-file-utils (0.22-1) ...
Processing triggers for mime-support (3.8) ...
pi@raspberrypi:~$ sudo apt purge dreamchess
Reading package lists... Done
Building dependency tree
```

## STEP 4

Uninstalled packages using **apt** doesn't remove all of the files. Some are left in case you decide to reinstall the program later. To completely remove a package from your Raspberry Pi use the **purge** command. Enter: **sudo apt purge dreamchess** to remove all of the supporting files. You can also use **sudo apt clean** to tidy up your packages and free up some drive space.

```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~$ sudo apt purge dreamchess
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
dreamchess-data libsm1
Use 'apt-get autoremove' to remove them.
The following packages will be REMOVED:
dreamchess
0 upgraded, 0 newly installed, 1 to remove and 21 not upgraded.
After this operation, 459 kB disk space will be freed.
Do you want to continue? [Y/n] y
The following files and directories currently installed:
Removing dreamchess (0.2.0-3) ...
Processing triggers for man-db (2.7.0-2.5) ...
Processing triggers for gnome-menus (3.13.2-3) ...
Processing triggers for desktop-file-utils (0.22-1) ...
Processing triggers for mime-support (3.8) ...
pi@raspberrypi:~$ sudo apt purge dreamchess
Reading package lists... Done
Building dependency tree
```



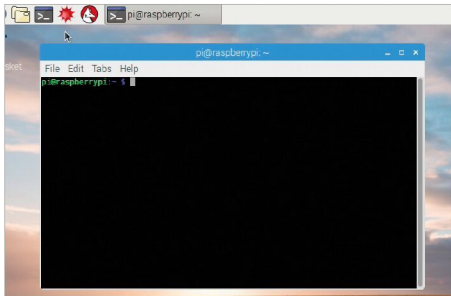
# Get More from the Terminal App

Terminal is an app in Raspbian that enables you to access the command line and issue text commands to your Pi. You'll spend a lot of time in Terminal, so it's a great idea to get to know the app and set it up to work your way.

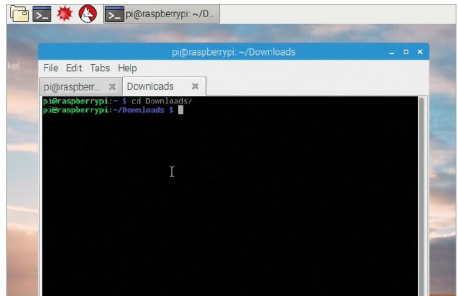
## CUSTOM TERMINAL

The Terminal gives you access to the command line, where the real power of the Linux OS lies. Getting used to the Terminal is key to mastering the Pi and your projects.

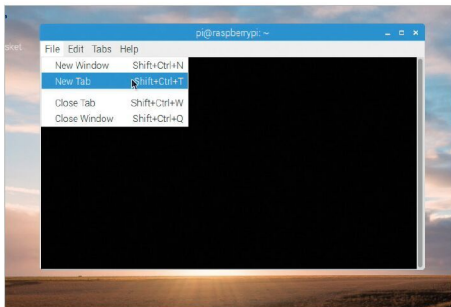
**STEP 1** Start Terminal by clicking the Terminal icon in the Application Launch Bar. You can also start Terminal by pressing Control-Alt-T or choosing Menu > Accessories > Terminal. Terminal emulates the old style video terminals from before desktop interfaces; so by default it displays bright (mostly green) on a black background.



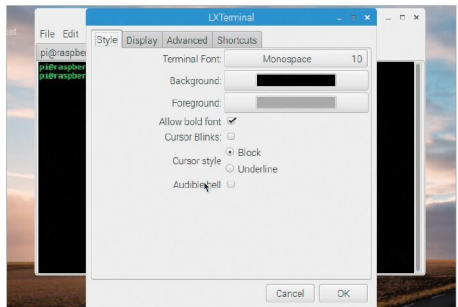
**STEP 3** Naming tabs makes them much easier to recognise. Choose Tabs > Name Tab and enter a name for the tab such as "Home" or "Downloads". Click OK to set the name. You can quickly switch between tabs using Control-Page Up and Control-Page Down and rearrange them using the Tabs Move Tab Left / Move Tab Right options.



**STEP 2** The first thing most people overlook is the ability to run multiple terminals at once in different tabs. Choose File > New Tab (Shift-Control-T) to create a new tab. Navigating multiple directories simultaneously can be a challenge in Terminal and tabs makes it that much easier.



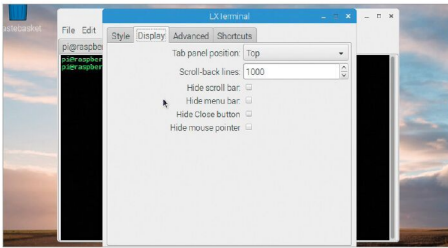
**STEP 4** Choose Edit > Preferences to personalise the look and feel of LXTerminal. One of the best tricks is to click Background and drag the Opacity slider to the half-way point. This enables you to view through the terminal background and see the windows below. You can also personalise the colour of the background.





## STEP 5

While still in Preferences click the Display tab. Here you can adjust the Tab panel positions, placing them on the Left makes them more prominent. You can also adjust the scroll-back line count. This is the number of lines you can scroll up with using the mouse, not the history accessed by the Up and Down arrows.

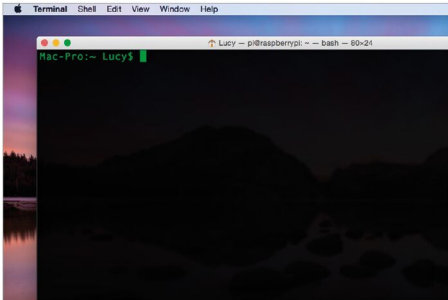


## CONNECT USING SSH

Use a different computer to control your Raspberry Pi.

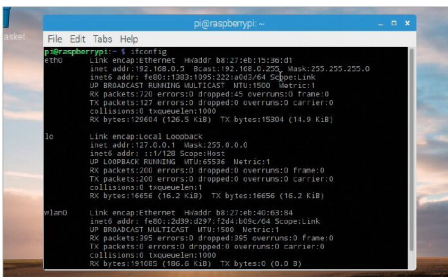
## STEP 1

If you have your Raspberry Pi on the same network as a Mac or Linux-based computer, you can control it using the Terminal program on that computer. We're going to use Terminal in macOS here but the process is the same on a Linux machine. Open the Terminal app on your computer.



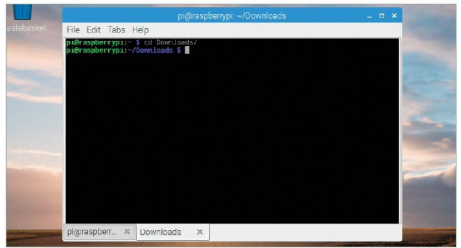
## STEP 2

Enter: `ifconfig` into the Terminal app on your Raspberry Pi. This will let you know which IP address it is using. Look for the four blocks of digits following "inet addr:". They should begin with 192.168.0 followed by a three-digit number. Ours is 192.168.0.179. You need to use that number in Terminal on your Mac to connect.



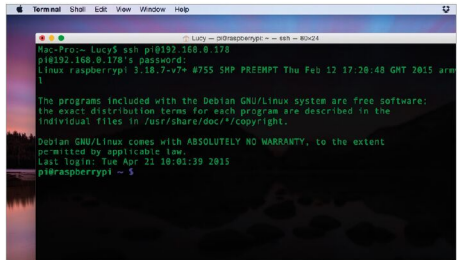
## STEP 6

If you prefer a more minimalist approach try moving the Tabs to the bottom and selecting the Hide scroll bar, Hide menu bar and Hide Close button options. When combined with a low opacity background this makes for a subtle terminal window. You can still access menu settings using a right-click on the mouse.



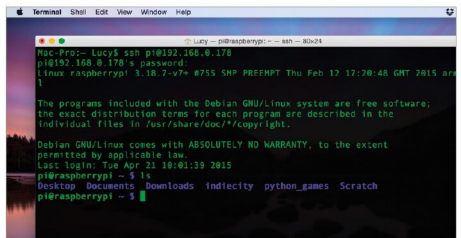
## STEP 3

Switch back to Terminal on your Mac or Linux machine and enter: `ssh pi@192.168.0.178` (using your own IP address). The "pi" bit is the name of the default user account, so if you have changed that it needs to match your user account on the Raspberry Pi. You'll be asked for a password. This is the password that you use to log in to the Raspberry Pi (not your Mac or Linux computer).



## STEP 4

You are now logged in to your Raspberry Pi and can enter commands directly into the Terminal on your Mac or Linux computer. Unlike remote networking you don't see the commands being entered on the screen of the Raspberry Pi, you are accessing the computer from behind the scenes. Many people prefer to set up the Raspberry Pi so they can control it from a more powerful computer. Enter: `exit` to close the connection.



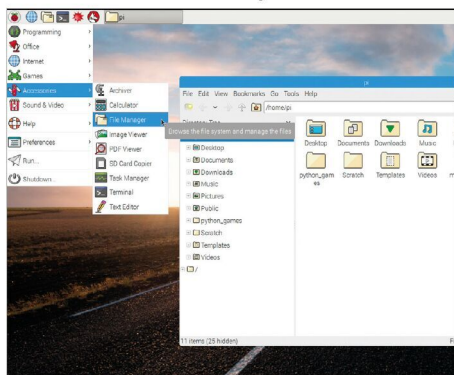
# Using the File Manager

Because you'll be creating programs on your Raspberry Pi, you'll need to know more about the file system than you would on other computers. One vital tool to become familiar with is the File Manager. This is used to find, move and remove files from your Raspberry Pi.

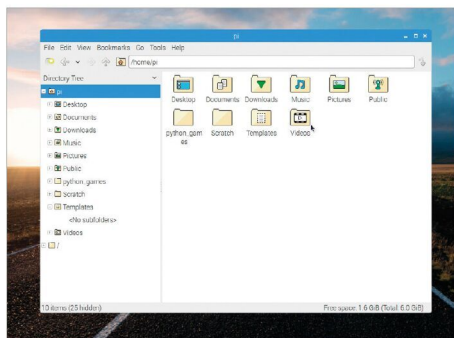
## VIEWING AND MOVING FILES

You can move, manage and delete files using the Command Line but like all modern operating systems Raspbian has a program to help you manage your files. Discover how to use the File Manager app.

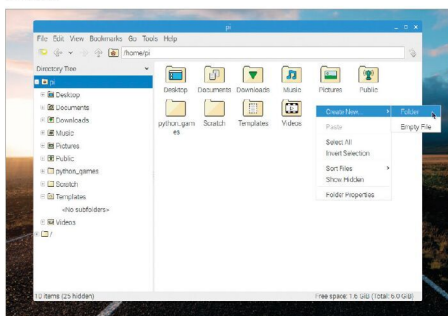
**STEP 1** Raspbian includes a great file management program called "File Manager PCManFM" or just "File Manager" for short. Click on the FileManager PCManFM icon in the Launch Bar or choose Menu > Accessories > File Manager.



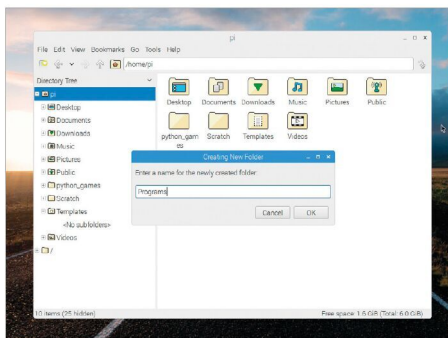
**STEP 2** File Manager displays the folders in your Home folder (this should match your user name, "Pi" by default). By default you should have Desktop, Documents, Downloads, Music, Pictures, Public, python\_games, Scratch and folders. Double-click any folder to open it and view its contents. Click the Parent Folder icon or press Alt-Up Arrow to head back up.



**STEP 3** In the left-hand part of the File Manager sits the Side Pane. By default this displays the Directory Tree, which is another way of navigating the folders on your hard drive. Alternatively click the Side Pane menu and choose Places. Now the Side Pane displays common locations such as Home Folder, Desktop, Rubbish Bin (also known as the Wastebasket), Applications and your SD Card.



**STEP 4** You can create a new folder in the current location by choosing File > Create New > Folder or press Shift-Control-N. Enter a name for the folder and click OK. Files and folders can be dragged on top of one another to move them around. You can also drag files to the folders in the Side Pane, which is an easy way to move them back up the folder tree.

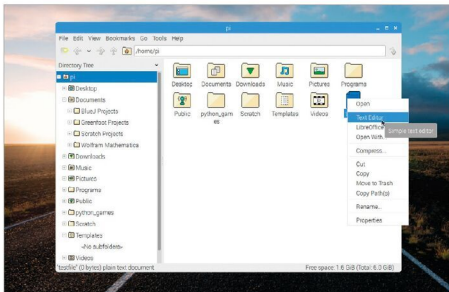






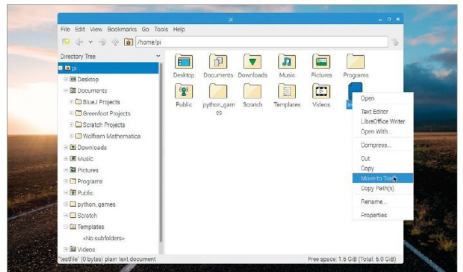
## STEP 5

Double-click a file in File Manager to open it with the default app. You can also right-click a file to view compatible applications in Raspbian. Pick an application from the list provided to open it. Click Properties and use the Open with menu to automatically open that type of file with that app in future.



## STEP 6

Drag items you no longer want to the Wastebasket to delete them or right-click and choose Move to Trash. To empty the wastebasket and permanently delete the unwanted files double click Wastebasket to open it. Now right-click the black space in File Manager and choose Empty Rubbish Bin. Click Yes in the alert window and the files will be removed.

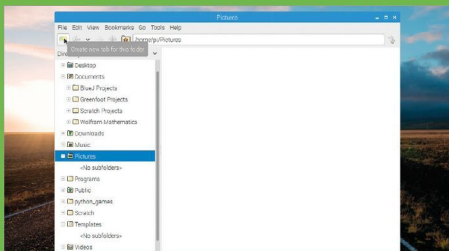


## ADVANCED FILE MANAGER TRICKS

These handy tricks and tips make File Manager more powerful.

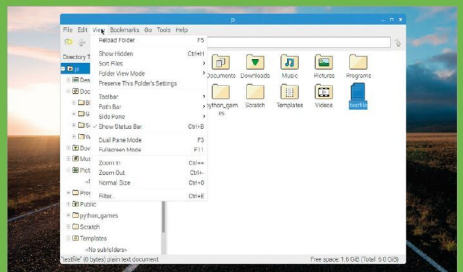
### STEP 1

As you start to move files around you will find navigating between the same folders tiresome. It is possible to open locations in File Manager as tabs, so you can quickly jump back and forth. Click the Create new tab for this folder icon, just below the File menu, and the current folder opens as a tab. Click the tabs to jump between locations.



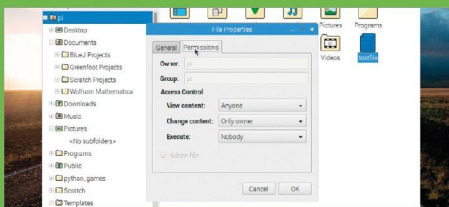
### STEP 3

It's worth taking time to explore the View menu in File Manager. Here you'll find a Show Hidden option, which enables you to view hidden files, as well as Sort Files. You can also adjust the Folder View Mode as well as Toolbar, Path Bar, Side Pane and Status Bar.



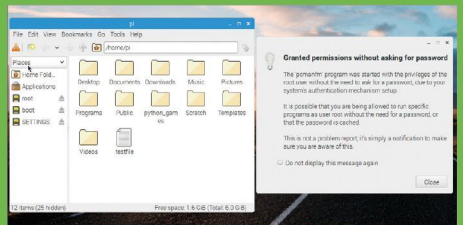
### STEP 2

To view the properties of a file right-click and choose Properties. Here you can view information about the File Type and the Open With application. More importantly, if you click Permissions you can view the permissions associated with that file. The concept of file permissions, especially the 'Execute' setting, becomes increasingly important as you become more advanced in using your Raspberry Pi.



### STEP 4

There are times when you will want to move files but find you can't because you don't have root (`sudo`) access in File Manager. If you want to open File Manager with root mode choose Menu > Run and enter: `gksu pman.fm`. File Manager will open and display root and boot() folders. Be careful when running File Manager in root mode, and close it when you are finished.





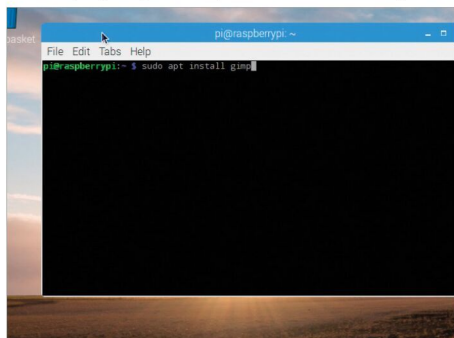
# Edit Images with GIMP

The name may be funny, but this image editing app is incredibly serious. The Raspberry Pi is perfectly adept at photo editing, and you can use GIMP to create icons, images and works of art for your programs.

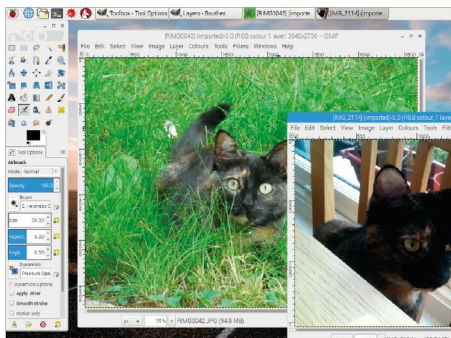
## GET TO KNOW GIMP

Your Raspberry Pi can display images with its built-in Image View app but for any image editing you'll need GIMP (GNU Image Manipulation Program). GIMP is a powerful software package for photo editing and is a great tool to install on your Raspberry Pi.

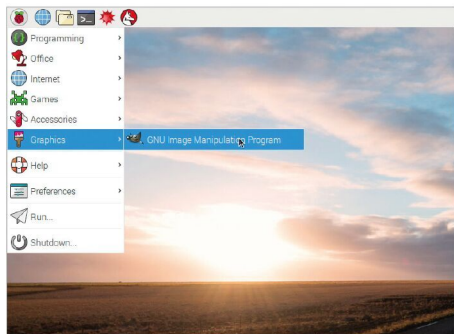
**STEP 1** Open terminal and enter: `sudo apt install gimp` to download and install the GIMP package. An alert will appear saying: "after this operation, 117 MB of additional disk space will be used. Do you want to continue [Y/n]?" Enter: `y` and press **Return**. GIMP will now be installed in Raspbian. GIMP is a lot larger than most programs you'll install, so the installation takes longer.



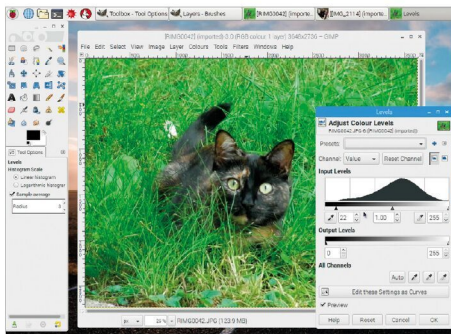
**STEP 3** Images can be opened in GIMP using File > Open or by right-clicking on files in File Manager and choosing GNU Image Manipulation Program. You can open multiple images at once in GIMP but each opens inside a different window. Choose Window > Single Window Mode to gather them together. Click on the tabs at the top of the screen to switch from one image to another.



**STEP 2** When the installation has completed you will find GIMP under Menu > Graphics > GNU Image Manipulation Program. It has a more detailed interface than many Linux programs with two boxes, Tool Options and Brushes offering a range of image editing options. Users of software like Adobe Photoshop will feel right at home and it's ideal for image editing.



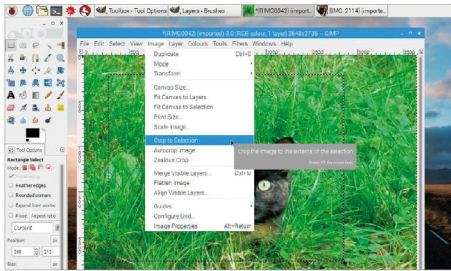
**STEP 4** You can perform powerful edits in GIMP but we don't have space to go over them all. Probably the most useful you'll find is Colours > Levels. This window enables you to adjust the tonal range and colour balance of an image. Drag the left and right handles in slightly and slide the centre handle to the right to improve the contrast of an image.





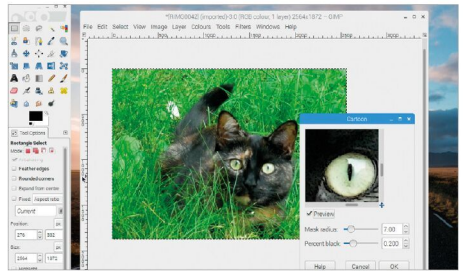
## STEP 5

To crop an image click the Rectangle Select Tool and drag a square on the image. Choose Image > Crop To Selection to remove the unwanted parts of the image. Use the Image > Scale Image and Image > Canvas Size options to adjust the image to specific sizes. Cropping and resizing images is a vital technique to know when working on websites.



## STEP 6

There are a huge range of image effects and filters you can add to images using GIMP. Far more than we have room to cover here. Try Filters > Artistic > Cartoon to give your image a black outline or Filters > Artistic > Ollivie to simulate a painted artwork. Take a look at [www.gimp.org/tutorials](http://www.gimp.org/tutorials) for creative inspiration.

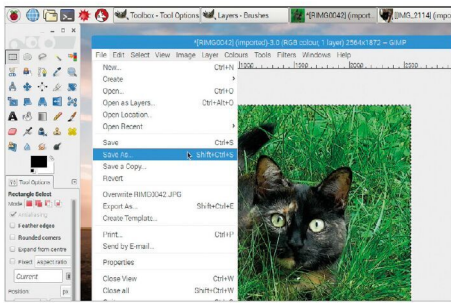


## EXPORTING IMAGES

Save your images for use in a website.

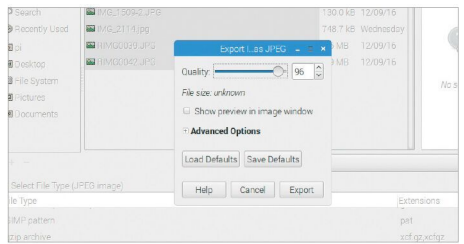
## STEP 1

Images are saved using File > Save As in the xcf format (Experimental Computing Facility). This is GIMP's native format so should only be used to save files you want to view and work on in GIMP.



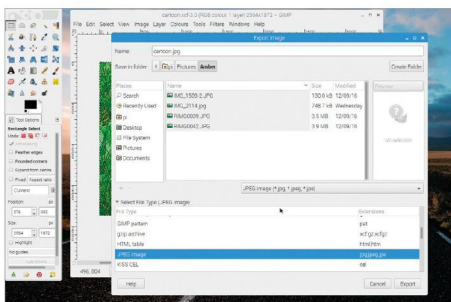
## STEP 3

The Export Image as JPEG window appears, displaying a Quality slider. The Quality range is from 0 (very poor) to 100 (perfect). While you might be tempted to set the quality at 100, reducing it slightly will create much smaller files. Smaller files ensure that your web page loads much more quickly. You can typically reduce the Quality to 80 and get a much smaller file with little discernible difference.



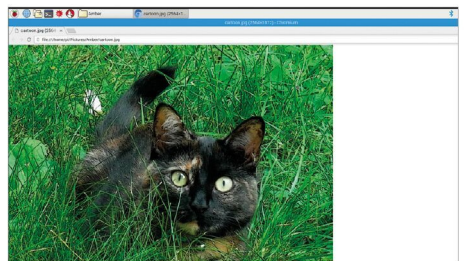
## STEP 2

If you want to save files for use in projects, such as websites, use File > Export. Click the File Type option and choose a file type; typically you will use JPEG for web images. Choose a Name and location and click Export.



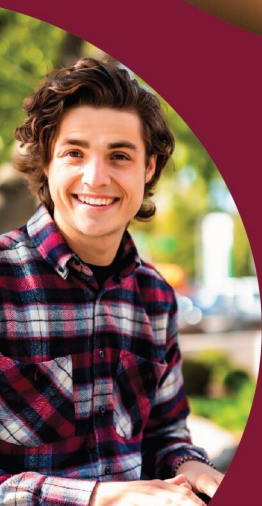
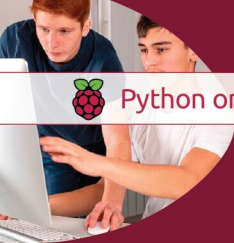
## STEP 4

Right-click a JPEG image in the File Manager and choose Open With. Expand Internet and choose Chromium Web Browser and OK. This enables you to see how it will appear when you add it to your website. You can also view images more quickly by right clicking and choosing Image Viewer. You can use Save File As in Image Viewer to quickly change the image format.





# Python on the Pi





# Python on the Pi

Being able to code is part of making not just the Raspberry Pi, but all the connected devices that can access the Internet, do what you want them to. The Raspberry Pi is a perfect coding base that comes pre-installed with the latest version of one of the world's most popular programming languages, Python.

Python is a powerful yet easy to understand programming language that enables you to do anything, from displaying simple messages on the screen to producing action-packed arcade games. In this section of the book, you will learn how to get Python working and how to code your first Python program.

Learning the tricks, hacks and fixes of Python will enable you to create something brilliant that can be shared with everyone who owns a Raspberry Pi.

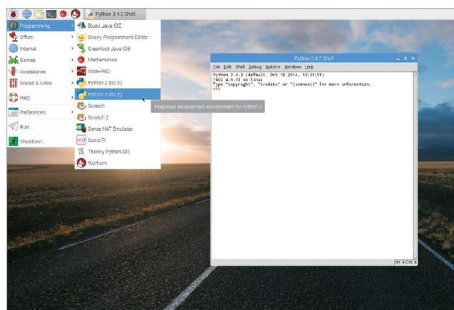
# Starting Python for the First Time

If you're using the new Raspberry Pi, together with its latest release of Raspbian, then you will need to manually install the Python IDLE. This is due to the Pi team removing the core Python IDLE in favour of replacing it with their own coding text editor.

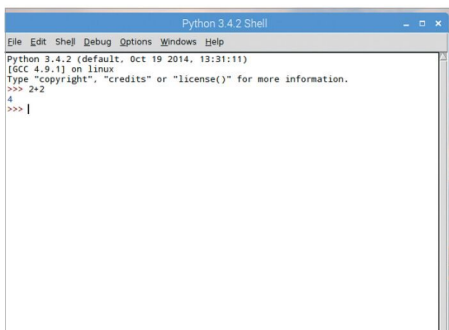
## STARTING PYTHON

For those using the Pi 4 and new Raspbian, drop into a Terminal and enter: `sudo apt-get install idle3`. Older versions of Raspbian already have the official Python IDLE pre-installed.

**STEP 1** With the Raspbian desktop loaded, click on the Menu button followed by Programming > Python 3 (IDLE). This will open the Python 3 Shell. Windows and Mac users can find the Python 3 IDLE Shell from within the Windows Start button menu and via Finder.

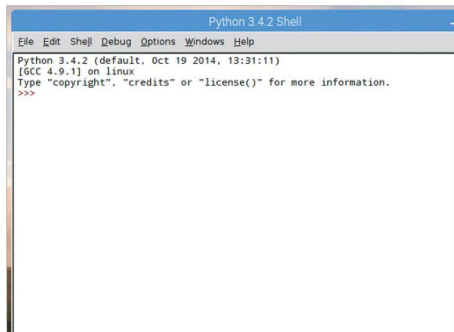


**STEP 3** For example, in the Shell enter: `2+2`. After pressing Enter, the next line will display the answer: 4. Basically, Python has taken the 'code' and produced the relevant output.



**STEP 2** The Shell is where you can enter code and see the responses and output of code you've programmed into Python. This is a kind of sandbox, where you're able to try out some simple code and processes.

**STEP 4** The Python Shell acts very much like a calculator, since code is basically a series of mathematical interactions with the system. Integers, which are the infinite sequence of whole numbers can easily be added, subtracted, multiplied and so on.





## STEP 5

While that's very interesting, it's not particularly exciting. Instead, try this:

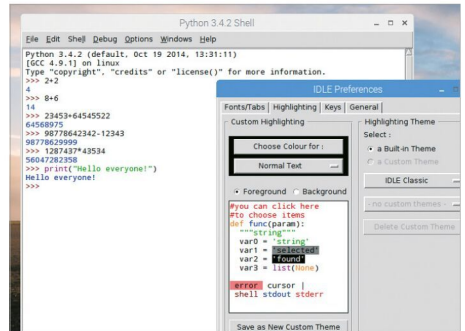
```
print("Hello everyone!")
```

Just enter it into the IDLE as you've done in the previous steps.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> 2+2
4
>>> 8+6
14
>>> 23453+64545522
64568975
>>> 98778642342-12343
98778629999
>>> 1287437*43534
56047282358
>>> print("Hello everyone!")
Hello everyone!
>>>
```

## STEP 8

The Python IDLE is a configurable environment. If you don't like the way the colours are represented, then you can always change them via Options > Configure IDLE and clicking on the Highlighting tab. However, we don't recommend that as you won't be seeing the same as our screenshots.



## STEP 6

This is a little more like it, since you've just produced your first bit of code. The Print command is fairly self-explanatory, it prints things. Python 3 requires the brackets as well as quote marks in order to output content to the screen, in this case the Hello everyone! bit.

```
>>> print("Hello everyone!")
Hello everyone!
>>> |
```

## STEP 9

Just like most programs available, regardless of the operating system, there are numerous shortcut keys available. We don't have room for them all here but within the Options > Configure IDLE and under the Keys tab, you can see a list of the current bindings.



## STEP 7

You may have noticed the colour coding within the Python IDLE. The colours represent different elements of Python code. They are:

- Black – Data and Variables
- Green – Strings
- Purple – Functions
- Orange – Commands
- Blue – User Functions
- Dark Red – Comments
- Light Red – Error Messages

### IDLE Colour Coding

Colour	Use for	Examples
Black	Data & variables	23.6 area
Green	Strings	"Hello World"
Purple	Functions	len() print()
Orange	Commands	if for else
Blue	User functions	get_area()
Dark red	Comments	#Remember VAT
Light red	Error messages	SyntaxError:

## STEP 10

The Python IDLE is a powerful interface, and one that's actually been written in Python using one of the available GUI toolkits. If you want to know the many ins and outs for the Shell, we recommend you take a few moments to view [www.docs.python.org/3/library/idle.html](http://www.docs.python.org/3/library/idle.html), which details many of the IDLE's features.

25.5. IDLE

Source code: [Cnididid](#)

IDLE is Python's Integrated Development and Learning Environment.

IDLE has the following features:

- color in multi-page Python interactive (IDLE Shell)
- color (python) syntax across the same as windows, can use color
- Python shell window interactive (interactive window) of code input, and error messages
- multi-window text editor with multi-line, Python comments, insert, insert, cut, file, auto completion, and other features
- auto completion window, multi-line (interactive) window, and search through files (shell)
- debugger with interactive (interactive, stepping, and viewing of global and local namespaces)
- configuration, resources, and other things

25.5.1. Menus

IDLE has two main window types, the Shell window and the Editor window. It is possible to have multiple editor windows simultaneously (split windows), such as used for File / Open or File, and a separate shell for the interactive mode as Editor window for a different sub-shell or control mode.

IDLE's menus are dynamically changed based on which window is currently selected. Each menu item below indicates which window type it is associated with.

25.5.1.1. File menu (Shell and Editor)

New File  
 Create a new file editing window.

Open...  
 Open an existing file with an Open dialog.

Recent files...  
 Open a list of recent files. Click on file to open it.

Open Recent...



# Your First Code

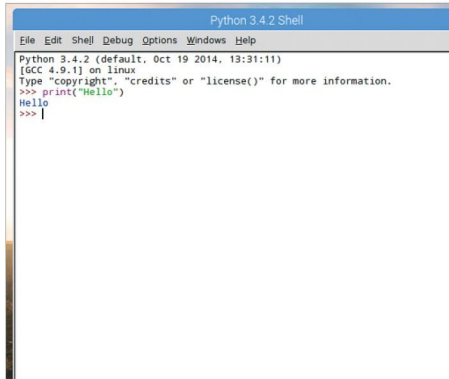
Essentially, you've already written your first piece of code with the 'print("Hello everyone!")' function from the previous tutorial. However, let's expand that and look at entering your code and playing around with some other Python examples.

## PLAYING WITH PYTHON

With most languages, computer or human, it's all about remembering and applying the right words to the right situation. You're not born knowing these words, so you need to learn them.

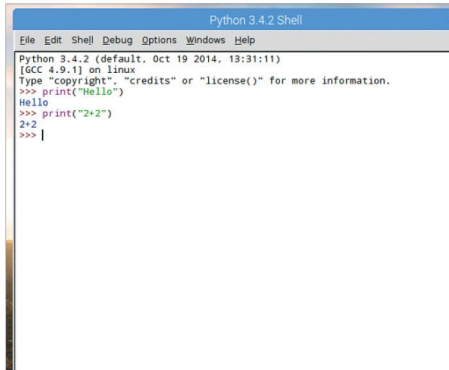
**STEP 1** If you've closed Python 3 IDLE, reopen it in whichever operating system version you prefer. In the Shell, enter the familiar following:

```
print("Hello")
```



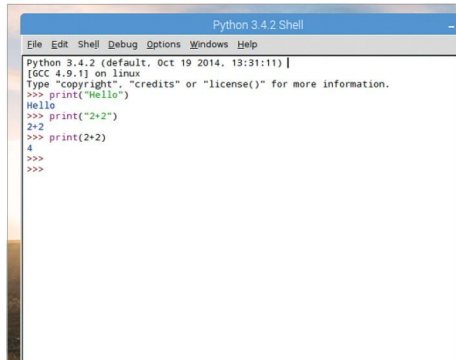
**STEP 2** Just as predicted, the word Hello appears in the Shell as blue text, indicating output from a string. It's fairly straightforward and doesn't require too much explanation. Now try:

```
print("2+2")
```



**STEP 3** You can see that instead of the number 4, the output is the 2+2 you asked to be printed to the screen. The quotation marks are defining what's being outputted to the IDLE Shell; to print the total of 2+2 you need to remove the quotes:

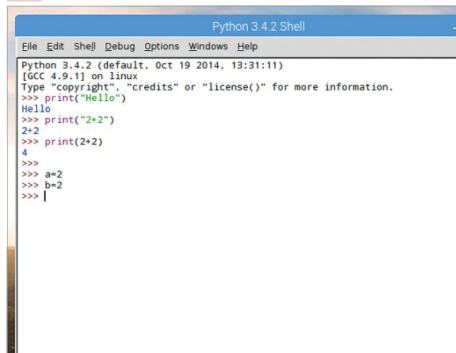
```
print(2+2)
```



**STEP 4** You can continue as such, printing 2+2, 464+2343 and so on to the Shell. An easier way is to use a variable, which is something we will cover in more depth later. For now, enter:

```
a=2
```

```
b=2
```





**STEP 5**

What you have done here is assign the letters a and b two values: 2 and 2. These are now variables, which can be called upon by Python to output, add, subtract, divide and so on for as long as their numbers stay the same. Try this:

```
print(a)
print(b)
```

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> print("Hello")
Hello
>>> print("2*2")
2*2
>>> print(2*2)
4
>>>
>>> a=2
>>> b=2
>>> print(a)
2
>>> print(b)
2
>>> |
```

**STEP 6**

The output of the last step displays the current values of both a and b individually, as you've asked them to be printed separately. If you want to add them up, you can use the following:

```
print(a+b)
```

This code simply takes the values of a and b, adds them together and outputs the result.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> print("Hello")
Hello
>>> print("2*2")
2*2
>>> print(2*2)
4
>>>
>>> a=2
>>> b=2
>>> print(a)
2
>>> print(b)
2
>>> print(a+b)
4
>>> |
```

**STEP 7**

You can play around with different kinds of variables and the Print function. For example, you could assign variables for someone's name:

```
name="David"
print(name)
```

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> print("Hello")
Hello
>>> print("2*2")
2*2
>>> print(2*2)
4
>>>
>>> a=2
>>> b=2
>>> print(a)
2
>>> print(b)
2
>>> print(a+b)
4
>>> name="David"
>>> print(name)
David
>>> |
```

**STEP 8**

Now let's add a surname:

```
surname="Hayward"
```

```
print(surname)
```

You now have two variables containing a first name and a surname and you can print them independently.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> name="David"
>>> print(name)
David
>>> surname="Hayward"
>>> print(surname)
Hayward
>>> |
```

**STEP 9**

If we were to apply the same routine as before, using the + symbol, the name wouldn't appear correctly in the output in the Shell. Try it:

```
print(name+surname)
```

You need a space between the two, defining them as two separate values and not something you mathematically play around with.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> name="David"
>>> print(name)
David
>>> surname="Hayward"
>>> print(surname)
Hayward
>>> print(name+surname)
DavidHayward
>>> |
```

**STEP 10**

In Python 3 you can separate the two variables with a space using a comma:

```
print(name, surname)
```

Alternatively, you can add the space ourselves:

```
print(name+" "+surname)
```

The use of the comma is much neater, as you can see. Congratulations, you've just taken your first steps into the wide world of Python.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> name="David"
>>> print(name)
David
>>> surname="Hayward"
>>> print(surname)
Hayward
>>> print(name+surname)
DavidHayward
>>> print(name, surname)
David Hayward
>>> print(name+" "+surname)
David Hayward
>>> |
```

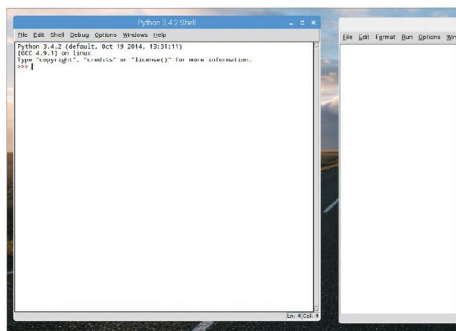
# Saving and Executing Your Code

While working in the IDLE Shell is perfectly fine for small code snippets, it's not designed for entering longer program listings. In this section you're going to be introduced to the IDLE Editor, where you will be working from now on.

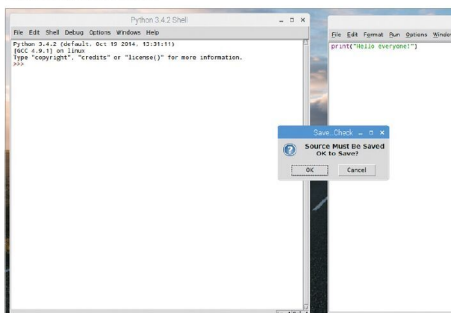
## EDITING CODE

You will eventually reach a point where you have to move on from inputting single lines of code into the Shell. Instead, the IDLE Editor will allow you to save and execute your Python code.

**STEP 1** First, open the Python IDLE Shell and when it's up, click on File > New File. This will open a new window with Untitled as its name. This is the Python IDLE Editor and within it you can enter the code needed to create your future programs.

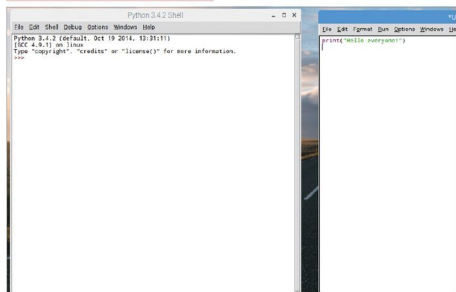


**STEP 3** You can see that the same colour coding is in place in the IDLE Editor as it is in the Shell, enabling you to better understand what's going on with your code. However, to execute the code you need to first save it. Press F5 and you get a Save...Check box open.

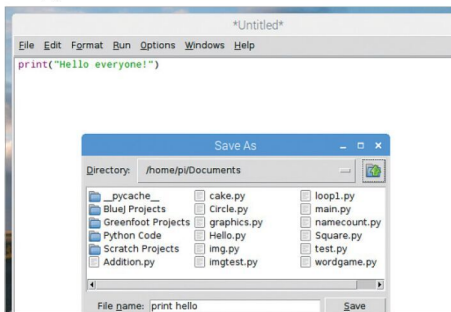


**STEP 2** The IDLE Editor is, for all intents and purposes, a simple text editor with Python features, colour coding and so on; much in the same vein as Sublime. You enter code as you would within the Shell, so taking an example from the previous tutorial, enter:

```
print("Hello everyone!")
```



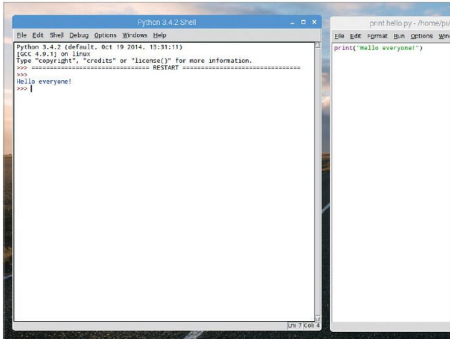
**STEP 4** Click on the OK button in the Save box and select a destination where you'll save all your Python code. The destination can be a dedicated folder called Python or you can just dump it wherever you like. Remember to keep a tidy drive though, to help you out in the future.





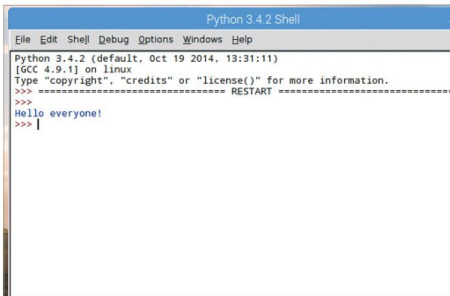
## STEP 5

Enter a name for your code, 'print hello' for example, and click on the Save button. Once the Python code is saved it's executed and the output will be detailed in the IDLE Shell. In this case, the words 'Hello everyone!'.



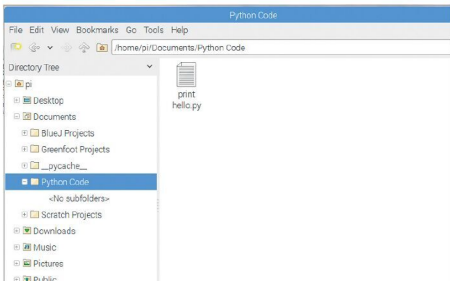
## STEP 6

This is how the vast majority of your Python code will be conducted. Enter it into the Editor, hit F5, save the code and look at the output in the Shell. Sometimes things will differ, depending on whether you've requested a separate window, but essentially that's the process. It's the process we will use throughout this book, unless otherwise stated.



## STEP 7

If you open the file location of the saved Python code, you can see that it ends in a .py extension. This is the default Python file name. Any code you create will be whatever.py and any code downloaded from the many internet Python resource sites will be .py. Just ensure that the code is written for Python 3.

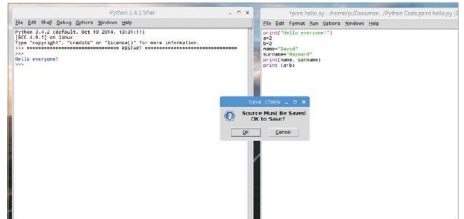


## STEP 8

Let's extend the code and enter a few examples from the previous tutorial:

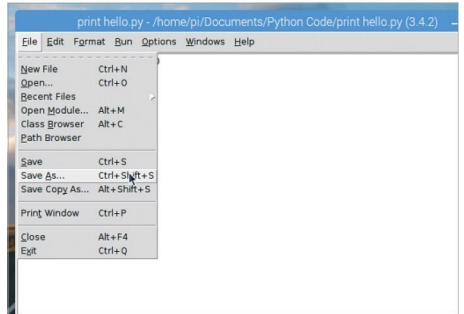
```
a=2
b=2
name="David"
surname="Hayward"
print(name, surname)
print(a+b)
```

If you press F5 now you'll be asked to save the file, again, as it's been modified from before.



## STEP 9

If you click the OK button, the file will be overwritten with the new code entries, and executed, with the output in the Shell. It's not a problem with just these few lines but if you were to edit a larger file, overwriting can become an issue. Instead, use File > Save As from within the Editor to create a backup.

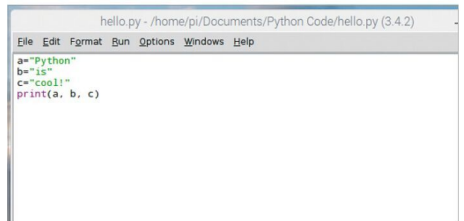


## STEP 10

Now create a new file. Close the Editor, and open a new instance (File > New File from the Shell). Enter the following and save it as hello.py:

```
a="Python"
b="is"
c="cool!"
print(a, b, c)
```

You will use this code in the next tutorial.



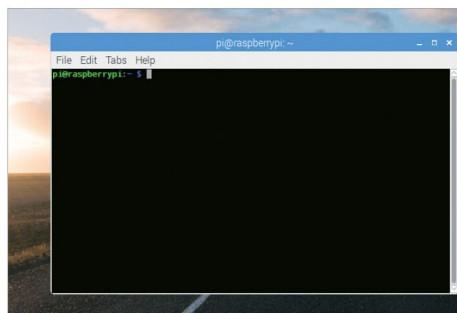
# Executing Code from the Command Line

Although we're working from the GUI IDLE throughout this book, it's worth taking a look at Python's command line handling. We already know there's a command line version of Python but it's also used to execute code.

## COMMAND THE CODE

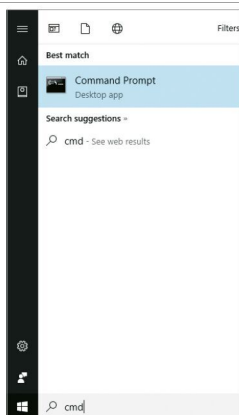
Using the code we created in the previous tutorial, the one we named `hello.py`, let's see how you can run code that was made in the GUI at the command line level.

**STEP 1** Python, in Linux, comes with two possible ways of executing code via the command line. One of the ways is with Python 2, whilst the other uses the Python 3 libraries and so on. First though, drop into the command line or Terminal on your operating system.

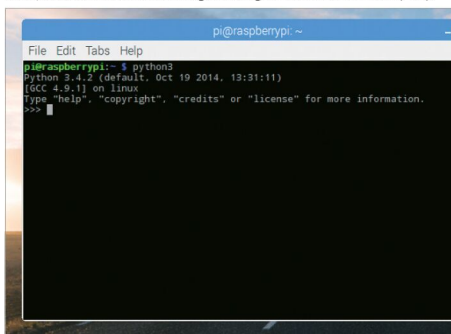


**STEP 2** Just as before, we're using a

Raspberry Pi: Windows users will need to click the Start button and search for CMD, then click the Command Line returned search; and macOS users can get access to their command line by clicking Go > Utilities > Terminal.



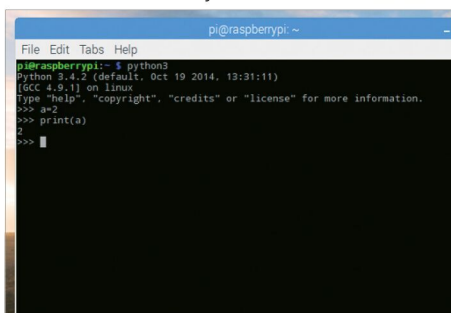
**STEP 3** Now you're at the command line we can start Python. For Python 3 you need to enter the command `python3` and press Enter. This will put you into the command line version of the Shell, with the familiar three right-facing arrows as the cursor (`>>>`).



**STEP 4** From here you're able to enter the code you've looked at previously, such as:

```
a=2
print(a)
```

You can see that it works exactly the same.





**STEP 5** Now enter: `exit()` to leave the command line Python session and return you back to the command prompt. Enter the folder where you saved the code from the previous tutorial and list the available files within; hopefully you should see the `hello.py` file.

```

pi@raspberrypi: ~/Documents/Python Code
File Edit Tabs Help
pi@raspberrypi:~$ python3
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a=2
>>> print(a)
2
>>> exit()
pi@raspberrypi:~$ cd Documents/
pi@raspberrypi:~/Documents $ cd Python\ Code/
pi@raspberrypi:~/Documents/Python Code $ ls
hello.py print hello.py
pi@raspberrypi:~/Documents/Python Code $
    
```

**STEP 6** From within the same folder as the code you're going to run, enter the following into the command line:

```

python3 hello.py
This will execute the code we created, which to remind you is:
a="Python"
b="is"
c="cool!"
print(a, b, c)
    
```

```

pi@raspberrypi: ~/Documents/Python Code
File Edit Tabs Help
pi@raspberrypi:~$ python3
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a=2
>>> print(a)
2
>>> exit()
pi@raspberrypi:~$ cd Documents/
pi@raspberrypi:~/Documents $ cd Python\ Code/
pi@raspberrypi:~/Documents/Python Code $ ls
hello.py print hello.py
pi@raspberrypi:~/Documents/Python Code $ python3 hello.py
Python is cool!
pi@raspberrypi:~/Documents/Python Code $
    
```

**STEP 7** Naturally, since this is Python 3 code, using the syntax and layout that's unique to Python 3, it only works when you use the `python3` command. If you like, try the same with Python 2 by entering:

```

pi@raspberrypi: ~/Documents/Python Code
File Edit Tabs Help
pi@raspberrypi:~$ python3
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a=2
>>> print(a)
2
>>> exit()
pi@raspberrypi:~$ cd Documents/
pi@raspberrypi:~/Documents $ cd Python\ Code/
pi@raspberrypi:~/Documents/Python Code $ ls
hello.py print hello.py
pi@raspberrypi:~/Documents/Python Code $ python3 hello.py
Python is cool!
pi@raspberrypi:~/Documents/Python Code $ python hello.py
('Python', 'is', 'cool!')
pi@raspberrypi:~/Documents/Python Code $
    
```

**STEP 8** The result of running Python 3 code from the Python 2 command line is quite obvious. Whilst it doesn't error out in any way, due to the differences between the way Python 3 handles the Print command over Python 2, the result isn't as we expected. Using Sublime for the moment, open the `hello.py` file.

```

C:\Users\david\Documents\Python\hello.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
hello.py
1 a="Python"
2 b="is"
3 c="cool!"
4 print(a, b, c)
5
    
```

**STEP 9** Since Sublime Text isn't available for the Raspberry Pi, you're going to temporarily leave the Pi for the moment and use Sublime as an example that you don't necessarily need to use the Python IDE. With the `hello.py` file open, alter it to include the following:

```

name=input("What is your name? ")
print("Hello," , name)
    
```

```

C:\Users\david\Documents\Python\hello.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
hello.py
1 a="Python"
2 b="is"
3 c="cool!"
4 print(a, b, c)
5 name=input("What is your name? ")
6 print("Hello," , name)
7
    
```

**STEP 10** Save the `hello.py` file and drop back to the command line. Now execute the newly saved code with:

`python3 hello.py`  
The result will be the original Python is cool! statement, together with the added input command asking you for your name, and displaying it in the command window.

```

pi@raspberrypi: ~/Documents/Python Code
File Edit Tabs Help
pi@raspberrypi:~/Documents/Python Code $ python3 hello.py
Python is cool!
What is your name? David
Hello, David
pi@raspberrypi:~/Documents/Python Code $
    
```

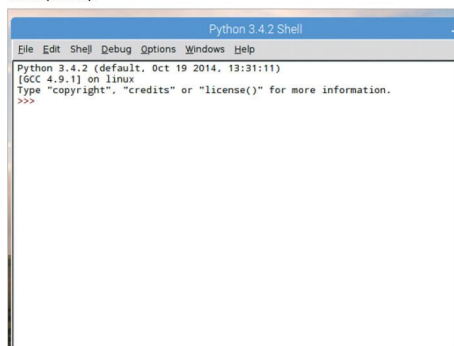
# Numbers and Expressions

We've seen some basic mathematical expressions with Python, simple addition and the like. Let's expand on that now and see just how powerful Python is as a calculator. You can work within the IDLE Shell or in the Editor, whichever you like.

## IT'S ALL MATHS, MAN

You can get some really impressive results with the mathematical powers of Python; as with most, if not all, programming languages, maths is the driving force behind the code.

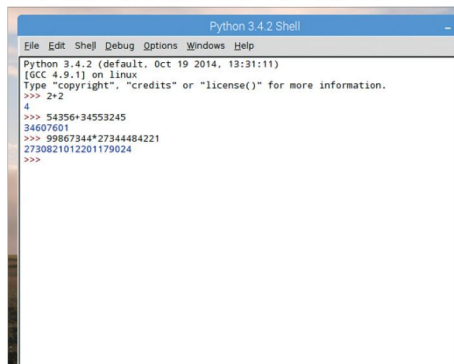
**STEP 1** Open up the GUI version of Python 3, as mentioned you can use either the Shell or the Editor. For the time being, you're going to use the Shell just to warm our maths muscle, which we believe is a small gland located at the back of the brain (or not).



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
```

**STEP 2** In the Shell enter the following:  
`2+2`  
`54356+34553245`  
`99867344*27344484221`

You can see that Python can handle some quite large numbers.

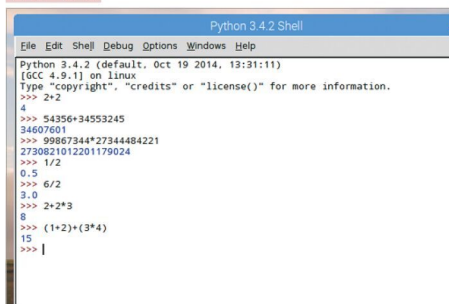


```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> 2+2
4
>>> 54356+34553245
34607601
>>> 99867344*27344484221
2730821012201179024
>>>
```

**STEP 3** You can use all the usual mathematical operations: divide, multiply, brackets and so on. Practise with a

few, for example:

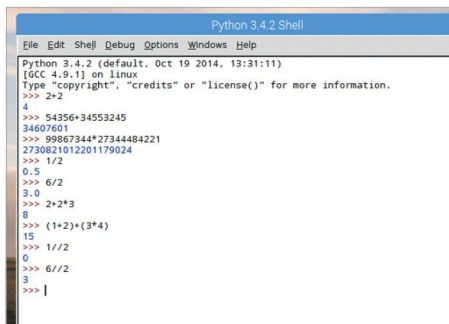
`1/2`  
`6/2`  
`2+2*3`  
`(1+2)+(3*4)`



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> 2+2
4
>>> 54356+34553245
34607601
>>> 99867344*27344484221
2730821012201179024
>>> 1/2
0.5
>>> 6/2
3.0
>>> 2+2*3
8
>>> (1+2)+(3*4)
15
>>> |
```

**STEP 4** You've no doubt noticed, division produces a decimal number. In Python these are called floats, or floating point arithmetic. However, if you need an integer as opposed to a decimal answer, then you can use a double slash:

`1//2`  
`6//2`  
 And so on.



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> 2+2
4
>>> 54356+34553245
34607601
>>> 99867344*27344484221
2730821012201179024
>>> 1/2
0.5
>>> 6/2
3.0
>>> 2+2*3
8
>>> (1+2)+(3*4)
15
>>> 1//2
0
>>> 6//2
3
>>> |
```



## STEP 5

You can also use an operation to see the remainder left over from division. For example: `10/3`

Will display `3.333333333`, which is of course 3.3-recurring.

If you now enter: `10%3`

This will display `1`, which is the remainder left over from dividing `10` by `3`.

```
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> 2+2
4
>>> 54356+34553245
34607601
>>> 99867344*27344484221
2730821012201179024
>>> 1/2
0.5
>>> 6/2
3.0
>>> 2*2*3
8
>>> (1+2)*(3*4)
15
>>> 1//2
0
>>> 6//2
3
>>> 10/3
3.3333333333333335
>>> 10%3
1
>>>
```

## STEP 6

Next up we have the power operator, or exponentiation if you want to be technical. To work out the power of something you can use a double multiplication symbol or double-star on the keyboard:

`2**3`

`10**10`

Essentially, it's `2x2x2` but we're sure you already know the basics behind maths operators. This is how you would work it out in Python.

```
>>> 99867344*27344484221
2730821012201179024
>>> 1/2
0.5
>>> 6/2
3.0
>>> 2*2*3
8
>>> (1+2)*(3*4)
15
>>> 1//2
0
>>> 6//2
3
>>> 10/3
3.3333333333333335
>>> 10%3
1
>>> 2**3
8
>>> 10**10
10000000000
>>>
```

## STEP 7

Numbers and expressions don't stop there. Python has numerous built-in functions to work out sets of numbers, absolute values, complex numbers and a host of mathematical expressions and Pythagorean tongue-twisters. For example, to convert a number to binary, use:

`bin(3)`

```
2730821012201179024
>>> 1/2
0.5
>>> 6/2
3.0
>>> 2*2*3
8
>>> (1+2)*(3*4)
15
>>> 1//2
0
>>> 6//2
3
>>> 10/3
3.3333333333333335
>>> 10%3
1
>>> 2**3
8
>>> 10**10
10000000000
>>> bin(3)
'0b11'
>>> |
```

## STEP 8

This will be displayed as `'0b11'`, converting the integer into binary and adding the prefix `0b` to the front. If you want to remove the `0b` prefix, then you can use:

`format(3, 'b')`

The `Format` command converts a value, the number `3`, to a formatted representation as controlled by the format specification, the `'b'` part.

```
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> 2+2
4
>>> 54356+34553245
34607601
>>> 99867344*27344484221
2730821012201179024
>>> 1/2
0.5
>>> 6/2
3.0
>>> 2*2*3
8
>>> (1+2)*(3*4)
15
>>> 1//2
0
>>> 6//2
3
>>> 10/3
3.3333333333333335
>>> 10%3
1
>>> 2**3
8
>>> 10**10
10000000000
>>> bin(3)
'0b11'
>>> format(3, 'b')
'11'
>>>
```

## STEP 9

A Boolean Expression is a logical statement that will either be true or false. We can use these to compare data and test to see if it's equal to, less than or greater than. Try this in a New File:

```
a = 6
b = 7
print(1, a == 6)
print(2, a == 7)
print(3, a == 6 and b == 7)
print(4, a == 7 and b == 7)
print(5, not a == 7 and b == 7)
print(6, a == 7 or b == 7)
print(7, not a == 7 and b == 6)
print(8, not (a == 7 and b == 6))
print(9, not a == 7 and b == 6)
```

```
BooleanTest.py - /home/pi/D
File Edit Format Run Options Window
a = 6
b = 7
print(1, a == 6)
print(2, a == 7)
print(3, a == 6 and b == 7)
print(4, a == 7 and b == 7)
print(5, not a == 7 and b == 7)
print(6, a == 7 or b == 7)
print(7, not a == 7 and b == 6)
print(8, not (a == 7 and b == 6))
print(9, not a == 7 and b == 6)
```

## STEP 10

Execute the code from Step 9, and you can see a series of True or False statements, depending on the result of the two defining values: `6` and `7`. It's an extension of what you've looked at, and an important part of programming.

```
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> -----RESTART-----
1 True
2 False
3 True
4 False
5 True
6 True
7 False
8 True
9 False
>>> |
```

# Using Comments

When writing your code, the flow, what each variable does, how the overall program will operate and so on is all inside your head. Another programmer could follow the code line by line but over time, it can become difficult to read.

## #COMMENTS!

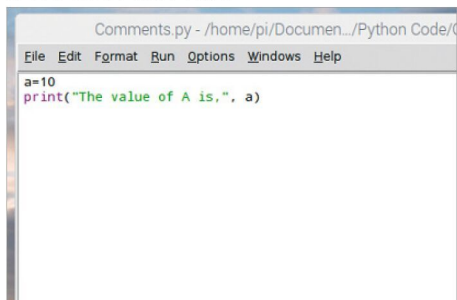
Programmers use a method of keeping their code readable by commenting on certain sections. If a variable is used, the programmer comments on what it's supposed to do, for example. It's just good practise.

**STEP 1** Start by creating a new instance of the IDLE Editor (File > New File) and create a simple variable and print

command:

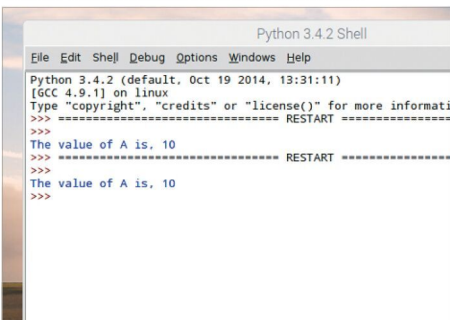
```
a=10
print("The value of A is,", a)
```

Save the file and execute the code.



```
Comments.py - /home/pi/Documen.../Python Code/C
File Edit Format Run Options Windows Help
a=10
print("The value of A is,", a)
```

**STEP 3** Resave the code and execute it. You can see that the output in the IDLE Shell is still the same as before, despite the extra lines being added. Simply put, the hash symbol (#) denotes a line of text the programmer can insert to inform them, and others, of what's going on without the user being aware.

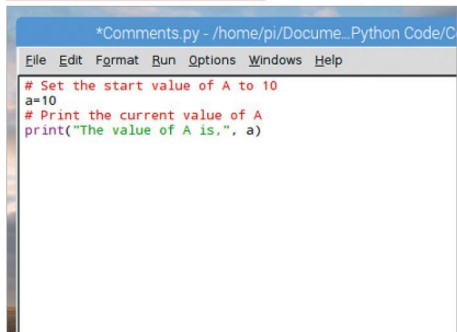


```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more informati
>>> ===== RESTART =====
>>>
The value of A is, 10
>>>
>>> ===== RESTART =====
>>>
The value of A is, 10
>>>
```

**STEP 2** Running the code will return the line: The value of A is, 10 into the IDLE Shell window, which is what we expected. Now, add some of the types of comments you'd normally see within code:

```
# Set the start value of A to 10
```

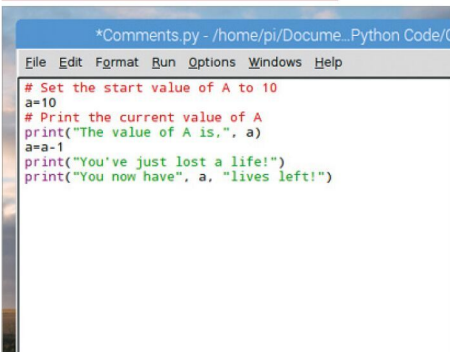
```
a=10
# Print the current value of A
print("The value of A is,", a)
```



```
*Comments.py - /home/pi/Docume...Python Code/C
File Edit Format Run Options Windows Help
# Set the start value of A to 10
a=10
# Print the current value of A
print("The value of A is,", a)
```

**STEP 4** Let's assume that the variable A that we've created is the number of lives in a game. Every time the player dies, the value is decreased by 1. The programmer could insert a routine along the lines of:

```
a=a-1
print("You've just lost a life!")
print("You now have", a, "lives left!")
```



```
*Comments.py - /home/pi/Docume...Python Code/C
File Edit Format Run Options Windows Help
# Set the start value of A to 10
a=10
# Print the current value of A
print("The value of A is,", a)
a=a-1
print("You've just lost a life!")
print("You now have", a, "lives left!")
```





## STEP 5

While we know that the variable A is lives, and that the player has just lost one, a casual viewer or someone checking the code may not know. Imagine for a moment that the code is twenty thousand lines long, instead of just our seven. You can see how handy comments are.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[[GCC 4.9.1]] on linux
Type "copyright", "credits()" or "license()" for more information.
>>> ----- RESTART -----
The value of A is, 10
>>> ----- RESTART -----
The value of A is, 10
>>> ----- RESTART -----
The value of A is, 10
You've just lost a life!
You now have 9 lives left!
>>>
```

## STEP 8

Inline comments are comments that follow a section of code. Take our examples from above, instead of inserting the code on a separate line, we could use:

```
a=10 # Set the start value of A to 10
print("The value of A is,", a) # Print the current value of A
a=a-1 # Player lost a life!
print("You've just lost a life!")
print("You now have", a, "lives left!") # Inform player, and display current value of A (lives)
```

```
Comments.py - /home/pi/Documents/Python Code/Comments.py (3.4.2)
File Edit Format Run Options Windows Help
a=10 # Set the start value of A to 10
print("The value of A is,", a) # Print the current value of A
a=a-1 # Player lost a life!
print("You've just lost a life!")
print("You now have", a, "lives left!") # Inform player, and display current value of A (lives)
```

## STEP 6

Essentially, the new code together with comments could look like:

```
# Set the start value of A to 10
a=10
# Print the current value of A
print("The value of A is,", a)
# Player lost a life!
a=a-1
# Inform player, and display current value of A (lives)
print("You've just lost a life!")
print("You now have", a, "lives left!")
```

```
*Comments.py - /home/pi/Docume... Python Code/Comments.py (3.4.2)*
File Edit Format Run Options Windows Help
# Set the start value of A to 10
a=10
# Print the current value of A
print("The value of A is,", a)
# Player lost a life!
a=a-1
# Inform player, and display current value of A (lives)
print("You've just lost a life!")
print("You now have", a, "lives left!")
```

## STEP 9

The comment, the hash symbol, can also be used to comment out sections of code you don't want to be executed in your program. For instance, if you wanted to remove the first print statement, you would use:

```
# print("The value of A is,", a)
```

```
*Comments.py - /home/pi/Documents/Pytho...
File Edit Format Run Options Windows Help
# Set the start value of A to 10
a=10
# Print the current value of A
# print("The value of A is,", a)
# Player lost a life!
a=a-1
# Inform player, and display current value of A (lives)
print("You've just lost a life!")
print("You now have", a, "lives left!")
```

## STEP 7

You can use comments in different ways. For example, Block Comments are a large section of text that details what's going on in the code, such as telling the code reader what variables you're planning on using:

```
# This is the best game ever, and has been developed by a crack squad of Python experts
# who haven't slept or washed in weeks. Despite being very smelly, the code at least works really well.
# works really well.
```

```
*Comments.py - /home/pi/Documents/Python Code/Comments.py (3.4.2)*
File Edit Format Run Options Windows Help
# This is the best game ever, and has been developed by a crack squad of Python experts
# who haven't slept or washed in weeks. Despite being very smelly, the code at least
# works really well.
# Set the start value of A to 10
a=10
# Print the current value of A
print("The value of A is,", a)
# Player lost a life!
a=a-1
# Inform player, and display current value of A (lives)
print("You've just lost a life!")
print("You now have", a, "lives left!")
```

## STEP 10

You also use three single quotes to comment out a Block Comment or multi-line section of comments. Place them before and after the areas you want to comment for them to work:

```
'''
This is the best game ever, and has been developed by a crack squad of Python experts who haven't slept or washed in weeks. Despite being very smelly, the code at least works really well.
'''
```

```
*Comments.py - /home/pi/Documents/Python Code/Comments.py (3.4.2)*
File Edit Format Run Options Windows Help
'''
This is the best game ever, and has been developed by a crack squad of Python experts
who haven't slept or washed in weeks. Despite being very smelly, the code at least
works really well!
'''
# Set the start value of A to 10
a=10
# Print the current value of A
print("The value of A is,", a)
# Player lost a life!
a=a-1
# Inform player, and display current value of A (lives)
print("You've just lost a life!")
print("You now have", a, "lives left!")
```

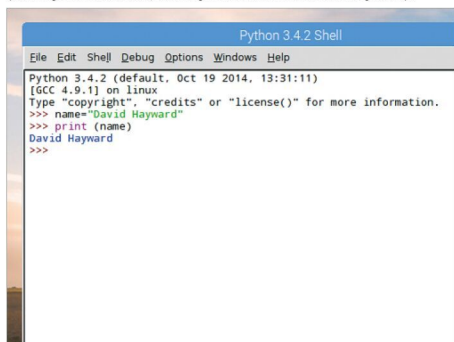
# Working with Variables

Variables are where all the action takes place in your code. A Variable can be anything, from the number of lives in a game, that changes as you progress through the game, to a person's name, age and address. Variables can be static or alter depending on what you want the code to do.

## VARIOUS VARIABLES

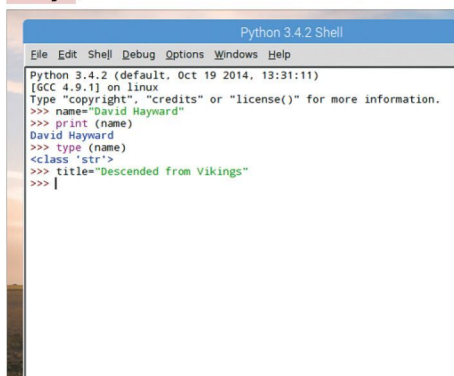
You'll be working with the Python 3 IDLE Shell in this tutorial. If you haven't already, open Python 3 or close down the previous IDLE Shell to clear up any old code.

**STEP 1** In some programming languages you're required to use a dollar sign to denote a string, which is a variable made up of multiple characters, such as a name of a person. In Python this isn't necessary. For example, in the Shell enter: `name="David Hayward"` (or use your own name, unless you're also called David Hayward).



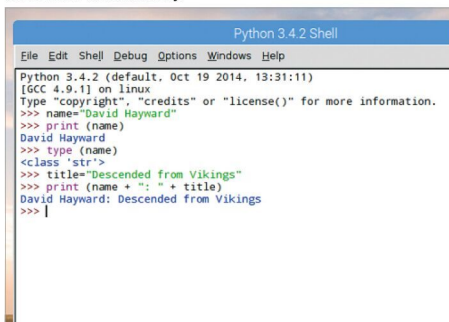
```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> name="David Hayward"
>>> print(name)
David Hayward
>>>
```

**STEP 2** You can check the type of variable in use by issuing the **type()** command, placing the name of the variable inside the brackets. In our example, this would be: `type(name)`. Add a new string variable: `title="Descended from Vikings"`.



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> name="David Hayward"
>>> print(name)
David Hayward
>>> type(name)
<class 'str'>
>>> title="Descended from Vikings"
>>> |
```

**STEP 3** You've seen previously that variables can be concatenated using the plus symbol between the variable names. In our example we can use: `print(name + " " + title)`. The middle part between the quotations allows us to add a colon and a space, as variables are connected without spaces, so we need to add them manually.



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> name="David Hayward"
>>> print(name)
David Hayward
>>> type(name)
<class 'str'>
>>> title="Descended from Vikings"
>>> print(name + " " + title)
David Hayward: Descended from Vikings
>>> |
```

**STEP 4** You can also combine variables within another variable. For example, to combine both name and title variables into a new variable we use:

`character=name + " " + title`  
Then output the content of the new variable as:

`print(character)`  
Numbers are stored as different variables:  
`age=44`  
**Type (age)**  
Which are integers, as we know.



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> name="David Hayward"
>>> print(name)
David Hayward
>>> type(name)
<class 'str'>
>>> title="Descended from Vikings"
>>> print(name + " " + title)
David Hayward: Descended from Vikings
>>> character=name + " " + title
>>> print(character)
David Hayward: Descended from Vikings
>>> age=44
>>> type(age)
```

**STEP 5**

However, you can't combine both strings and integer type variables in the same command, as you would a set of similar variables. You need to either turn one into the other or vice versa. When you do try to combine both, you get an error message: `print (name + age)`

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> name="David Hayward"
>>> print (name)
David Hayward
>>> type (name)
<class 'str'>
>>> title="Descended from Vikings"
>>> print (name + " " + title)
David Hayward: Descended from Vikings
>>> character=name + " " + title
>>> print (character)
David Hayward: Descended from Vikings
>>> age=44
>>> type (age)
<class 'int'>
>>> print (name+age)
Traceback (most recent call last):
  File "<psyshell#>", line 1, in <module>
    print (name+age)
TypeError: Can't convert 'int' object to str implicitly
```

**STEP 6**

This is a process known as TypeCasting. The Python code is:

```
print (character + " is " + str(age) + " years old.")
```

or you can use:

```
print (character, "is", age, "years old.")
```

Notice again that in the last example, you don't need the spaces between the words in quotes as the commas treat each argument to print separately.

```
>>> print (name + age)
Traceback (most recent call last):
  File "<psyshell#18>", line 1, in <module>
    print (name + age)
TypeError: Can't convert 'int' object to str implicitly
>>> print (character + " is " + str(age) + " years old.")
David Hayward: Descended from Vikings is 44 years old.
>>> print (character, "is", age, "years old.")
David Hayward: Descended from Vikings is 44 years old.
>>> |
```

**STEP 7**

Another example of TypeCasting is when you ask for input from the user, such as a name, for example, enter:

```
age= input ("How old are you? ")
```

All data stored from the Input command is stored as a string variable.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> age= input ("How old are you? ")
How old are you? 44
>>> type(age)
<class 'str'>
>>> |
```

**STEP 8**

This presents a bit of a problem when you want to work with a number that's been inputted by the user, as `age + 10` won't work due to being a string variable and an integer. Instead, you need to enter:

```
int (age) + 10
```

This will TypeCast the age string into an integer that can be worked with.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> age= input ("How old are you? ")
How old are you? 44
>>> type(age)
<class 'str'>
>>> age + 10
Traceback (most recent call last):
  File "<psyshell#2>", line 1, in <module>
    age + 10
TypeError: Can't convert 'int' object to str implicitly
>>> int(age) + 10
54
>>> |
```

**STEP 9**

The use of TypeCasting is also important when dealing with floating point arithmetic; remember: numbers that have a decimal point in them. For example, enter:

```
shirt=19.99
```

Now enter: `type (shirt)` and you'll see that Python has allocated the number as a 'float', because the value contains a decimal point.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> shirt=19.99
>>> type(shirt)
<class 'float'>
>>> |
```

**STEP 10**

When combining integers and floats Python usually converts the integer to a float, but should the reverse ever be applied it's worth remembering that Python doesn't return the exact value. When converting a float to an integer, Python will always round down to the nearest integer, called truncating; in our case instead of 19.99 it becomes 19.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> shirt=19.99
>>> type(shirt)
<class 'float'>
>>> int(shirt)
19
>>> |
```

# User Input

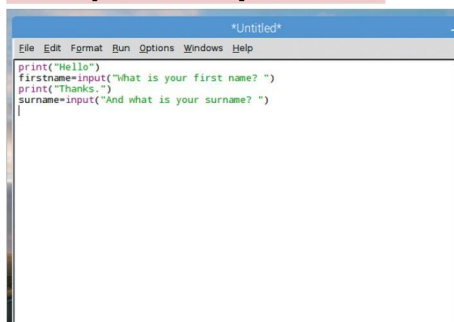
We've seen some basic user interaction with the code from a few of the examples earlier, so now would be a good time to focus solely on how you would get information from the user then store and present it.

## USER FRIENDLY

The type of input you want from the user will depend greatly on the type of program you're coding. For example, a game may ask for a character's name, whereas a database can ask for personal details.

**STEP 1** If it's not already, open the Python 3 IDLE Shell, and start a New File in the Editor. Let's begin with something really simple, enter:

```
print("Hello")
firstname=input("What is your first name? ")
print("Thanks.")
surname=input("And what is your surname? ")
```



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> ----- RESTART -----
>>>
>>> Hello
>>> What is your first name? David
>>> Thanks.
>>> And what is your surname? Hayward
>>> |
```

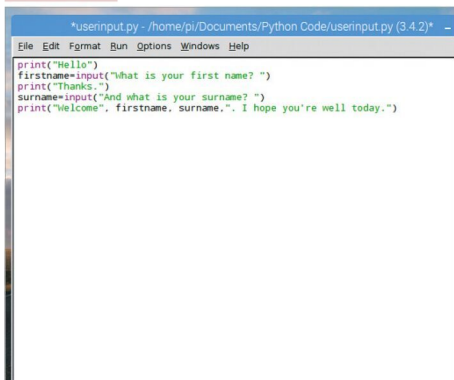
**STEP 2** Save and execute the code, and as you already no doubt suspected, in the IDLE Shell the program will ask for your first name, storing it as the variable `firstname`, followed by your surname; also stored in its own variable (`surname`).



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> ----- RESTART -----
>>>
>>> Hello
>>> What is your first name? David
>>> Thanks.
>>> And what is your surname? Hayward
>>> |
```

**STEP 3** Now that we have the user's name stored in a couple of variables we can call them up whenever we want:

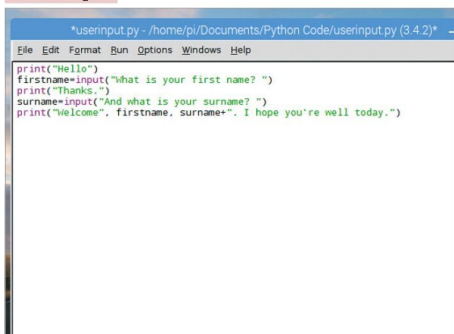
```
print("Welcome", firstname, surname, ". I hope you're well today.")
```



```
*userinput.py - /home/pi/Documents/Python Code/userinput.py (3.4.2)*
File Edit Fgmat Bun Options Windows Help
print("Hello")
firstname=input("What is your first name? ")
print("Thanks.")
surname=input("And what is your surname? ")
print("Welcome", firstname, surname, ". I hope you're well today.")
```

**STEP 4** Run the code and you can see a slight issue, the full stop after the surname follows a blank space. To eliminate that we can add a plus sign instead of the comma in the code:

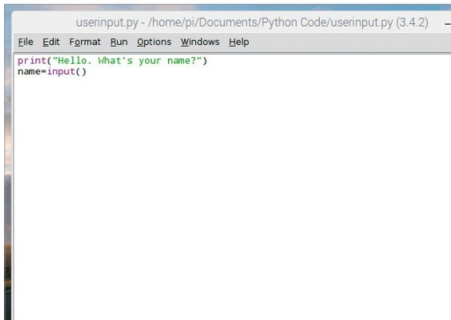
```
print("Welcome", firstname, surname+". I hope you're well today.")
```



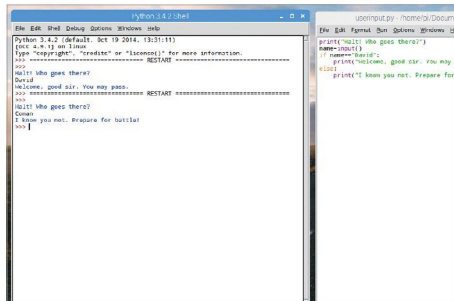
```
*userinput.py - /home/pi/Documents/Python Code/userinput.py (3.4.2)*
File Edit Fgmat Bun Options Windows Help
print("Hello")
firstname=input("What is your first name? ")
print("Thanks.")
surname=input("And what is your surname? ")
print("Welcome", firstname, surname+". I hope you're well today.")
```

**STEP 5** You don't always have to include quoted text within the input command. For example, you can ask the user their name, and have the input in the line below:

```
print("Hello. What's your name?")
name=input()
```

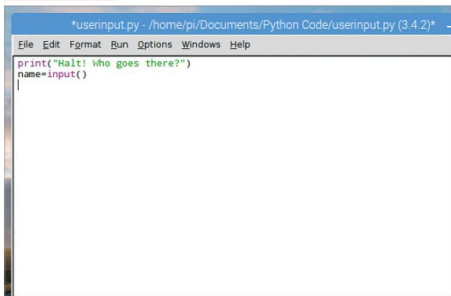


**STEP 8** What you've created here is a condition, which we will cover soon. In short, we're using the input from the user and measuring it against a condition. So, if the user enters David as their name, the guard will allow them to pass unhindered. Else, if they enter a name other than David, the guard challenges them to a fight.



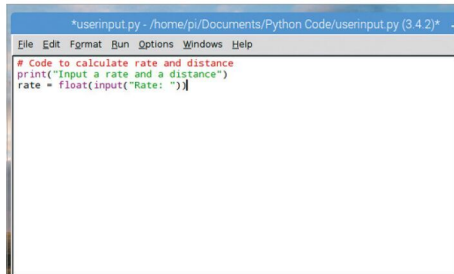
**STEP 6** The code from the previous step is often regarded as being a little neater than having a lengthy amount of text in the input command, but it's not a rule that's set in stone, so do as you like in these situations. Expanding on the code, try this:

```
print("Halt! Who goes there?")
name=input()
```



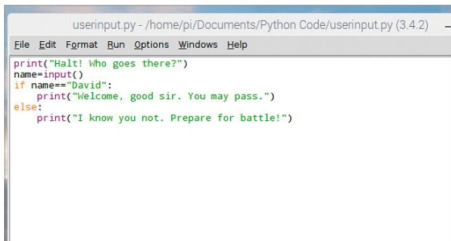
**STEP 9** Just as you learned previously, any input from a user is automatically a string, so you need to apply a TypeCast in order to turn it into something else. This creates some interesting additions to the input command. For example:

```
# Code to calculate rate and distance
print("Input a rate and a distance")
rate = float(input("Rate: "))
```



**STEP 7** It's a good start to a text adventure game, perhaps? Now you can expand on it and use the raw input from the user to flesh out the game a little:

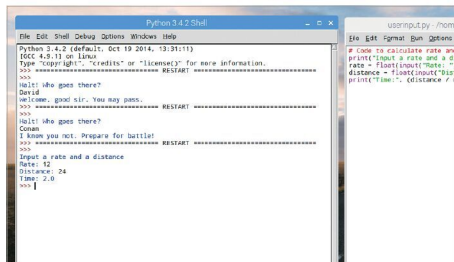
```
if name=="David":
    print("Welcome, good sir. You may pass.")
else:
    print("I know you not. Prepare for battle!")
```



**STEP 10** To finalise the rate and distance code, we can add: `distance = float(input("Distance: "))`

`print("Time:", (distance / rate))`

Save and execute the code and enter some numbers. Using the float(input element, we've told Python that anything entered is a floating point number rather than a string.



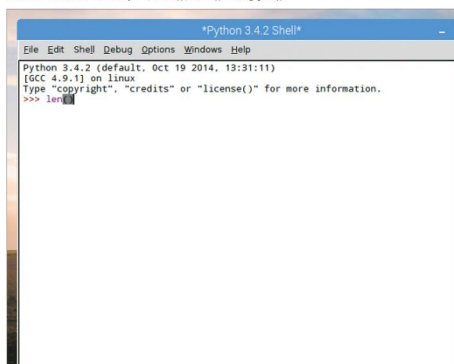
# Creating Functions

Now that you've mastered the use of variables and user input, the next step is to tackle functions. You've already used a few functions, such as the print command but Python enables you to define your own functions.

## FUNKY FUNCTIONS

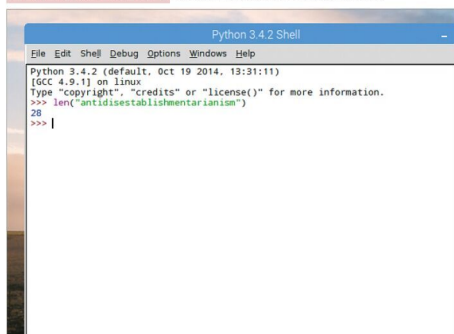
A function is a command that you enter into Python to do something. It's a little piece of self-contained code that takes data, works on it and then returns the result.

**STEP 1** It's not just data that a function works on. They can do all manner of useful things in Python, such as sort data, change items from one format to another and check the length or type of items. Basically, a function is a short word that's followed by brackets. For example, **len()**, **list()** or **type()**.



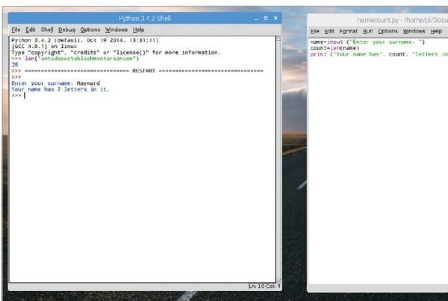
```
Python 3.4.2 Shell
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> len("antidisestablishmentarianism")
28
```

**STEP 2** A function takes data, usually a variable, works on it depending on what the function is programmed to do and returns the end value. The data being worked on goes inside the brackets, so if you wanted to know how many letters are in the word **antidisestablishmentarianism**, then you'd enter: **len("antidisestablishmentarianism")** and the number 28 would return.



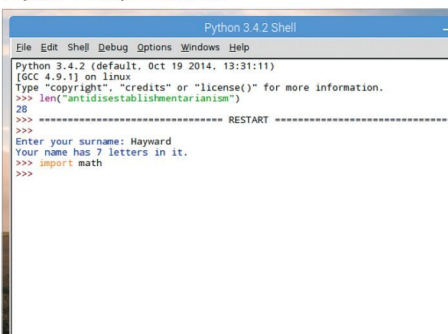
```
Python 3.4.2 Shell
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> len("antidisestablishmentarianism")
28
>>> |
```

**STEP 3** You can pass variables through functions in much the same manner. Let's assume you want the number of letters in a person's surname, you could use the following code (enter the text editor for this example):  
**name=input("Enter your surname: ")**  
**count=len(name)**  
**print("Your surname has", count, "letters in it.")**  
 Press F5 and save the code to execute it.



```
Python 3.4.2 Shell
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> name=input("Enter your surname: ")
count=len(name)
print("Your name has", count, "letters in it.")
>>>
Enter your surname: Hayward
Your name has 7 letters in it.
```

**STEP 4** Python has tens of functions built into it, far too many to get into in the limited space available here. However, to view the list of built-in functions available to Python 3, navigate to [www.docs.python.org/3/library/functions.html](http://www.docs.python.org/3/library/functions.html). These are the predefined functions, but since users have created many more, they're not the only ones available.



```
Python 3.4.2 Shell
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> len("antidisestablishmentarianism")
28
>>> ----- RESTART -----
>>>
Enter your surname: Hayward
Your name has 7 letters in it.
>>> import math
>>>
```

**STEP 5** Additional functions can be added to Python through modules. Python has a vast range of modules available that can cover numerous programming duties. They add functions and can be imported as and when required. For example, to use advanced mathematics functions enter: `import math`. Once entered, you have access to all the Math module functions.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on Linux
Type "copyright", "credits" or "license()" for more information.
>>> len("antidisestablishmentarianism")
28
>>>
-----RESTART-----
>>>
Enter your surname: Hayward
Your name has 7 letters in it.
>>> import math
>>>
```

**STEP 6** To use a function from a module enter the name of the module followed by a full stop, then the name of the function. For instance, using the Math module, since you've just imported it into Python, you can utilise the square root function. To do so, enter: `math.sqrt(16)`. You can see that the code is presented as module.function(data).

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on Linux
Type "copyright", "credits" or "license()" for more information.
>>> len("antidisestablishmentarianism")
28
>>>
-----RESTART-----
>>>
Enter your surname: Hayward
Your name has 7 letters in it.
>>> import math
>>> math.sqrt(16)
4.0
>>> |
```

## FORGING FUNCTIONS

There are many different functions you can import, created by other Python programmers, and you will undoubtedly come across some excellent examples in the future; you can also create your own with the `def` command.

**STEP 1** Choose File > New File to enter the editor, let's create a function called Hello, that greets a user. Enter: `def Hello () :`  
`print ("Hello")`  
`Hello ()`  
 Press F5 to save and run the script. You can see Hello in the Shell, type in Hello() and it returns the new function.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on Linux
Type "copyright", "credits" or "license()" for more information.
>>>
-----RESTART-----
>>>
Enter your surname: Hayward
Your name has 7 letters in it.
>>> import math
>>>
-----RESTART-----
>>>
Enter your surname: Hayward
Your name has 7 letters in it.
>>> import math
>>> math.sqrt(16)
4.0
>>> |
```

**STEP 3** To modify it further, delete the Hello("David") line, the last line in the script and press Ctrl+S to save the new script. Close the Editor and create a new file (File > New File). Enter the following:  
`from Hello import Hello`  
`Hello ("David")`  
 Press F5 to save and execute the code.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on Linux
Type "copyright", "credits" or "license()" for more information.
>>>
-----RESTART-----
>>>
Enter your surname: Hayward
Your name has 7 letters in it.
>>> import math
>>> math.sqrt(16)
4.0
>>> |
```

**STEP 2** Let's now expand the function to accept a variable, the user's name for example. Edit your script to read:  
`def Hello (name) :`  
`print ("Hello", name)`  
`Hello ("David")`  
 This will now accept the variable name, otherwise it prints Hello David. In the Shell, enter: `name="Bob"` , then : `Hello (name)`. Your function can now pass variables through it.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on Linux
Type "copyright", "credits" or "license()" for more information.
>>>
-----RESTART-----
>>>
Enter your surname: Hayward
Your name has 7 letters in it.
>>> import math
>>> math.sqrt(16)
4.0
>>> |
```

**STEP 4** What you've just done is import the Hello function from the saved Hello.py program and then used it to say hello to David. This is how modules and functions work: you import the module then use the function. Try this one, and modify it for extra credit:  
`def add (a, b) :`  
`result = a + b`  
`return result`

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on Linux
Type "copyright", "credits" or "license()" for more information.
>>>
-----RESTART-----
>>>
Enter your surname: Hayward
Your name has 7 letters in it.
>>> import math
>>> math.sqrt(16)
4.0
>>> |
```

# Conditions and Loops

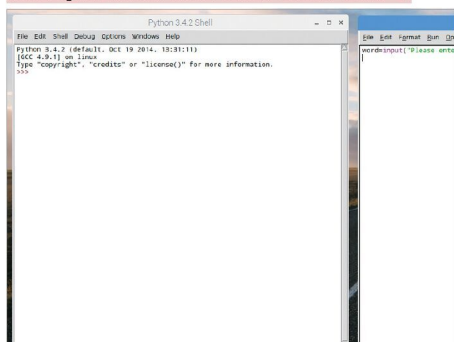
Conditions and loops are what makes a program interesting; they can be simple or rather complex. How you use them depends greatly on what the program is trying to achieve; they could be the number of lives left in a game or just displaying a countdown.

## TRUE CONDITIONS

Keeping conditions simple to begin with makes learning to program a more enjoyable experience. Let's start then by checking if something is TRUE, then doing something else if it isn't.

**STEP 1** Let's create a new Python program that will ask the user to input a word, then check it to see if it's a four-letter word or not. Start with File > New File, and begin with the input variable:

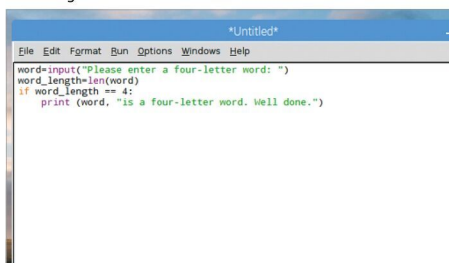
```
word=input("Please enter a four-letter word: ")
```



**STEP 3** Now you can use an if statement to check if the word\_length variable is equal to four and print a friendly confirmation if it applies to the rule:

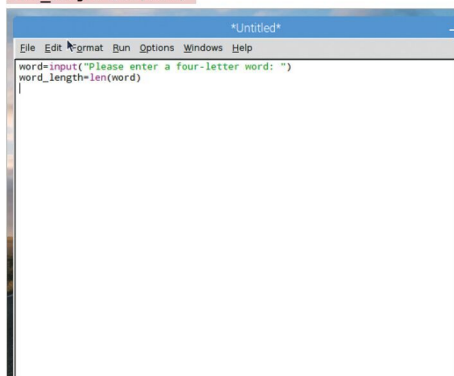
```
word=input("Please enter a four-letter word: ")
word_length=len(word)
if word_length == 4:
    print (word, "is a four-letter word. Well done.")
```

The double equal sign (==) means check if something is equal to something else.



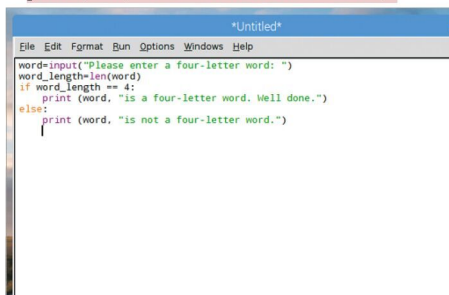
**STEP 2** Now we can create a new variable, then use the len function and pass the word variable through it to get the total number of letters the user has just entered:

```
word=input("Please enter a four-letter word: ")
word_length=len(word)
```



**STEP 4** The colon at the end of if tells Python that if this statement is true do everything after the colon that's indented. Next, move the cursor back to the beginning of the Editor:

```
word=input("Please enter a four-letter word: ")
word_length=len(word)
if word_length == 4:
    print (word, "is a four-letter word. Well done.")
else:
    print (word, "is not a four-letter word.")
```







## STEP 5

Press F5 and save the code to execute it. Enter a four-letter word in the Shell to begin with, you should have the returned message that it's the word is four letters. Now press F5 again and rerun the program but this time enter a five-letter word. The Shell will display that it's not a four-letter word.

```
Python 3.4.2 Shell
Python 3.4.2 (default, Oct 19 2014, 18:31:11)
[GC 4.2.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> word=input("Please enter a four-letter word: ")
word_length=len(word)
if word_length==4:
    print(word, "is a four-letter word.")
else:
    print(word, "is not a four-letter word.")
>>>
Please enter a four-letter word: Well
Well
Please enter a four-letter word: Well done
Well done
Please enter a four-letter word: Frost
Frost is not a four-letter word.
>>>
```

## STEP 6

Now expand the code to include another condition. Eventually, it could become quite complex. We've added a condition for three-letter words:

```
word=input("Please enter a four-letter word: ")
word_length=len(word)
if word_length == 4:
    print(word, "is a four-letter word. Well done.")
elif word_length == 3:
    print(word, "is a three-letter word. Try again.")
else:
    print(word, "is not a four-letter word.")
```

```
Python 3.4.2 Shell
Python 3.4.2 (default, Oct 19 2014, 18:31:11)
[GC 4.2.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> word=input("Please enter a four-letter word: ")
word_length=len(word)
if word_length==4:
    print(word, "is a four-letter word. Well done.")
elif word_length==3:
    print(word, "is a three-letter word. Try again.")
else:
    print(word, "is not a four-letter word.")
>>>
Please enter a four-letter word: Well
Well
Please enter a four-letter word: Well done
Well done
Please enter a four-letter word: Frost
Frost is not a four-letter word.
Please enter a four-letter word: Egg
Egg is a three-letter word. Try again.
>>>
```

## LOOPS

A loop works quite similar to a condition but they are somewhat different in their operation. A loop will run through the same block of code a number of times, usually with the support of a condition.

## STEP 1

Let's start with a simple While statement. Like IF, this will check to see if something is TRUE, then run the indented code:

```
x = 1
while x < 10:
    print (x)
    x = x + 1
```

```
*Untitled*
File Edit Format Run Options Windows Help
x=1
while x<10:
    print(x)
    x=x+1
1
2
3
4
5
6
7
8
9
10
RESTART
>>>
Cat
Dog
Unicorn
>>>
```

## STEP 3

The For loop is another example. For is used to loop over a range of data, usually a list stored as variables inside square brackets. For example:

```
words=["Cat", "Dog", "Unicorn"]
for word in words:
    print (word)
```

```
Python 3.4.2 Shell
Python 3.4.2 (default, Oct 19 2014, 18:31:11)
[GC 4.2.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> words=["Cat", "Dog", "Unicorn"]
for word in words:
    print (word)
>>>
Cat
Dog
Unicorn
>>>
```

## STEP 2

The difference between if and while is when while gets to the end of the indented code, it goes back and checks the statement is still true. In our example x is less than 10. With each loop it prints the current value of x, then adds one to that value. When x does eventually equal 10 it stops.

```
Python 3.4.2 Shell
Python 3.4.2 (default, Oct 19 2014, 18:31:11)
[GC 4.2.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> x=1
while x<10:
    print(x)
    x=x+1
1
2
3
4
5
6
7
8
9
10
>>>
```

## STEP 4

The For loop can also be used in the countdown example by using the range function:

```
for x in range (1, 10):
    print (x)
```

The x=x+1 part isn't needed here because the range function creates a list between the first and last numbers used.

```
Python 3.4.2 Shell
Python 3.4.2 (default, Oct 19 2014, 18:31:11)
[GC 4.2.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> for x in range (1, 10):
    print (x)
1
2
3
4
5
6
7
8
9
>>>
```

# Python Modules

We've mentioned modules previously, (the Math module) but as modules are such a large part of getting the most from Python, it's worth dedicating a little more time to them. In this instance we're using the Windows version of Python 3.

## MASTERING MODULES

Think of modules as an extension that's imported into your Python code to enhance and extend its capabilities. There are countless modules available and as we've seen, you can even make your own.

**STEP 1** Although good, the built-in functions within Python are limited. The use of modules, however, allows us to make more sophisticated programs. As you are aware, modules are Python scripts that are imported, such as `import math`.

```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help
Python 3.6.2 (tags:3.6.2:5fd33b5, Jul 8 2017, 04:14:34) [MSC v.1900 32 bit (Intel) on win32]
Type "copyright", "credits" or "license()" for more information.
>>> import math
>>>
```

**STEP 2** Some modules, especially on the Raspberry Pi, are included by default, the Math module being a prime example. Sadly, other modules aren't always available. A good example on non-Pi platforms is the Pygame module, which contains many functions to help create games. Try: `import pygame`.

```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help
Python 3.6.2 (tags:3.6.2:5fd33b5, Jul 8 2017, 04:14:34) [MSC v.1900 32 bit (Intel) on win32]
Type "copyright", "credits" or "license()" for more information.
>>> import math
>>> import pygame
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    import pygame
ModuleNotFoundError: No module named 'pygame'
>>>
```

**STEP 3** The result is an error in the IDLE Shell, as the Pygame module isn't recognised or installed in Python. To install a module we can use PIP (Pip Installs Packages). Close down the IDLE Shell and drop into a command prompt or Terminal session. At an elevated admin command prompt, enter:

`pip install pygame`

```
Command Prompt
C:\Users\dauid>pip install pygame
```

**STEP 4** The PIP installation requires an elevated status due to installing components at different locations. Windows users can search for CMD via the Start button and right-click the result then click Run as Administrator. Linux and Mac users can use the Sudo command, with `sudo pip install package`.

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>pip install pygame
Collecting pygame
  Using cached pygame-1.9.3-cp36-cp36m-win32.whl
Installing collected packages: pygame
Successfully installed pygame-1.9.3

C:\WINDOWS\system32>
```

**STEP 5**

Close the command prompt or Terminal and relaunch the IDLE Shell. When you now enter: `import pygame`, the module will be imported into the code without any problems. You'll find that most code downloaded or copied from the internet will contain a module, mainstream or unique, these are usually the source of errors in execution due to them being missing.

```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help
Python 3.6.2 (v3.6.2:15f033b5, Jul 8 2017, 04:14:34) [MSC v.1900 32 bit (IA
on win32
Type "copyright", "credits" or "license()" for more information.
>>> import pygame
>>>
```

**STEP 6**

The modules contain the extra code needed to achieve a certain result within your own code, as we've previously experimented with. For example:

`import random`

Brings in the code from the random number generator module. You can then use this module to create something like:

```
for i in range(10):
    print(random.randint(1, 25))
```

```
"Untitled"
File Edit Format Run Options Window Help
import random

for i in range(10):
    print(random.randint(1, 25))
```

**STEP 7**

This code, when saved and executed, will display ten random numbers from 1 to 25. You can play around with the code to display more or less, and from a great or lesser range. For example:

`import random`

```
for i in range(25):
    print(random.randint(1, 100))
```

```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help
Python 3.6.2 (v3.6.2:15f033b5, Jul 8 2017, 04:14:34) [MSC v.1900 32 bit (IA
on win32
Type "copyright", "credits" or "license()" for more information.
>>> import random
>>> for i in range(25):
>>>     print(random.randint(1, 100))
>>>
10
14
11
40
37
67
64
58
62
49
66
```

**STEP 8**

Multiple modules can be imported within your code. To extend our example, use:

```
import random
import math
for i in range(5):
    print(random.randint(1, 25))
print(math.pi)
```

```
Rnd Number.py - C:/Users/david/Documents/Python/Rnd Number.py (
File Edit Format Run Options Window Help

import random
import math

for i in range(5):
    print(random.randint(1, 25))

print(math.pi)
```

**STEP 9**

The result is a string of random numbers followed by the value of Pi as pulled from the Math module using the `print(math.pi)` function. You can also pull in certain functions from a module by using the `from` and `import` commands, such as:

```
from random import randint
for i in range(5):
    print(randint(1, 25))
```

```
Rnd Number.py - C:/Users/david/Documents/Python/Rnd Number.py (
File Edit Format Run Options Window Help

from random import randint

for i in range(5):
    print(randint(1, 25))
```

**STEP 10**

This helps create a more streamlined approach to programming. You can also use `import module*`, which will import everything defined within the named module. However, it's often regarded as a waste of resources but it works nonetheless. Finally, modules can be imported as aliases:

```
import math as m
print(m.pi)
```

Of course, adding comments helps to tell others what's going on.

```
*Rnd Number.py - C:/Users/david/Documents/Python/Rnd Number.py
File Edit Format Run Options Window Help

import math as m

print(m.pi)
```

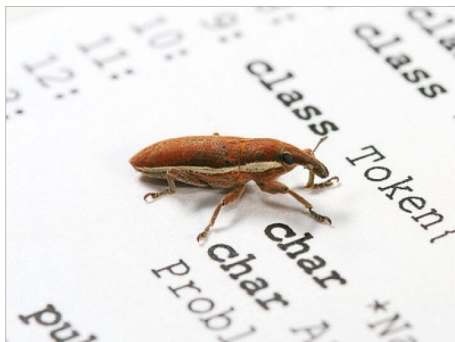
# Python Errors

It goes without saying that you'll eventually come across an error in your code, where Python declares it's not able to continue due to something being missed out, wrong or simply unknown. Being able to identify these errors makes for a good programmer.

## DEBUGGING

Errors in code are called bugs and are perfectly normal. They can often be easily rectified with a little patience. The important thing is to keep looking, experimenting and testing. Eventually your code will be bug free.

**STEP 1** Code isn't as fluid as the written word, no matter how good the programming language is. Python is certainly easier than most languages but even it is prone to some annoying bugs. The most common are typos by the user and whilst easy to find in simple dozen-line code, imagine having to debug multi-thousand line code.



**STEP 2** The most common of errors is the typo, as we've mentioned. The typos are often at the command level: mistyping the print command for example. However, they also occur when you have numerous variables, all of which have lengthy names. The best advice is to simply go through the code and check your spelling.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information
>>> apples=10
>>> pirnt(apples)
Traceback (most recent call last):
  File "<pyshe11#1>", line 1, in <module>
    pirnt(apples)
NameError: name 'pirnt' is not defined
>>> |
```

**STEP 3** Thankfully Python is helpful when it comes to displaying error messages. When you receive an error, in red text from the IDLE Shell, it will define the error itself along with the line number where the error has occurred. Whilst in the IDLE Editor this is a little daunting for lots of code; text editors help by including line numbering.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
Traceback (most recent call last):
  File "Name:/documents/graphics.py", line 83, in module
    pygame.draw.ellipse(windowSurface, RED, (300, 250, 40, 80), 1)
NameError: name 'pygame' is not defined
>>>
```

**STEP 4** Syntax errors are probably the second most common errors you'll come across as a programmer. Even if the spelling is correct, the actual command itself is wrong. In Python 3 this often occurs when Python 2 syntaxes are applied. The most annoying of these is the print function. In Python 3 we use `print("words")`, whereas Python2 uses `print "words"`.

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information
>>> print"Hello world!"
SyntaxError: invalid syntax
>>>
```



### STEP 5

Pecky brackets are also a nuisance in programming errors, especially when you have something like:

```
print(balanced_check(input()))
```

Remember that for every '(' there must be an equal number of ')'.  
 1 import sys  
 2 def balanced\_check(data):  
 3 stack = []  
 4 characters = list(data)  
 5  
 6 for character in characters:  
 7 reference = {  
 8 '(': ')',  
 9 '[': ']',  
 10 '{': '}',  
 11 ' ': ' ',  
 12 }  
 13 if character in reference.keys():  
 14 stack.append(character)  
 15  
 16 elif character in reference.values() and len(stack) > 0:  
 17 char = stack.pop()  
 18 if reference.get(char) != character:  
 19 return "NO"  
 20 else:  
 21 return "NO"  
 22  
 23 if len(stack) == 0:  
 24 return "YES"

### STEP 6

There are thousands of online Python resources, code snippets and lengthy discussions across forums on how best to achieve something. Whilst 99 per cent of it is good code, don't always be lured into copying and pasting random code into your editor. More often than not, it won't work and the worst part is that you haven't learnt anything.

You have a bare except clause, i.e.,

```
8 try:  
    some_code()  
except:  
    clean_up()
```

The problem with a bare except is that it will catch all exceptions, including ones you really don't want to be ignoring (like KeyboardInterrupt and SystemExit). It would be much better your except block only caught the specific exception you expect, and let all others bubble as normal.

A few other general comments on your code:

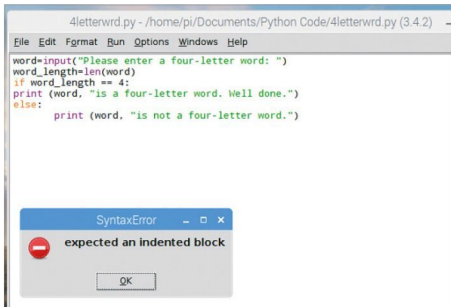
- In line 200, you have this construction:

```
for letter in range(len(chosen_word):  
    if player_guess == chosen_word[letter]:  
        word_guessed[letter] = player_guess
```

You're looping over the index variable, but also using the list element. It would be better to loop over the characters directly.

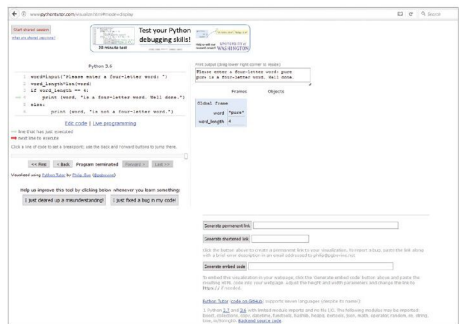
### STEP 7

Indents are a nasty part of Python programming that a lot of beginners fall foul of. Recall the If loop from the Conditions and Loops section, where the colon means everything indented following the statement is to be executed as long as it's true? Missing the indent, or having too much of indent, will come back with an error.



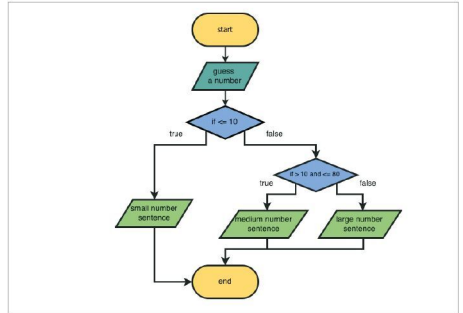
### STEP 8

An excellent way to check your code step-by-step is to use Python Tutor's Visualise web page, found at [www.pythontutor.com/visualize.html#mode=edit](http://www.pythontutor.com/visualize.html#mode=edit). Simply paste your code into the editor and click the Visualise Execution button to run the code line-by-line. This helps to clear bugs and any misunderstandings.



### STEP 9

Planning makes for good code. Whilst a little old school, it's a good habit to plan what your code will do before sitting down to type it out. List the variables that will be used and the modules too; then write out a script for any user interaction or outputs.



### STEP 10

Purely out of interest, the word debugging in computing terms comes from Admiral Grace Hopper, who back in the '40s was working on a monolithic Harvard Mark II electromechanical computer. According to legend Hopper found a moth stuck in a relay, thus stopping the system from working. Removal of the moth was hence called debugging.



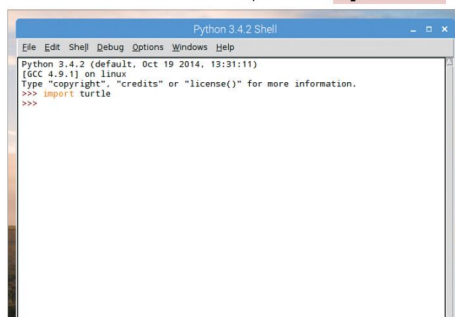
# Python Graphics

While dealing with text on the screen, either as a game or in a program, is great, there will come a time when a bit of graphical representation wouldn't go amiss. Python 3 has numerous ways in which to include graphics and they're surprisingly powerful too.

## GOING GRAPHICAL

You can draw simple graphics, lines, squares and so on, or you can use one of the many Python modules available, to bring out some spectacular effects.

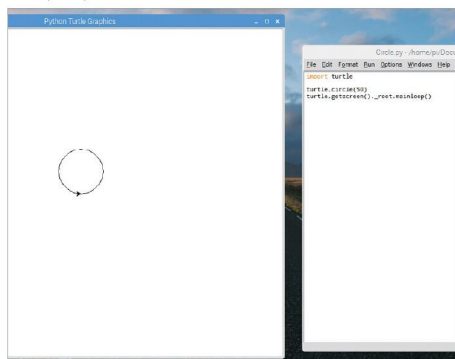
**STEP 1** One of the best graphical modules to begin learning Python graphics is Turtle. The Turtle module is, as the name suggests, based on the turtle robots used in many schools, that can be programmed to draw something on a large piece of paper on the floor. The Turtle module can be imported with: `import turtle`.



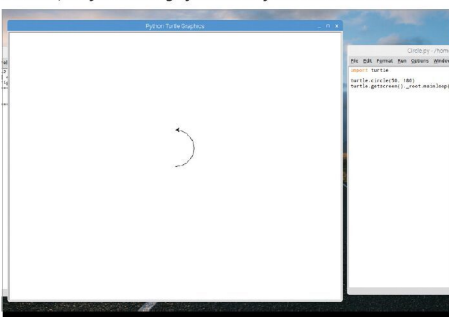
**STEP 2** Let's begin by drawing a simple circle. Start a New File, then enter the following code:

```
import turtle
turtle.circle(50)
turtle.getscreen().root.mainloop()
```

As usual press F5 to save the code and execute it. A new window will now open up and the 'Turtle' will draw a circle.



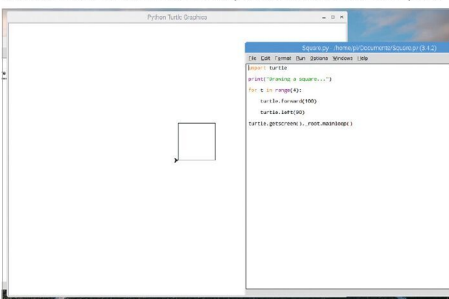
**STEP 3** The command `turtle.circle(50)` is what draws the circle on the screen, with 50 being the size. You can play around with the sizes if you like, going up to 100, 150 and beyond; you can draw an arc by entering: `turtle.circle(50, 180)`, where the size is 50, but you're telling Python to only draw 180° of the circle.



**STEP 4** The last part of the circle code tells Python to keep the window where the drawing is taking place to remain open, so the user can click to close it. Now, let's make a square:

```
import turtle
print("Drawing a square...")
for t in range(4):
    turtle.forward(100)
    turtle.left(90)
turtle.getscreen().root.mainloop()
```

You can see that we've inserted a loop to draw the sides of the square.



**STEP 5**

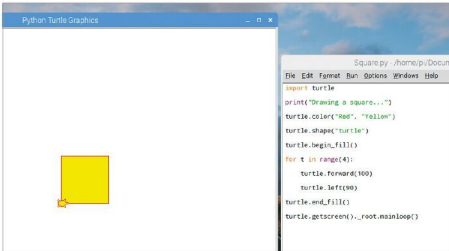
You can add a new line to the square code to add some colour:

```
turtle.color("Red")
```

Then you can even change the character to an actual turtle by entering:

```
turtle.shape("turtle")
```

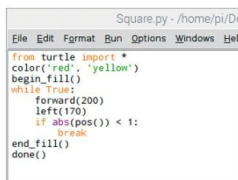
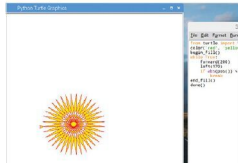
You can also use the command `turtle.begin_fill()`, and `turtle.end_fill()` to fill in the square with the chosen colours; red outline, yellow fill in this case.

**STEP 6**

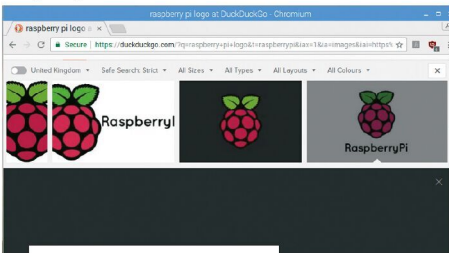
You can see that the Turtle module can draw out some pretty good shapes and become a little more complex as you begin to master the way it works. Enter this example:

```
from turtle import *  
color('red', 'yellow')  
begin_fill()  
while True:  
    forward(200)  
    left(170)  
    if abs(pos()) < 1:  
        break  
end_fill()  
done()
```

It's a different method, but very effective.

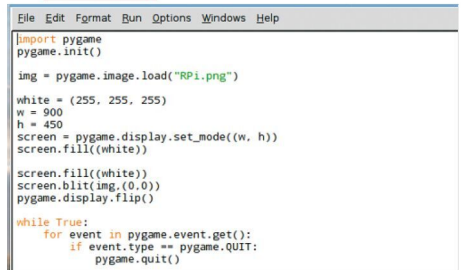
**STEP 7**

Another way in which you can display graphics is by using the Pygame module. There are numerous ways in which pygame can help you output graphics to the screen but for now let's look at displaying a predefined image. Start by opening a browser and finding an image, then save it to the folder where you save your Python code.

**STEP 8**

Now let's get the code by importing the Pygame module:

```
import pygame  
pygame.init()  
img = pygame.image.load("RPI.png")  
white = (255, 255, 255)  
w = 900  
h = 450  
screen = pygame.display.  
set_mode((w, h))  
screen.fill((white))  
screen.blit(img, (0,0))  
pygame.display.flip()  
while True:  
    for event in pygame.event.get():  
        if event.type == pygame.QUIT:  
            pygame.quit()
```

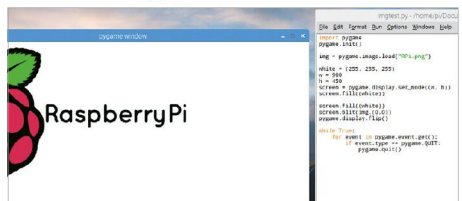
**STEP 9**

In the previous step you imported pygame, initiated the pygame engine and asked it to import our saved Raspberry Pi logo image, saved as RPI.png. Next you defined the background colour of the window to display the image and the window size as per the actual image dimensions. Finally you have a loop to close the window.

```
img = pygame.image.load("RPI.png")  
white = (255, 255, 255)  
w = 900  
h = 450  
screen = pygame.display.set_mode((w, h))  
screen.fill((white))  
screen.blit(img, (0,0))  
pygame.display.flip()  
while True:  
    for event in pygame.event.get():  
        if event.type == pygame.QUIT:  
            pygame.quit()
```

**STEP 10**

Press F5 to save and execute the code and your image will be displayed in a new window. Have a play around with the colours, sizes and so on and take time to look up the many functions within the Pygame module too.





# Glossary of Terms

Just like most technology, Python contains many confusing words and acronyms. Here then, for your own sanity, is a handy glossary to help you keep on top of what's being said when the conversation turns to Python programming.

## Argument

The detailed extra information used by Python to perform more detailed commands. Can also be used in the command prompt to specify a certain runtime event.

## Block

Used to describe a section or sections of code that are grouped together.

## Break

A command that can be used to exit a for or while loop. For example, if a key is pressed to quit the program, Break will exit the loop.

## Class

A class provides a means of bundling data and functionality together. They are used to encapsulate variables and functions into a single entity.

## Comments

A comment is a section of real world wording inserted by the programmer to help document what's going on in the code. They can be single line or multi-line and are defined by a # or ".

## Debian

A Linux-based distro or distribution that forms the Debian Project. This environment offers the user a friendly and stable GUI to interact with along with Terminal commands and other forms of system level administration.

## Def

Used to define a function or method in Python.

## Dictionaries

A dictionary in Python is a data structure that consists of key and value pairs.

## Distro

Also Distribution, an operating system that uses the Linux Kernel as its core but offers something different in its presentation to the end user.

## Editor

An individual program, or a part of the graphical version of Python, that enables the user to enter code ready for execution.

## Exceptions

Used as a means of breaking from the normal flow of a code block in order to handle any potential errors or exceptional

conditions within the program.

## Expression

Essentially, Python code that produces a value of something.

## Float

An immutable floating point number used in Python.

## Function

Used in Python to define a sequence of statements that can be called or referenced at any time by the programmer.

## GitHub

A web-based version control and collaboration portal designed for software developers to better manage source code.

## Global Variable

A variable that is useable anywhere in the program.

## Graphics

The use of visual interaction with a program, game or operating system. Designed to make it easier for the user to manage the program in question.

## GUI

Graphical User Interface. The interface which most modern operating systems use to enable the user to interact with the core programming of the system. A friendly, easy to use graphical desktop environment.

## High-Level Language

A programming language that's designed to be easy for people to read.

## IDLE

Stands for Integrated Development Environment or Integrated Development and Learning Environment.

## Immutable

Something that cannot be changed after it is created.

## Import

Used in Python to include modules together with all the accompanying code, functions and variables they contain.

## Indentation

Python uses indentation to delimit blocks of code. The indents are four spaces apart, and are often created automatically after a colon is used in the code.





## Integer

A number data type that must be a whole number and not a decimal.

## Interactive Shell

The Python Shell, which is displayed whenever you launch the graphical version of Python.

## Kernel

The core of an operating system, which handles data processing, memory allocation, input and output, and processes information between the hardware and programs.

## Linux

An open source operating system that's modelled on UNIX. Developed in 1991 by Finnish student Linus Torvalds.

## Lists

A Python data type that contains collections of values, which can be of any type and can readily be modified.

## Local Variable

A variable that's defined inside a function and is only useable inside that function.

## Loop

A piece of code that repeats itself until a certain condition is met. Loops can encase the entire code or just sections of it.

## Module

A Python file that contains various functions that can be used within another program to further extend the effectiveness of the code.

## Operating System

Also OS. The program that's loaded into the computer after the initial boot sequence has completed. The OS manages all the other programs, graphical user interface (GUI), input and output and physical hardware interactions with the user.

## Output

Data that is sent from the program to a screen, printer or other external peripheral.

## PIP

Pip Installs Packages. A package management system used to install and manage modules and other software written in Python.

## Print

A function used to display the output of something to the screen.

## Prompt

The element of Python, or the Command Line, where the user enters their commands. In Python it's represented as >>> in the interactive shell.

## Pygame

A Python module that's designed for writing games. It includes graphics and sound libraries and was first developed in October 2000.

## Python

An awesome programming language that's easy to learn and use, whilst still being powerful enough to enjoy.

## Random

A Python module that implements a pseudo-random character generator using the Mersenne Twister PRNG.

## Range

A function that's used to return a list of integers, defined by the arguments passed through it.

## Root

The bottom level user account used by the system itself. Root is the overall system administrator and can go anywhere, and do anything, on the system.

## Sets

Sets are a collection of unordered but unique data types.

## Strings

Strings can store characters that can be modified. The contents of a string are alphanumerical and can be enclosed by either single or double quote marks.

## Terminal

Also Console or Shell. The command line interface to the operating system, namely Linux, but also available in macOS. From there you can execute code and navigate the filesystem.

## Tkinter

A Python module designed to interact with the graphical environment, specifically the tk-GUI (Tool Kit Graphical User Interface).

## Try

A try block allows exceptions to be raised, so any errors can be caught and handled according to the programmer's instructions.

## Tuples

An immutable Python data type that contains an ordered set of either letters or numbers.

## UNIX

A multitasking, multiuser operating system designed in the '70s at the Bell Labs Research Centre. Written in C and assembly language.

## Variables

A data item that has been assigned a storage location in the computer's memory.

## X

Also X11 or X-windows. The graphical desktop used in Linux-based systems, combining visual enhancements and tools to manage the core operating system.

## Zen of Python

When you enter: `import this` into the IDLE, the Zen of Python is displayed.

Read  
More

Now you've got the basics down,  
you can improve and learn more  
essential skills in our next level guide...



Now Available on



[www.pclpublications.com](http://www.pclpublications.com)

Read  
More

# For Beginners


Tech Guides Available on  Readly

GET STARTED WITH YOUR WHATSAPP ACCOUNT

## WhatsApp For Beginners

✔️ Beginner-friendly Tips & Advice    ✔️ Step-by-Step Tutorials    ✔️ Cheat Full Colour Guides

Over 450 Tips & How-Tos Inside!



www.pcpublications.com    Papercut    100% RESPONSIVE

GET STARTED WITH GOOGLE'S CHROME OS NOTEBOOKS

## Chromebook & Chrome OS For Beginners

✔️ Beginner-friendly Tips & Advice    ✔️ Step-by-Step Tutorials    ✔️ Cheat Full Colour Guides

Over 475 Tips & How-Tos Inside!

For use with ALL Chromebooks & Chrome OS




www.pcpublications.com    Papercut    100% RESPONSIVE

GET STARTED WITH ADOBE ELEMENTS PHOTO EDITING

## Adobe Elements For Beginners

✔️ Beginner-friendly Tips & Advice    ✔️ Step-by-Step Tutorials    ✔️ Cheat Full Colour Guides

Over 445 Tips & How-Tos Inside!



www.pcpublications.com    Papercut    100% RESPONSIVE

GET STARTED WITH IOS 15 & IPADOS 15 FOR PHONE & IPAD

## iOS 15 For Beginners

✔️ Beginner-friendly Tips & Advice    ✔️ Step-by-Step Tutorials    ✔️ Cheat Full Colour Guides

Over 665 Tips & How-Tos Inside!




www.pcpublications.com    Papercut    100% RESPONSIVE

GET STARTED WITH ADOBE LIGHTROOM PHOTO EDITING

## Adobe Lightroom For Beginners

✔️ Beginner-friendly Tips & Advice    ✔️ Step-by-Step Tutorials    ✔️ Cheat Full Colour Guides

Over 455 Tips & How-Tos Inside!



www.pcpublications.com    Papercut    100% RESPONSIVE

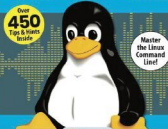
GET STARTED WITH LINUX CODING & PROGRAMMING

## Linux For Beginners

✔️ Beginner-friendly Tips & Advice    ✔️ Step-by-Step Tutorials    ✔️ Cheat Full Colour Guides

Over 450 Tips & How-Tos Inside!

Master the Linux Command Line!



www.pcpublications.com    Papercut    100% RESPONSIVE


GET STARTED WITH APPLE MACS & macOS VENTURA

## Mac For Beginners

✔️ Beginner-friendly Tips & Advice    ✔️ Step-by-Step Tutorials    ✔️ Cheat Full Colour Guides

Over 475 Tips & How-Tos Inside!

For use with ALL Mac OSes, MacBooks & Mac PCs




www.pcpublications.com    Papercut    100% RESPONSIVE

GET STARTED WITH APPLE NOTEBOOKS & SOFTWARE

## MacBook For Beginners

✔️ Beginner-friendly Tips & Advice    ✔️ Step-by-Step Tutorials    ✔️ Cheat Full Colour Guides

Over 460 Tips & How-Tos Inside!




www.pcpublications.com    Papercut    100% RESPONSIVE

GET STARTED WITH APPLE'S macOS MONTEREY

## macOS Monterey For Beginners

✔️ Beginner-friendly Tips & Advice    ✔️ Step-by-Step Tutorials    ✔️ Cheat Full Colour Guides

Over 475 Tips & How-Tos Inside!




www.pcpublications.com    Papercut    100% RESPONSIVE

GET STARTED WITH DIGITAL CAMERAS & PHOTOGRAPHY

## Digital Photography For Beginners

✔️ Beginner-friendly Tips & Advice    ✔️ Step-by-Step Tutorials    ✔️ Cheat Full Colour Guides

Over 455 Tips & How-Tos Inside!




www.pcpublications.com    Papercut    100% RESPONSIVE

GET STARTED WITH ADOBE PHOTOSHOP IMAGE EDITING

## Adobe Photoshop For Beginners

✔️ Beginner-friendly Tips & Advice    ✔️ Step-by-Step Tutorials    ✔️ Cheat Full Colour Guides

Over 450 Tips & How-Tos Inside!




www.pcpublications.com    Papercut    100% RESPONSIVE

GET STARTED WITH PYTHON CODING & PROGRAMMING

## Python For Beginners

✔️ Beginner-friendly Tips & Advice    ✔️ Step-by-Step Tutorials    ✔️ Cheat Full Colour Guides

Over 450 Tips & How-Tos Inside!




www.pcpublications.com    Papercut    100% RESPONSIVE

GET STARTED WITH RASPBERRY PI CODING PROJECTS

## Raspberry Pi For Beginners

✔️ Beginner-friendly Tips & Advice    ✔️ Step-by-Step Tutorials    ✔️ Cheat Full Colour Guides

Over 480 Tips & How-Tos Inside!




www.pcpublications.com    Papercut    100% RESPONSIVE

GET STARTED WITH WINDOWS 10 ON PC, LAPTOP & SURFACE

## Windows 10 For Beginners

✔️ Beginner-friendly Tips & Advice    ✔️ Step-by-Step Tutorials    ✔️ Cheat Full Colour Guides

Over 460 Tips & How-Tos Inside!




www.pcpublications.com    Papercut    100% RESPONSIVE

GET STARTED WITH MICROSOFT'S WINDOWS 11

## Windows 11 For Beginners

✔️ Beginner-friendly Tips & Advice    ✔️ Step-by-Step Tutorials    ✔️ Cheat Full Colour Guides

Over 475 Tips & How-Tos Inside!




www.pcpublications.com    Papercut    100% RESPONSIVE

GET STARTED WITH CODING & PROGRAMMING

## C++ & Python For Beginners

✔️ Beginner-friendly Tips & Advice    ✔️ Step-by-Step Tutorials    ✔️ Cheat Full Colour Guides

Over 455 Tips & How-Tos Inside!

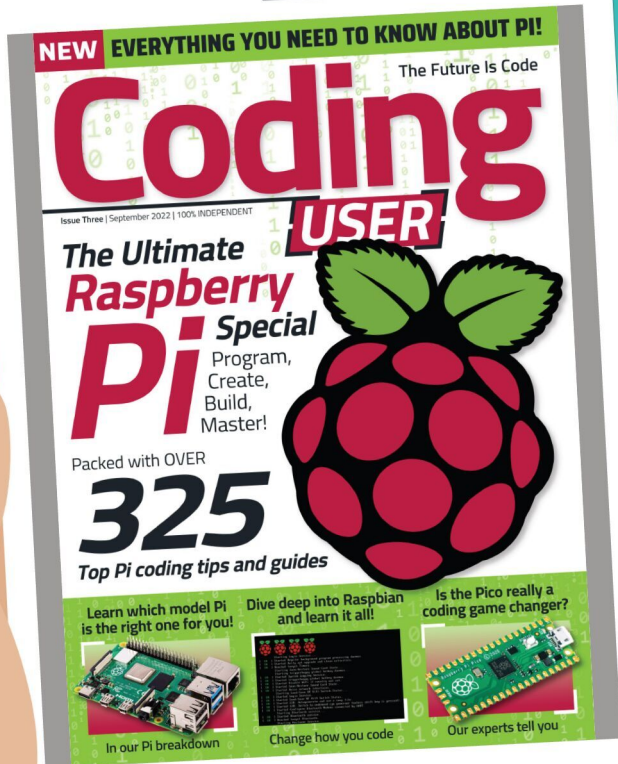


www.pcpublications.com    Papercut    100% RESPONSIVE

For a full list of titles available visit:

[www.pcpublications.com](http://www.pcpublications.com)

Want to master your Code?  
Then don't miss our **NEW** Programming  
& Coding magazine on  **Readly** now!



Click our handy link to read now: <https://bit.ly/3Ocl1zx>

**Raspberry Pi For Beginners**  
14th Edition - ISBN: 978-1-912847-33-4

Published by: Papercut Limited  
Digital distribution by: Readly AB - [www.readly.com](http://www.readly.com)  
© 2022 Papercut Limited. All rights reserved. No part of this publication may be reproduced in any form, stored in a retrieval system or integrated into any other publication, database or commercial programs without the express written permission of the publisher. Under no circumstances should this publication and its contents be resold, loaned out or used in any form by way of trade without the publisher's written permission. While we pride ourselves on the quality of the information we provide, Papercut Limited reserves the right not to be held responsible for any mistakes or inaccuracies found within the text of this publication. Due to the nature of the tech industry, the publisher cannot

guarantee that all apps and software will work on every version of device. It remains the purchaser's sole responsibility to determine the suitability of this book and its content for whatever purpose. Any app images reproduced on the front cover are solely for design purposes and are not representative of content. We advise all potential buyers to check listing prior to purchase for confirmation of actual content. All editorial opinion herein is that of the reviewer - as an individual - and is not representative of the publisher or any of its affiliates. Therefore the publisher holds no responsibility in regard to editorial opinion and content. This is an independent publication and as such does not necessarily reflect the views or opinions of the producers of apps or products contained within. This publication is 100% unofficial. All copyrights, trademarks and registered trademarks for the respective companies are acknowledged. Relevant graphic imagery reproduced with courtesy of brands and

products. Additional images contained within this publication are reproduced under licence from Shutterstock. Prices, international availability, ratings, titles and content are subject to change. All information was correct at time of publication. Some content may have been previously published in other volumes or titles.

 **Papercut Limited**  
Registered in England & Wales No: 04308513

ADVERTISING - For our latest media packs please contact:  
Richard Rowe - [richard@tandemmedia.co.uk](mailto:richard@tandemmedia.co.uk)  
Will Smith - [will@tandemmedia.co.uk](mailto:will@tandemmedia.co.uk)

INTERNATIONAL LICENSING - Papercut Limited has many great publications and all are available for licensing worldwide. For more information email: [james@papercutltd.co.uk](mailto:james@papercutltd.co.uk)

# Want to master your PC?

Then don't miss our **NEW** Windows PC & Laptop magazine on  **Readly** now!



Click our handy link to read now: <https://bit.ly/3y7gwFG>