Programming with Python

Module 01

In this module, you learn about what Python is and how it is used. This is only the start of our course, so the content will only be at a high-level overview.

# What is Python

Python Programming Language – Official Website

"Python is a programming language that lets you work more quickly and integrate your systems more effectively. You can learn to use Python and see almost immediate gains in productivity and lower maintenance costs.

* Python runs on Windows, Linux/Unix, Mac OS X
* Python is free to use, even for commercial products, because of its OSI-approved open source license
* There are two main versions of Python 2.x and 3.x. Both work, but 3.x has improved and advanced features.
* You can find out more about the difference from the Python website here: <http://wiki.python.org/moin/Python2orPython3> (opens external site)
* Our textbook uses version 3.x, but Mac already has 2.x installed. I recommend installing and using 3.x. You will just have to remember to use the correct version when you are running your scripts (Figure 1).

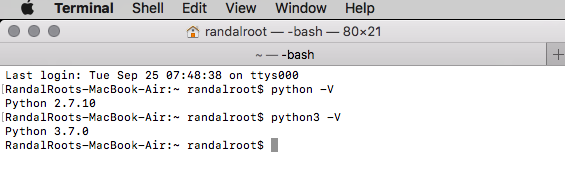


Figure 1: Running multiple versions of Python on the Mac OS.

# Installing Python

Installation is easy, but you can always search for video tutorials based on your chosen OS. <https://www.google.com/search?q=How+to+install+python> (opens external site)

# Running Python

You can create and run python code using its built-in editor called IDLE.

<https://www.google.com/search?q=How+to+use+Python+idle> (opens external site)

One Windows, the basic steps are (Figure 2):

1. Open the Start menu
2. Open the Start menu locate or search for IDLE under Python
3. Launch IDLE from the link you found
4. Wait for the IDLE Python Shell to open

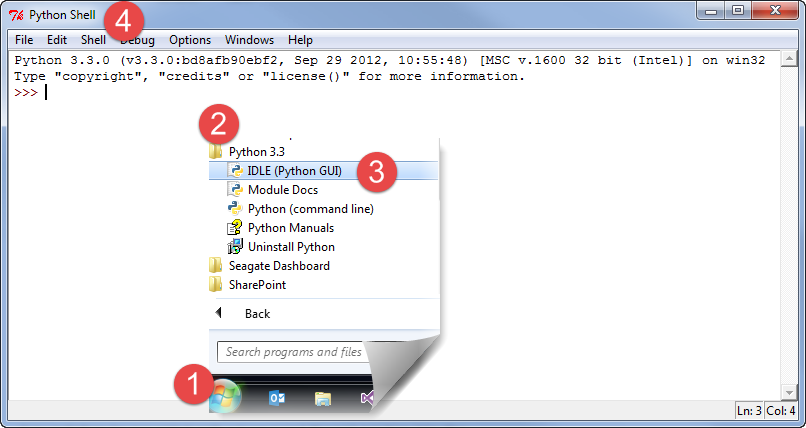


Figure 2. Steps to open the IDLE Python Shell Application

## Console Applications

Once you have installed Python, you can create a program that runs as a Console application in the OS Command Console/Terminal.

To open a command console in Windows, you open use the Start Menu ➤ Run (windows key + r) and type in the command “CMD” (Figure 3).

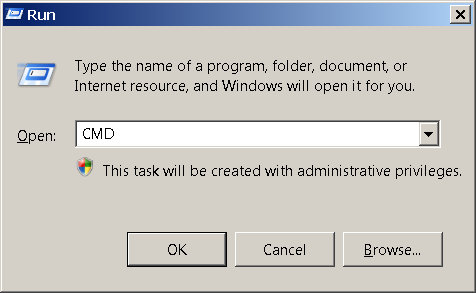


Figure 3. The Windows Run Dialog window

You will then be presented with a command prompt in the command console window that looks like Figure 4.

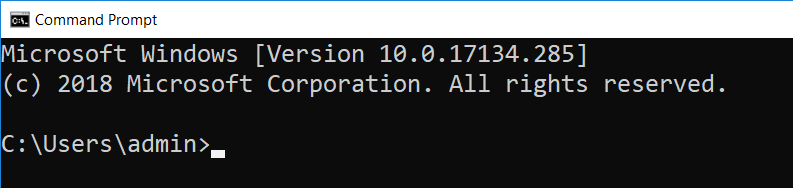


Figure 4. A Windows Command prompt

If you are using a Mac, this it is almost the same, but now it is called a Terminal window. Open a terminal window using Finder > Applications > Utilities > Terminal.app and you will see something similar to Figure 5.

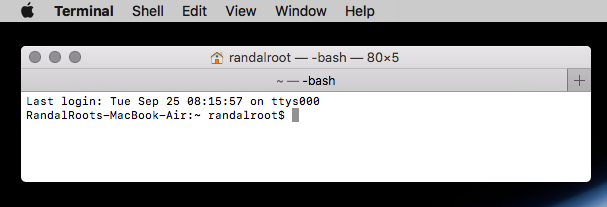


Figure 5. A Mac Command prompt

With the Command Console/Terminal open you can run “Console Applications”. These applications are not fancy, but they do allow you to accomplish useful tasks on a computer with the minimal fuss as to making your application look nice! IPConfig.exe is a good example of a Console application.

To see what it does, type in the command “IPConfig.exe” on the Windows OS and hit the Enter key to run the application(Figure 6) or ifconfig on the Mac OS (Figure 7).

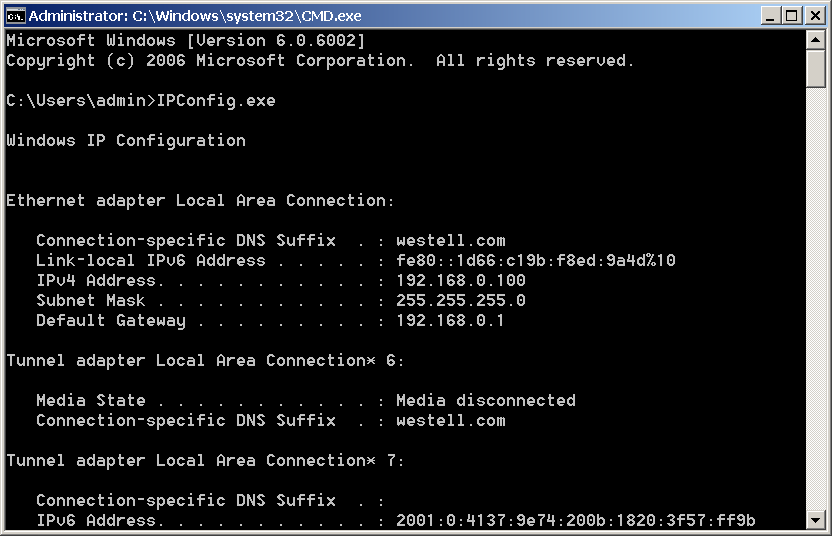


Figure 6. Results from running IpConfig.exe

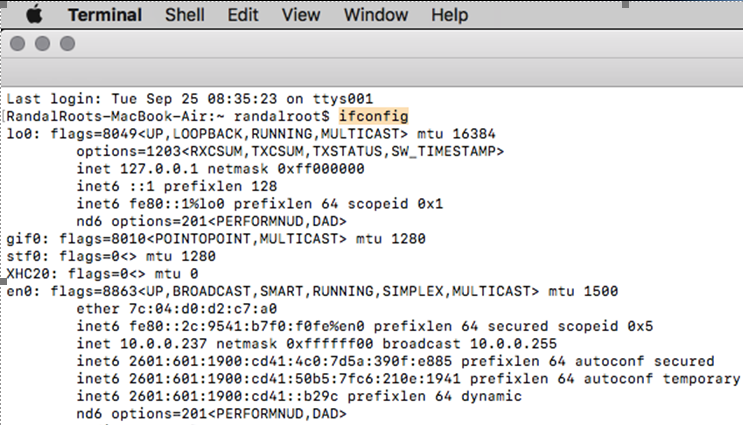


Figure 7. Results from running IpConfig.exe

# Programming Basics

In a very general way, programs work with two things:

* ***Data:*** the information you want to work with, such as a person’s name and phone number.
* ***Operations:*** are the things you want to do with the data, such as printing data.

While data and operations are the core of the program, your program’s code will include things like *comments*, *namespaces*, *directives, and statement*:

* ***Comments:*** provide additional information to humans.
* ***Namespaces:*** provide an easy way to organize your code into named groups.
* ***Directives:*** provide additional information to the computer, but are not directly part of the program.
* ***Statements:*** the commands you add to a Python code file.
* A *statement* is one instruction to the computer
* Each of these statements will be made up of one or more keywords or symbols (sometimes called *tokens*)
* Since there can be more than one token per statement, you need a way to indicate to the computer that you are done with a statement. In Python, you do so with a *carriage return, or* optionally a semicolon (;), at the end of the statement.

**Note**: *Semicolons are considered wrong by many of the python faithful!*

Here is an example of three Python statements (Listing 1).

x = 4 # This is one statement

y = 5 # this is another

z= x + y # and another as well

Listing 1

## Comments

Programmers often use comments to identify the purpose of each statement (as we did in Listing 1).

**#** This is a standard, *inline*, Python comment.

Commenting code is useful to add notes like these, but also useful when you want to see if disabling a set of statements solves a problem. Any code that follows a comment will not be processed.

If you “*comment out”* a section of code, and the problem disappears, then you know that the error is related to that set of statements.

print('test')

**#**print 'test' without parenthesis only works in 2.x

A comment only affects one line of code unless you use a "block" comment. Block or *multi-line*, comments can be un-officially made in Python using 3 single quotes like this:

**'''**

Both these statements are commented out for testing

int x = 5;

int y = 10;

**'''**

Listing 2

**TIP:** Block, or *multi-line*, comments may not officially be available in Python, but other languages commonly support them and often look like this:

/\*

C Style languages use a slash-star and star-slash pair for a block comment.

Note to self: Both these statements are commented out for testing

int x = 5;

int y = 10;

\*/

Listing 3

# Case-Sensitivity

Python is a case-sensitive language, so you must be careful as you type (Listing 4).

x = 4 # This places the value of four into a variable called x

X = 13 # But, this places the value of four into a variable called X !

print(X) # displays the value 13 to the user

**PRINT**(X) # this command is not understood by Python

Listing 4

# Functions

Programmers have found that it a good practice to organize your code into groups, so statements are often grouped into *functions (*also known as *methods* or *sub-procedures)*. After you create a function, you can run its group of statements by *calling* the method (Listing 5).

def DemoFunction():#Create a function

print("This is a statement in DemoMethod")

print("This is another statement in DemoMethod")

#End DemoFunction

DemoFunction() #Call the function

Listing 5

Code in a function is processed one statement at a time, starting at the “top” of the function and ending at the bottom of that function. In Python, functions have a clearly defined start, indicated by the “def” keyword, but the end of the function is determined solely though indentation of its code. Any statements indented under the function’s definition statement, will be considered part of the function. You can place a comment at the bottom of the function if you want to make the end more obvious.

## The print() Function

The print() function was created in Python to print out information to the command window. If you remember, IPConfig.exe wrote out its data to the command window for a human user to read. You can send output to your users in the same way.

## The input() Function

The input() function gets data from the program’s user. It is also used to “pause” the program and wait for a user’s response.

**Note**: In Python 2.x you use the raw\_input() function instead of the input() function to avoid and error!

The Input command can also be used to pause or stop the CMD (command) window from closing once the script finishes (Figure 8).

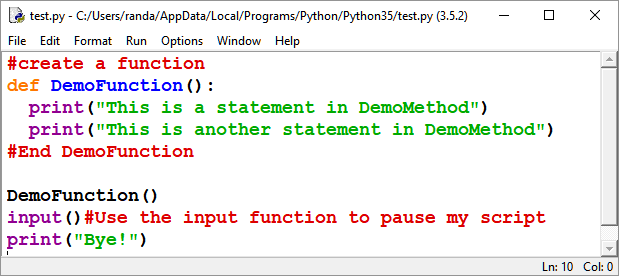


Figure 8. A Python script using the input function

**Tip:** While it is used in your book and in the assignment this module to just pause the program, we will see more practical uses for it later.

You can see the “pausing” behavior if you run a Python script from Windows Explorer with and without the input function (Figure 9).

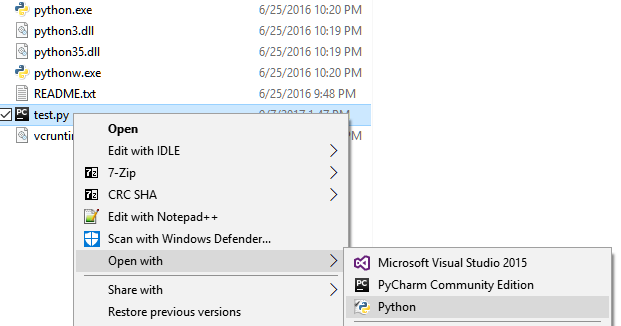


Figure 9. Running a Python script file from Windows Explorer

Running it without will cause the command window to immediately close once the script’s code has finished. Adding the input() function will force it to pause and wait for the user presses the Enter key (Figure 10).

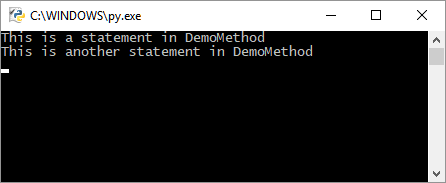


Figure 10. The results of the script file while paused

# The Main Body of a Script

Many languages use a special “Main()” function to indicate which code should run as when a program or script is started. However, in Python this method is implied. Any code inside of a Python script is implicitly part of the script’s Main body.

Since the entire body of the script file is the Main() function, any code you type in will be processed one line after the other from the top of the script to the bottom.

If the script finds a function definition at the top of the script it will load its code into memory, but will not run that code until it is latter called (often from the Main script body).

When you call another method from the Main body, it will jump to that method, run the statements inside of the called method, and return to the Main body once it is done. The example below outlines the order in which your statements will be processed (Listing 6).

def DemoMethod(): # 3) jumps to here and run both statements…

print("This is a statement in DemoMethod")

print("This is another statement in DemoMethod")

#End DemoMethod

# 1) Start of Main

print("This is a statement in the invisible Main method")

DemoMethod() # 2) call the method DemoMethod()...

# 4) jumps back to here…

print("This is another statement in the invisible Main method")

# 5) End Main (the program ends! )

# LAB 1-1: Create a Console Application

1. Create a new script using IDLE that prints out the text, “This is a test!” Do not use the **input**() function yet.
2. Run the script using Window Explorer. Note how the command window closes immediately after the script finishes.
3. Add the **input**() function to pause the script until the user presses the **Enter** key

# Summary

In this module we looked at the what the Python language is, how it is installed and how it is used. We also covered the different components that make up a Python script. In the next module, we will look at more details about these components.