PYTHON
YEAR 10 RESOURCE

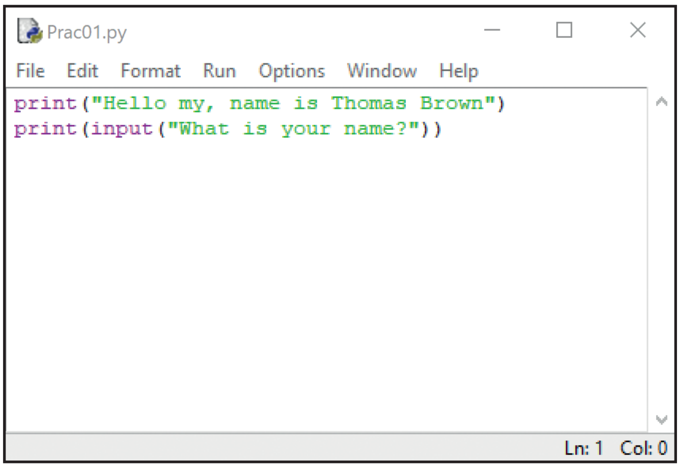
KS3

Practical 01: Printing to the Shell

To program in Python you need the latest version of Python, which is freely available at www.python.org. Your school will have this installed on the computers for you, but you can try some of the coding at home if you wish.

Integrated Development Environment

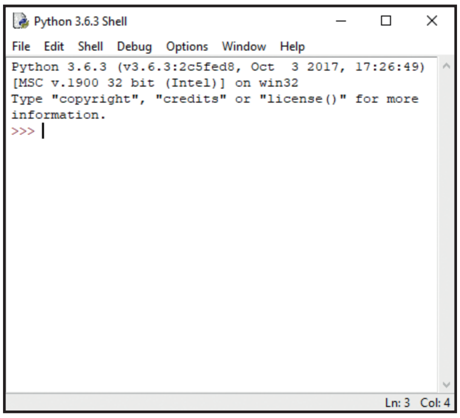
Within Python we have an Integrated Development Environment (an IDE). This is a fancy text editor, which allows us to enter our code, save it and then allow the Python interpreter to execute it. For coding Python, we will use the bundled IDE called IDLE. As shown in Figure 1, the editor resembles notepad and it will provide all the tools we need to build complex programs.



```
print("Hello my, name is Thomas Brown")
print(input("What is your name?"))
```

Figure 1

When we save our code, it is saved with the **.py** extension. This is similar to saving a Word document as **.docx** or a file from Notepad as **.txt**. We also mentioned that Python uses an interpreter to execute the code. This is similar to a language interpreter, which takes the Python code we typed into the IDE and "interprets" it to a language that is understood by the computer to allow it to carry out the tasks you have told it to. The output from these tasks is displayed in a separate window called the Python Shell (see Figure 2).



```
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49)
[MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more
information.
>>> 
```

Figure 2

One last thing before we start: the computer is only as clever as the person typing the code! If you want a program to do what you want, you must type the instructions (code) correctly and carefully.

Tic Tac Toe

Over the next few weeks, we will be developing your programming skills by learning and practicing with different programming tools. While we'll start by practicing the different skills individually, we will need to combine them to create a useful, real-world application. To practice this, as we master each skill, we will incorporate it into our own application: a version of the Tic Tac Toe game. You will then be able to test and play the game in pairs during the final few weeks of this course. In the first week, we will simply set out the grid that we need to play the game as seen in Figure 3 below. Then in the coming classes, we'll keep adding new functionality until the game is finished.

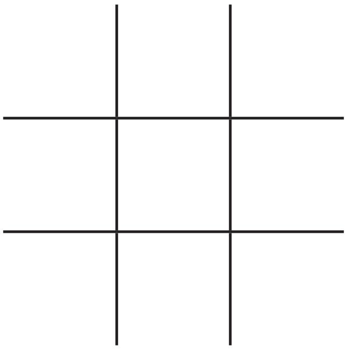


Figure 3

Before we try to print the grid, let's start by printing our name to the Shell. This may seem boring and a bit simple; however, complex programs are built by combining lots of simple functionality. In this case, we can use what we learn from printing out name to print out the Tic Tac Toe grid.

Let's Code!

The first thing we need to do is open the IDE for Python, called IDLE.

(1) Open IDLE on the computer you are working on. If this is a Windows computer running Windows 10 then type IDLE in the search bar at the bottom of the Desktop, see Figure 4, then click the program to run.

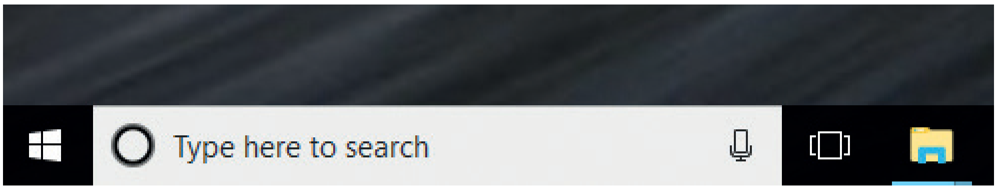


Figure 4

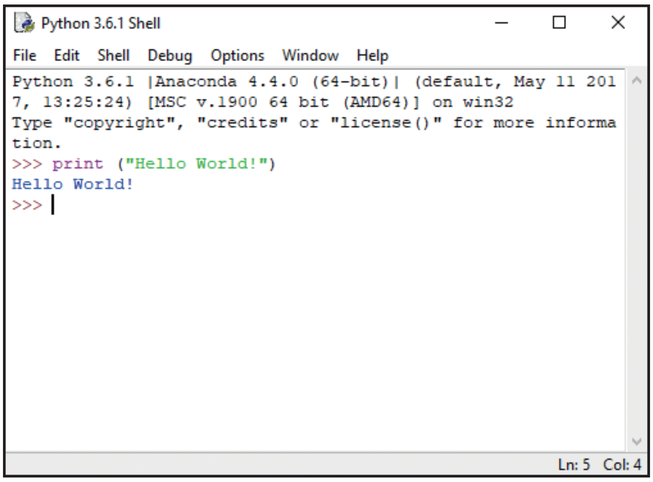
At this point the Python Shell should open. The first thing we will do is program directly in the shell window (see Figure 2). To start programming we need to see three greater than signs (>>>) .

(2) At the >>> characters type the following code:

```
print("Hello World!")
```

(3) Press enter on the keyboard and **Hello World!** should appear on the next line (in blue text), see Figure 5. If it does, well done! You have run your first program. If not, then we need to debug.

> Debug means to find and get rid of all bugs (or errors) from your code. Make sure you have typed the line of code EXACTLY as it is above. The most common mistakes include using uppercase or lowercase letters in the wrong place and having brackets/quotes in the wrong place/missing/unclosed.



```
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 201
7, 13:25:24) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more informa
tion.
>>> print ("Hello World!")
Hello World!
>>> 
```

Figure 5

(4) Let us try this again, but this time, print your name. Type the following code in the Shell window replacing the words "Your Name" with your own name:

```
print("Your Name")
```

(5) Press enter on the keyboard and your name should appear on the next line (in blue text). If it does, well done! You have now made your program a little more unique. If not, then we need to debug.

What can we learn? And what programming jargon can we pick up?

Let's look at what we typed.

```
print("Hello World!")
```

The easier part of this is the bit in quotes, it's called a string. In programming, a string is a series of letters, numbers and/or characters that join to form words or data that we need our program to store or use. Python doesn't really care about what is in between the quotes – it doesn't check for spelling, case sensitivity or anything like that. It trusts you not to make a mistake!

The `print()` statement is described as a **function**, which is going off into the Python library, finding where `print()` has been defined and executing what is in that function. In a bit more detail, the `print()` function stores all the code needed to print something out to the Python Shell, it simply needs to know what to print. This is the part that goes inside the brackets, whatever is there will get printed out to the Python Shell, as long as there are no errors.

> Something to remember!! Python is case-sensitive meaning that `print()` is not the same as `Print()` nor is it the same as `PRINT()`, or any other combination of upper and/or lower-case letters. This is one of the biggest issues when learning to program.

Programming in Script mode

What we have tried out so far is Python programming in interactive mode, by typing code directly into the Shell. This is beneficial for testing the logic of a line (or few lines) of code; however, you may have noticed there is no option to save this code. This isn't very useful if we can't save our code and run it at a later point! This is where Script mode comes in.

(6) From the IDLE (Python Shell) **File menu** choose **New File**. A new window should open, as per Figure 6.

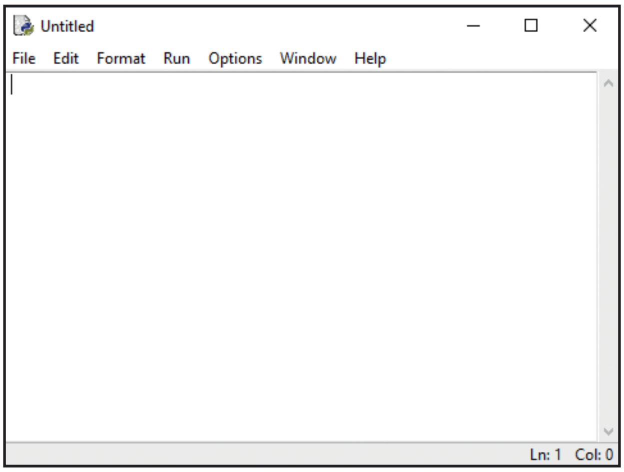


Figure 6

This is the window where we can type all the Python code we want, save it and run it at a later point, but remember to save often! We don't want to lose precious code (especially when it is working).

(7) From the IDLE **File menu**, choose **Save As...** and save the file to a place you will remember e.g. a USB drive, Dropbox, OneDrive. Name the file `hello_world.py`.

> Note the **.py** extension for this file. This will tell the computer system what to expect inside the file i.e. Python code. You should be familiar with **.docx** for Microsoft Word or .pdf for Portable Document Format files.

(8) In the newly saved **hello_world.py** file type the following line of code:

```
print("Hello World!")
```

(9) Press enter... what happens? Hopefully nothing! As we are in Script mode we can continue to type as many lines of code as we want in this window. So how do we execute (or run) this file to see the output?

(10) From the IDLE **Run menu**, choose **Run Module**. This should pop the Shell back to the front of your windows being displayed and you should see Hello World! printed to the Shell again, similar to Figure 5.

Next Week

Next time we will be looking at what happens when there are errors in the code, how to add code comments, how to get user input and introduce the use of variables.