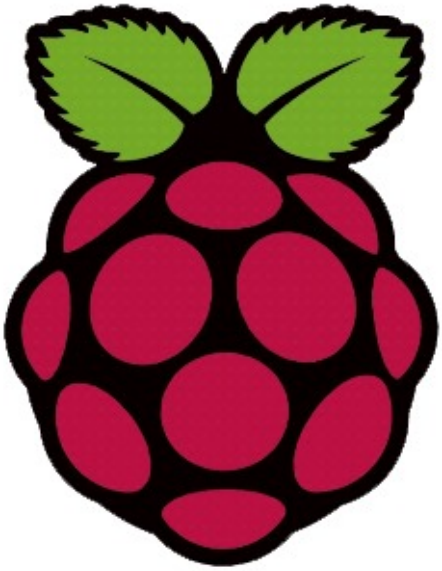
the centre for computing history

python



Introduction to Python

Using the Raspberry Pi

Before we get into any programming, let's first get started using the Raspberry Pi. The Raspberry Pi is a small, ARM-based single-board computer designed with education in mind. It can run several different Operating Systems and can be used for simple things such as browsing the web or watching movies, to more complex things such as robotics.

So let's get started. First, find and plug in your HDMI cable which you need to plug into the monitor you're going to use with your Raspberry Pi. Then grab your mouse and keyboard, and plug them into the USB slots on the Raspberry Pi. Lastly, find the your Micro USB power lead and plug it straight into the slot on the board, which is just next to where the SD Card is sticking out. This will turn it straight on, no need to press a button! If you haven't got Raspbian on your SD card, head to the Raspberry Pi site for a guide on how to install it on your SD.

So now some complicated text should be making its way up your screen, then it should say "raspberrypi login:". The default login for a Raspberry Pi is "pi" without quotes and all lowercase, so start typing that in and hit enter. Now it will say "Password:", which by default is "raspberry", or whatever you set it to when you set up the Raspberry Pi. When you type this in you won't be able to see it, this is so other people can't see your password over your shoulder. You should now be logged onto the Raspberry Pi Terminal!

Then to get to the Desktop where we will be doing the work, enter "startx" and hit enter. As a last step, connect your raspberry pi to the internet, then load up the terminal and type the following:

Sudo apt-get update && sudo apt-get install geany

Give it a minute, and we should be good to go!

So now we're at the desktop, it's time to start programming. For our programming we're going to be using that piece of software you just installed called 'Geany'. This is basically a text editor which can make our code a bit easier to read by splitting it into colours and giving structure to our program automatically. So load it up, the icon should be right there on the desktop.

Once it's up, make sure you create a new document for you to use by going in the top left to File then selecting New.

One last thing before we can get started is we need to save the file. Click on File once more, then Save As. Now in this new window, on the left click on Desktop and then at the top put in the file name. It can be anything you want as long as at the end of the file you put .py which is the file extension for Python files. Now we're all set up.

Click on the blank space to make sure that when we type it is entered, and let's enter our first line of code. Type this into the text editor:

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

As you see the colour separates the code out. The function (print) is blue, and the text we want it to print is orange. It is surrounded by a (and) because it is a function, and when we use this function whatever we tell it should be printed into the console when we execute the code. So let's try that. Press F5 on the keyboard.

After a few moments, the terminal should pop up with what we told it to say, easy! You can go back and tell it to print whatever you want. You can have it print out multiple lines too by repeating the Print command for each new line that you want. Try this:

```
print ("Hello world!")
print ("How are you?")
```

Press F5 again to execute the code, and you should see two lines now, great!

So while we've printed something to the console, we want to do something a bit more programmer like, so let's start using Variables. A Variable is a basically a word or name which holds onto a value. So say you wanted to use the number 10 somewhere in your program once or even a few times and you can just make it a variable. So let's give it a try. Either delete the print's we used a minute ago, or just put it underneath. When you're done press F5 to execute.

```
x = 10
print x
```

So here what we did: we defined the variable "x" and gave it the value of 10. So every time that x is used in our code it is interchangeable with 10. Because it was a number, x is an Integer. Also notice that we don't put quotation marks around x this time at print, this is because when something is surrounded by the " " it is a String, and all a string is a word or a sentence like we did before. If you want to see the difference, put some quotation marks around the x and see what happens.
But you can set a variable to anything, including a string so let's try this.

```
x = "Hello how are you!"
print x
```

See how a variable can be set to a string as well?

Now what about maths I hear you ask? Well Python is good at that too, so let's give that a quick shot. Try each of these separately. When you're done, try just putting in any numbers you want into it to see what you get, no matter how big or small. In these we're using lots of variables, and some more prints.

Addition

```
x = 5 + 7
one = 1
two = 2
three = one + two
print x
print three
```

Subtraction

```
x = 5 -7
y = 12
z = 5
print x
print y-z
```

Multiplication

```
x = 4
y = 6
result = 4 * 6
print result
```

Division

```
x  = 25
y = 5
result = 25 / 5
print result
```

These are the four basic maths operators. So + is adding, - subtracting, * is multiplying and / is dividing. You can do more complex things like remainder (%, also called modulo) of division and exponents (**) but for now let's not worry about that.

Spend a bit of time messing around this, doing whatever you want. You can do more than one maths operator in a line too for instance x = 5 * 3 + 7 / 25. Remember also that you can name a variable anything you want!

It's all good doing these things, but it's a bit boring without any interaction and getting involved, so let's make it so that you can get some user input!
Let's start by asking someone for their name:

```
print("Welcome to my awesome program!")
name = raw_input("What is your name? ")
print("Hello " + name)
```

Then hit F5 to get to the console. You'll notice that after it asks what your name is, the program stops and the cursor begins to blink. This is it waiting for the users input and will begin running again when they hit enter. What raw_input does is ask the user a question, and you set the question in between the brackets after raw_input. In this case we ask the user for their name. Then once we have their name, we want to print it. Here we do some addition on the strings by just adding them together, so that it makes one sentence for us to print, and we do all this straight after the print no need for other variables!

But what if we want to make it so that if one particular person enters their name, that they get to see a secret message. This is where conditions come in. Before we got straight to only displaying a message to certain people, let's look at a few quick examples. Try putting these a blank code editor window and running them to see what happens. Remember to include all the capital letter too!

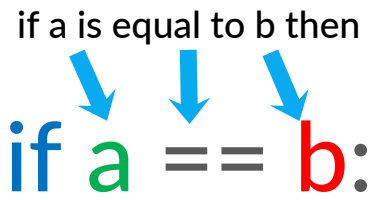
Example 1

```
win = True
if win == True:
        print("You won!")
        print("Congratulations!")
else:
        print("You did not win")
print("We hope you're happy with the results!")
```

Example 2

```
x = 11
if x == 10:
   print("The number equals 10!")
elif x > 10:
   print("The number is larger than 10!")
elif x < 10:
   print("The number is smaller than 10!")
```

The first thing introduced here is the if statement. What the if statement does is it takes the half of the statement, and see's whether it is equal too/greater/lesser or not the same as the second half. The colon at the end represents then. So in plain English this is what it means.



Then from there, everything that is tabbed in from that point is run from the if, so if we look at the first example you'll always notice that the final print statement will always print. This is because in python and indent to the right means begin, and an indent to the left means end. A good way to think of it is in blocks, so after the if we start a block, and then when it gets indented back to the left a block then ends. If this is a bit confusing, just ask the person teaching you to explain it again. Remember to tab we use the tab key.

```
win = True
if win == True:
    print("You won!")
    print("Congratulations!")
else:
    print("You did not win")
print("We hope you're happy with the results!")
```

Now that's out the way, let's go back to displaying a secret message to only one particular person. If you want to give it a try before reading how to do it on the next page then give it a shot, otherwise turn over and we'll go through it.

So this is where we were before. If you kept this file open with this in then skip ahead a bit, but if not here is our code from last time:

```
print("Welcome to my awesome program!")
name = raw_input("What is your name? ")
print("Hello " + name)
```

So we know we have to use an if statement, but first we want to figure out what person we want to let see the secret message. Put after that first section.

```
VIP_name = "Ed"
```

Replace Ed with your name or someone else's who may use your program. All this does is declare a variable, and attach a string to it shown by the " " with the content in-between them.

Now we have our name it's time to make an if statement. We want to make sure that this is exactly equal to the VIP's name so we need to use a ==. So decide what you want your secret message to be and enter this code below what you already have in the text editor.

```
if name == VIP_name:
        print("You're super awesome!")
else:
        print("Nothing to see here...")
```

What this does is compare the entered name with our VIP's name. But we're missing something, what if the person isn't they say they are, or someone else with the same name tries to use the program? Well to make it a bit more fool proof lets go back and ask the user for their age too. Put this after the program print statement for saying Hello to the user:

```
age = raw_input("And how old are you? ")
```

Now just after VIP_name = "Ed" put this:

```
VIP_age = "20"
```

Change the value to match that of the VIP. Now we have just one last thing to do, and that's to change the if statement we made a minute ago. We're going to be using the and condition. Change the if so it matches this:

```
if name == VIP_name and age == VIP_age:
```

This means that if and only if both the name and the age match the VIP's then they will get to see the secret message. You can replace the and statement with an or, which means that if the users name or age matches the VIP's then they will see the message.

If at any point you get a bit stuck or confused, here's the whole completed code for you to see.

```
print("Welcome to my awesome program!")
name = raw_input("What is your name? ")
print("Hello " + name)
age = raw_input("And how old are you? ")

VIP_name = "Ed"
VIP_age = "20"

if name == VIP_name and age == VIP_age:
        print("You're super awesome!")
else:
        print("Nothing to see here...")
```

Spend some time just going over this, making sure you understand it and if you don't then ask for some help from someone. You can also try expanding on this with what we've learnt so far, change some things and make your program even better!

Loops are an important part of any program. For instance you can go over a list of names or print out a times tables with only a few lines of code. Here's some examples of a few of the different types of loops and see if you can figure out what's happening and by typing them out and running them:

Example 1 – For Loop

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

Example 2 – While Loop

```
count = 0
while count < 10:
        print count
        count = count + 1
```

When you run these, you'll notice that they both print the same thing! In the first example we use one of pythons in built functions called range. This makes variables for every number between 0 in the form of an array and whatever number you put between the brackets. You then loop through each of these numbers, and print each one. You'll notice that it doesn't include 10 itself, so whenever you use range remember to put the number 1 higher than the number you want to stop at in the brackets.

In example two we use a while loop. The while loop continues going until it reaches its condition (Which is similar to the if statement syntax). In our while statement it carries on going while count is less than 10, and when it reaches 10 or higher the while loop will stop which is why it prints up to 9. Inside the loop after the print we have a bit which increments count up by 1, so if the count is 2 at the start of the while, at the end it will be 3. However you do need to be very careful with a while and make sure that after some time the loop will break, otherwise it will get stuck in an infinite loop and make your program unusable!

If you need any extra help with any of the content in this worksheet, don't be afraid to ask! We would be happy to help!