Touring Turing

[Fork me on GitHub](https://github.com/MarcScott/7-CS-Turing)

* [Week1](1_Lesson.html)
* [Week2](2_Lesson.html)
* [Week3](3_Lesson.html)
* [Week4](4_Lesson.html)
* [Week5](5_Lesson.html)
* [Week6](6_Lesson.html)
* [Week7](7_Lesson.html)
* [Assessment](assessment.html)

## 1 Alan Turing

Alan\_Turing\_photo.jpg

### Learn It

* In 1936, a few years before the outbreak of World War II, Alan Turing invented the **Turing Machine**.
* This was a *hypothetical* device (not one to be actually built) that could read and change symbols on an infinitely long tape, according to a table of rules.
* With this simple idea, Computer Science was born, as the **Turing Machine** was theoretically capable of performing any calculation that was computable.
* It took a few more years before actual computers were constructed, but even today we can talk about computers and programming languages in terms of their Turing completeness. This means that they are able to do exactly what a **Turing Machine** can do.

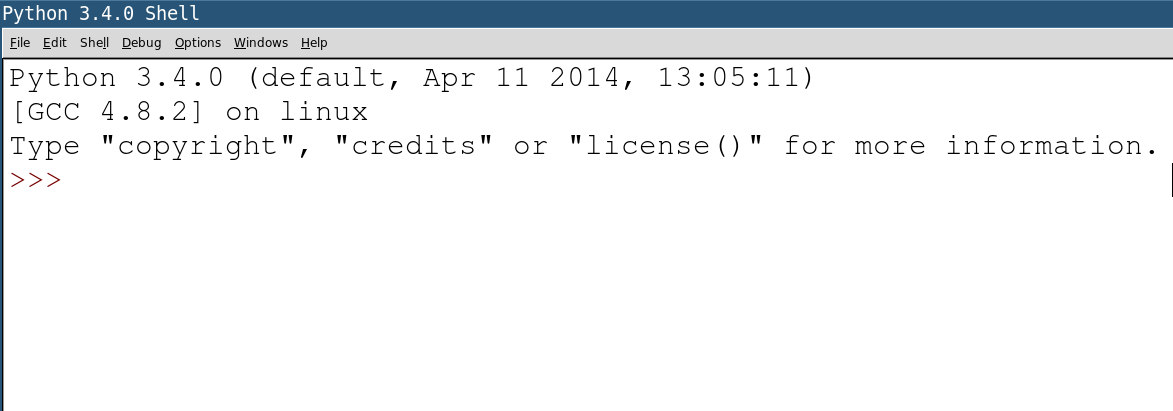
## 2 Python Variables

### Learn It

* For this project, we'll be using the Python 3 Programming language.
* You've used Scratch before, which is a Graphical Programming Language. Python does the same sort of thing, but you program using text instead of blocks.
* If you're working on this outside of school, you can download it from the following links.
* [Python 3.4 for Mac](https://www.python.org/ftp/python/3.4.2/python-3.4.2-macosx10.6.pkg)
* [Python 3.4 for Windows](https://www.python.org/ftp/python/3.4.2/python-3.4.2.amd64.msi)
* If you need Python 3 for Linux - use your package manager sudo apt-get update && sudo apt-get install python3 idle3 (on Ubuntu for instance)
* If you're working in school, then you should find a program called IDLE, in your programs menu.

### Try It

* Open up IDLE. You should be presented with a screen that looks a little like this:



* IDLE is an Integrated Development Environment (IDE) for Python. This means we can create, edit and run Python files using it.
* The screen that popped up is called the INTERPRETER. Remember this, and what it looks like.
* We can write code in the interpreter, and it gets executed straight away.

### Learn It

* One of Turing's key ideas was an infinitely long tape, upon which symbols could be added, read and changed.
* In modern computers this tape is called RAM, and is made up of very complex circuits.
* Just like in a Turning machine though, we can add data to RAM, read data from RAM and change data in RAM.

### Code It

* Let's write some data to RAM.

myName = 'Alice'

* You can obviously use your own name.
* You have now stored your name in RAM. You have done this using a variable
* For now think of a variable is being made up of an *identifier* (myName) and a *value* (Alice)

#### Note - Once you have hit ENTER you can not delete what has been executed, any more than you can unbreak an egg. If you make a mistake, just type it out again (or hit alt + p on your keyboard to bring up the last command again, and alter it from there.

### Try It

* Turing's original idea called for an infinitely long tape, that we could store an infinite amount of information on.
* RAM is pretty big, even if it's not infinite. A typical Computer might have anywhere between 2Gb and 16Gb or RAM.
* This would be enough space to store between 1000000000 and 4000000000 characters, which is enough for us.
* Try creating some variables for your friend's names. Each one must have it's own unique *identifier* though

firstFriendName = 'Robert'  
secondFriendName = 'Charles'  
thirdFriendName = 'David'  
fourthFriendName = 'Eve'

### Learn It

* We have to be careful with *identifiers* for our variables.
* Try this and see what happens:

5thFriendName = 'Faith'

* You'll get back a message saying you've used invalid syntax.
* This is known as a syntax error. It means you haven't conformed to the rules of the language.
* Here are the rules for Python's Variable Identifiers:
  1. Must not start with a number.
  2. Must not contain symbols like !, @, #, $, % etc.
  3. Must not contain spaces (use \_ or - instead)
  4. Are case sensitive (myname, myName, MyName and MYNAME would all be different variable identifiers.)
* There's also a few words you should avoid for identifiers, like not and continue, but you'll learn these as you go along.

### Code It

* Now we have some variables stored, we'd best learn how to read them, as that was one of the stipulations of the Turing machine's tape.
* You can read the value of a variable just by writing it's name.

myName  
secondFriendName

### Code It

* We also needed to be able to change the variables.
* Let's have a go at that.

firstFriendName = 'Bob'  
secondFriendName = 'Charlie'  
thirdFriendName = 'Dave'

* Now use the identifiers to read the values and make sure they have changed.

### Badge It - Silver

* Assign variables for all your family. For instance;

mum = 'Alice'  
dad = 'Bob'  
cat = 'Eve'

* Demonstrate that you can display the variable values in your INTERPRETER

## 3 Python Data Types

### Badge It - Gold

* Not all variable values are the same.
* Copy and paste the following into your interpreter - **one line at a time**

foo = 10  
bar = 3.142  
baz = 'Hello World!'  
qux = True

* Now, for each of the variables, you can find out it's type using the type() inbuilt function.

type(foo)

* The interpreter will tell you the class of the variable.

<class 'int'>

* This would tell me that the data-type was an int, which is short for *integer*.
* Look up each of the data-types for the variables above and then use the web to find out what they each mean (in simple English).

## 4 Calculations in Python

### Try It

* We can use Python to perform calculations.
* For instance, you can type into your INTERPRETER

6 + 4

* In Python we can use + and - for addition and subtraction, / and \* for division and multiplication.
* There are lots of mathematical operators we can use. Here is a table of all of them for reference later on.
* In the examples below a = 10 and b = 20

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| + | Addition - Adds values on either side of the operator | a + b will give 30 |
| - | Subtraction | b - a will give 10 |
| \* | Multiplication | a \* b will give 200 |
| / | Division | b / a will give 2 |
| % | Modulus - **remainder** of a division | b % a will give 0 |
| \*\* | To the power of | a\*\*b will give 10 to the power 20 |
| // | Division and round down | 9//2 is equal to 4 and 9.0//2.0 is equal to 4.0 |

* We can also use brackets, like you would do in maths.

(24+1)\*4

### Badge It - Platinum

* Use Python to find the answers to the following questions.
  1. 1929 x 64
  2. 39483 ÷ 321
  3. The remainder when 123 is divided by 2
  4. 1234 raised to the power of 4
  5. What is 15% of 50 (remember that % in Python is modulus and not percent)