PROGRAMMING WITH PYTHON

By Randal Root

Module 01

In this module, you learn about **what Python is and how to use it**.

What is Python

Python is a programing language designed to be **simple yet powerful**. It is one of the easier programming languages to learn but is **powerful due to the extensive pre-made code modules** you can download (code modules are files with code in them.) Python is **free to use, even for commercial products**!

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.

Installing Python

Python i**nstallation 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** (external site).

There are **two main versions of Python 2.x and 3.x**. Here are some facts you need to know.

- Both version 2.x and 3.x **can be installed on the same computer**
- **Mac** and some PCs **already have 2.x installed**
- Version **3.x has improved and advanced features**
- Our textbook uses version 3.x, so please **install use 3.x for this course**

You can find out more about the difference from the Python website here. **http.//wiki.python.org/moin/Python2orPython3** (external site)

Note. If you install both, make sure you **remember to use the correct version** when you are running your scripts. **You can check out which version you are running using the -V switch from the command terminal of your computer (Figure 1)**.

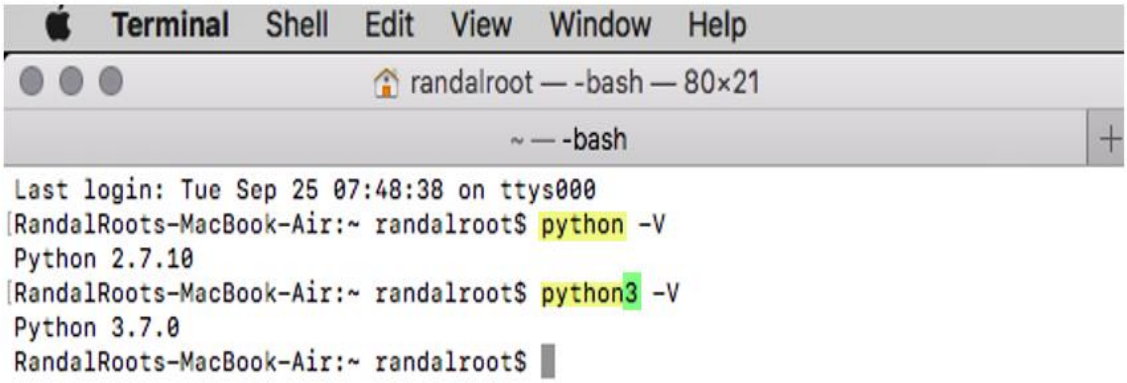

Figure 1. Running multiple versions of Python on the Mac OS.

Installing on Windows

To install on Windows.

1. Download the installation program (.exe) from **https.//www.python.org/downloads/** (external link) website (Figure 2).

Figure 2. Downloading Python's installation program

2. Run the .exe file to start the installation.

3. Use the **custom option to choose an easy to access location like C.\Python3.x** and **check the "Add Python 3.x to PATH" checkbox**, to including the Python executable in the **OS path** (Figure 3).

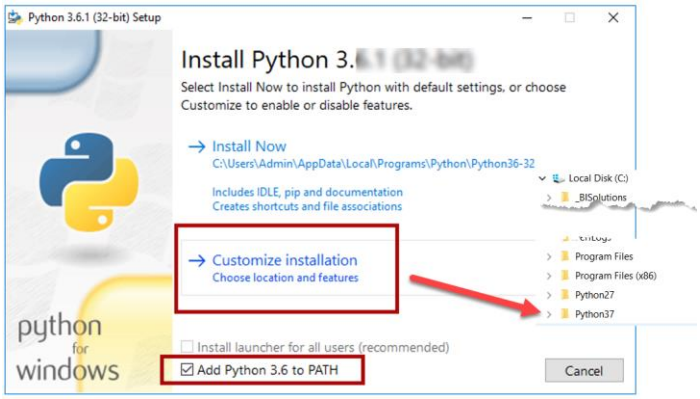

Figure 3. Customizing the Python installation

Installing on Mac

The simplest way to install Python 3 on a Mac is to.

1. Download the latest Python version from Python.org downloads page **https.//www.python.org/downloads/**

2. Run the Python Installer to install Python 3 onto your Mac.

Note. *More advanced courses use "Package Installers" like Visual Studio on Windows and HomeBrew on Mac, but in this course, we keep things as basic as we can.*

Running Python

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**, use the Windows key + r keyboard combination and type in the command “CMD” in the "Run" dialog window (Figure 4).

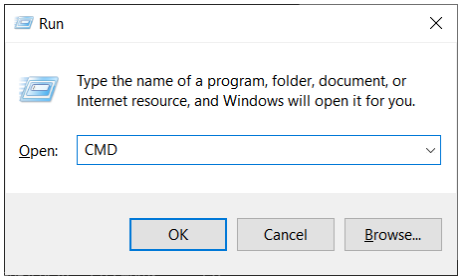

Figure 4. The Run dialog window

Clicking the OK button opens a command prompt window that looks like Figure 5.

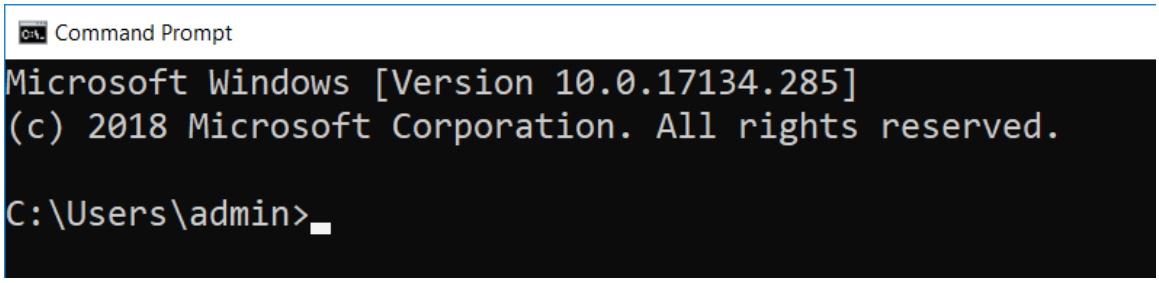

Figure 5. The Windows Command Prompt window

If you are using a **Mac**, it's almost the same, but now it is called a Terminal window. Open a terminal window using **Finder > Applications > Utilities > Terminal.app** (Figure 6).

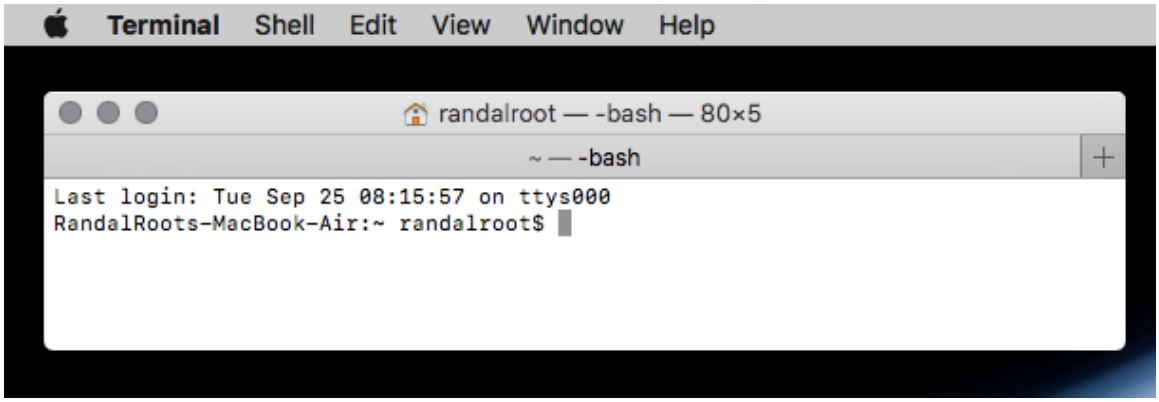

Figure 6. A Mac Command prompt

Running Python from an Editor

You can create and run python code using its **built-in editor called IDLE**. IDLE is **simple to use** software, and you can find several resources on how to use it via an internet search; **https.//www.google.com/search?q=How+to+use+Python+idle** (external site)

On Windows, the necessary steps are (Figure 7).

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

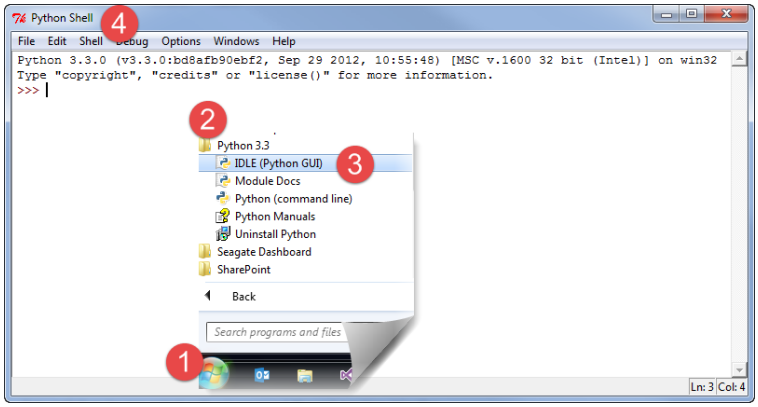

Figure 7. Steps to open the IDLE Python Shell Application

Note. Remember that on Mac, there are two versions of IDLE installed and make sure to open version 3.x (Figure 8).

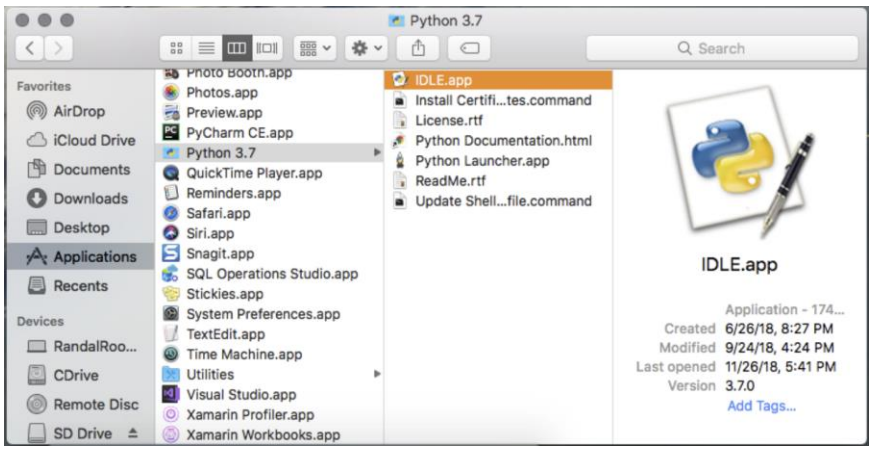

Figure 8. Steps to open the IDLE Python Shell Application on Mac

LAB 1-1. Install Python

1) Install Python on your computer as needed.

2) Open Idle and run the command, "print('test')" and note what it does.

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** the Start Menu and **type** in the command "**CMD."** Doing this opens the window and presents a flashing cursor (Figure 9).

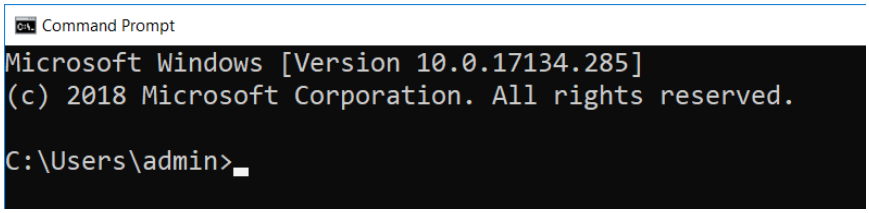

Figure 9. A Windows Command prompt

If you are using a **Mac**, it's almost the same, but now it is **called a Terminal window**.

Open a terminal window using **Finder > Applications > Utilities > Terminal.app** (Figure 10).

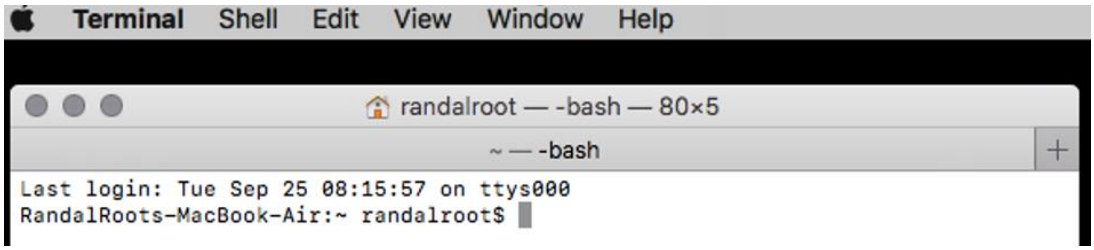

Figure 10. A Mac command prompt

Using Console Applications

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 minimal fuss, such as making your application look nice!

IPConfig.exe is an excellent 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 11) or "ifconfig" on the Mac OS (Figure 12).

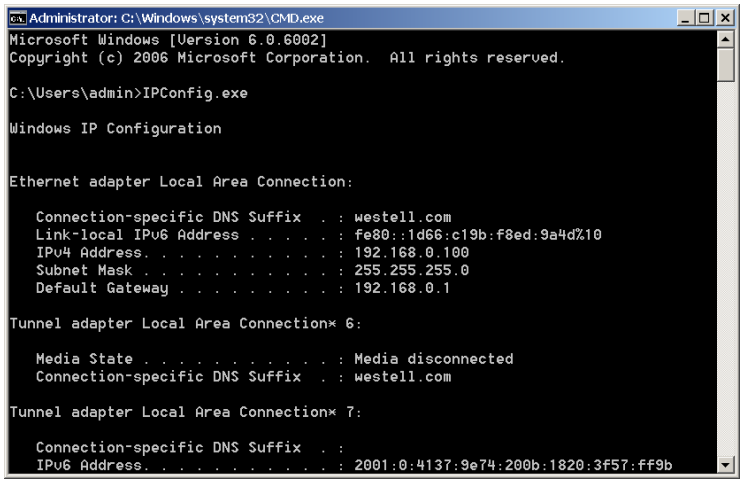

Figure 11. Results from running IpConfig.exe

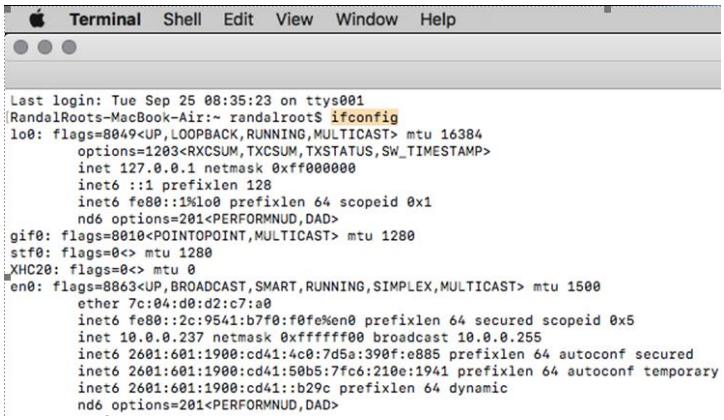

Figure 12. Results from running IpConfig.exe

LAB 1-2. Using a Console application

1) Open an command console/terminal window.

2) Run the command, "ipconfig" (or "ifconfig" on Mac)

3) Run the command, "python -?" (or "python3 -?" on Mac) and note what it does.

Python Script Files

*"In computing, the word script is used to refer to **a file containing a logical sequence of orders** or a batch processing file. This is **usually a simple program, stored in a plain text file**.*

***Scripts are always processed by some kind of interpreter**, which is responsible for **executing each command sequentially**."* (**https.//realpython.com/run-python-scripts/**, 2019)

You create a Python script **using text editor** (Figure 13).

1. **Open** a text editor.
2. **Type** in one or more commands.
3. **Save** the file with the appropriate extension (.py for Python).

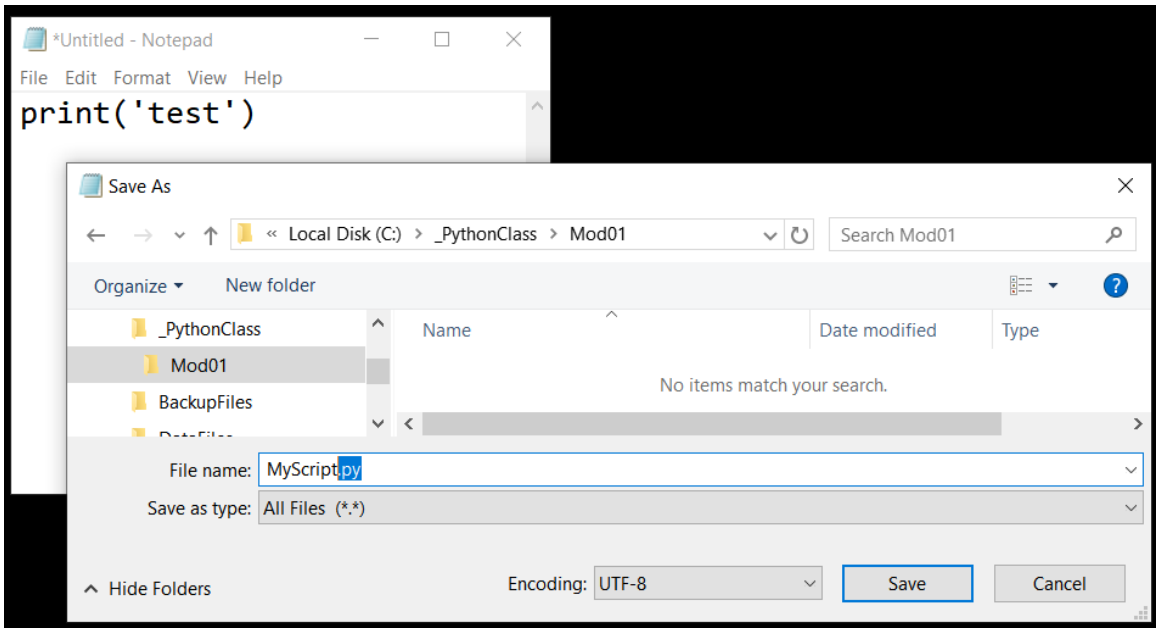

Figure 13. Creating a Python script using Notepad

***Note.** This process is the **same on Mac**, **but** you use a different editor, such as Te**xt Edit**.*

Programming Basics

Programing involves telling a computer what to do. You tell a computer what to do by **issuing commands in the form of statements**, **which perform operations**, **usually involving data**. These make up the **two standard components** of any program.

- ***Data.*** the stored information you want to work with, such as a person's name and phone number.
- ***Operations.*** the process of doing something with the data, such as printing it.

While data and operations are the core of the program, your **program's code can also include** things like *comments*, *namespaces*, *directives, and statements*.

- ***Comments.*** additional information for humans
- ***Namespaces.*** a way to organize code into named groups
- ***Directives.*** additional information to the computer, but are not directly part of the program

Statements

You type statements into a command console or a Python code file (To run later.)

- **A *statement* is one instruction** to the computer
- Each of these statements is **made up of one or more keywords or symbols** (sometimes called *tokens*)
- There can be more than one token per statement

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

You must **tell the computer when you complete each command** 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!*

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
```

Listing 1

Commenting code is **useful to add notes** to a program, making it easier to read its code.

Commenting is also useful when you want to see if disabling a set of statements solves a problem. Any code following an inline comment is not processed. Programers often ***"comment out"* a section of code temporarily**, to see if a programming error disappears. Once they **identify an error** is related to the commented of statements, they **fix the error and remove the comment** (Listing 2).

```
print('test')
#print 'test' without parenthesis only works in 2.x
```

Listing 2

Block Comments

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 (Listing 3).

```
'''
Both these statements are commented out for testing
int x = 5;
int y = 10;
'''
```

Listing 3

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 4

Case-Sensitivity

Python is a case-sensitive language, so you must be careful as you type (Listing 5).

```
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 5

Python Coding Standards ("Pythonics")

The Python community is a talkative group with strong opinions of how to write "good" code. Since this is more a Programing course than a Python course, we don't strictly adhere to these guidelines. Instead, **we focus on making the code easier to read for beginning programmers by using descriptive names along with upper- and lower-case lettering.**

Still, you should know that Python.org publishes a guideline to help with these discussions. **https.//www.python.org/dev/peps/pep-0008/**

Common Functions

Python includes **many built-in functions** that you can use to perform actions in your program. **We won't use all of them**, but we learn to use many! Let's start with two of the most commonly used ones, print() and input().

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**. **In this module, we use it to stop the command window from closing** once a script finishes (Figure 14).

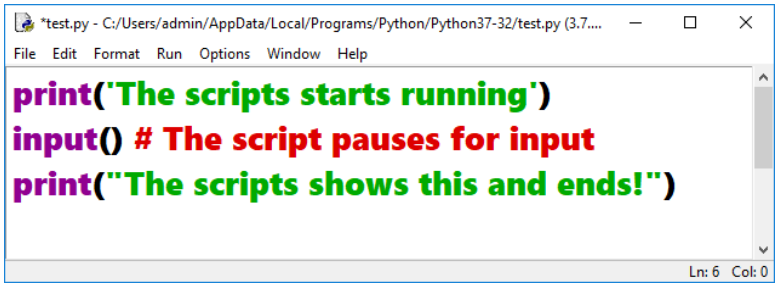

Figure 14. A Python script using the input function

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

Sometimes, running a script file causes a command window to open, run the code, and immediately close once the script's code finishes. Adding the input() function forces it to pause and waits for the user to press the Enter key (Figure 15).

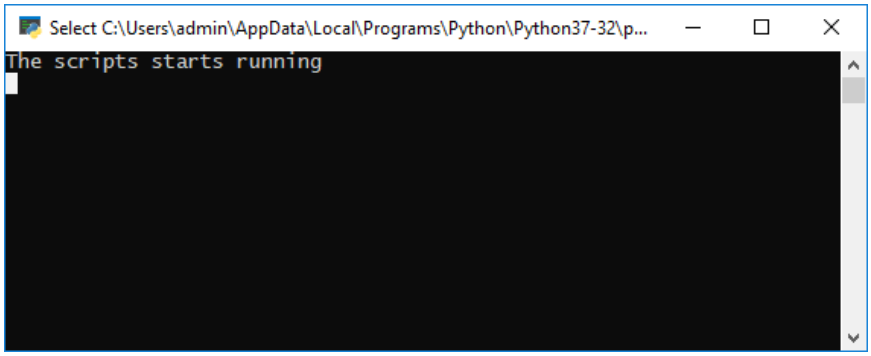

Figure 15. The results of the script file while paused

You can test this "pausing" behavior by running a Python script from Windows Explorer with and without the input function (Figure 16).

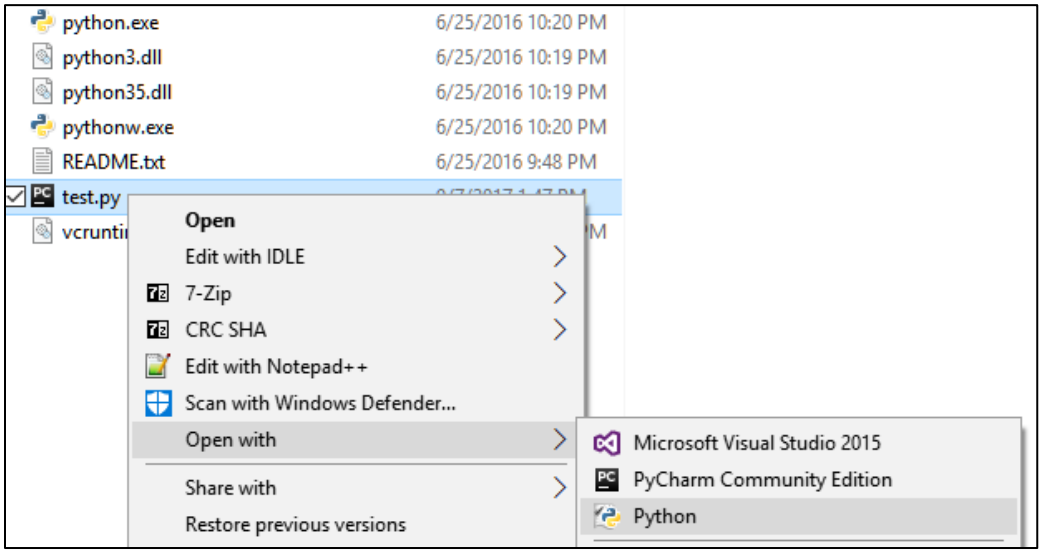

Figure 16. Running a Python script file from Windows Explorer

The input function is to capture the user's "input," as we will see in the next lab!

LAB 1-3. Create a Console Application

1) Create a new python script file in Idle called Lab1-1.py

2) Add the code shown in Figure 17 to the file.

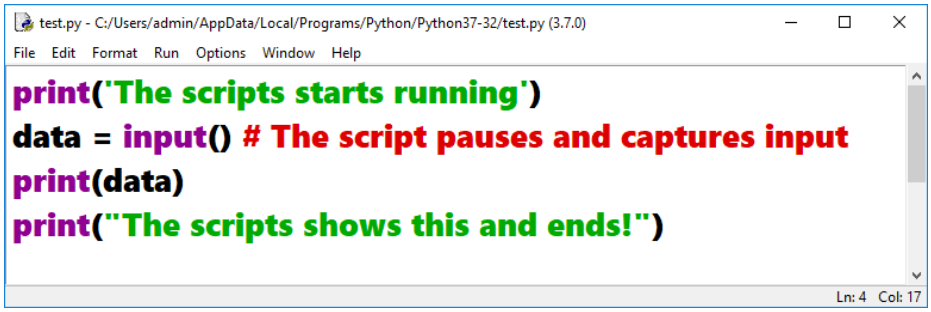

Figure 17. The code of the Lab 1-1 script file

3) Run the file and note what it does.

Custom Functions

Programmers have found it a good practice to organize code into groups of statements. These groups of statements are often called functions, but also known as methods, procedures, sub-procedures, and other names.

You create a function in Python by defining it using the "def" command followed by the name of your new function. The name can be almost anything but should represent the task and purpose of the statements within.

After you create a function, you can run its group of statements by *calling* the method which runs all the statements inside the function (Listing 6).

```
def demo_function().#Define a function
  print("This is a statement in DemoMethod")
  print("This is another statement in DemoMethod")
#End demo_function

demo_function() #Call the function
```

Listing 6

Functions must be defined first and then called, so Python developers often write all the functions at the beginning of a script file.

Code in a function is processed one statement at a time, starting at the "top," or first statement in 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 through the indentation of its code. Any statements indented under the function’s definition statement are considered part of the function.

Tip. You can place a comment at the bottom of the function if you want to make the end more obvious.

The Main Body of a Script

Many languages use a **special "Main()" function** to indicate where code starts running when a program or script starts. 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 processes one line after the other from the top of the script to the bottom by the script interpreter, Python.exe.

However, if the script interpreter finds a function definition at the top of the script it loads its code into memory, but does not run that code until it is called latter (often from the Main script body).

For example, when you call a function from the "Main" body, it jumps to that method, runs the statements inside of the called method, and returns to the Main body once the function's code is done. The example below outlines the order in which statements are processed (Figure 18).

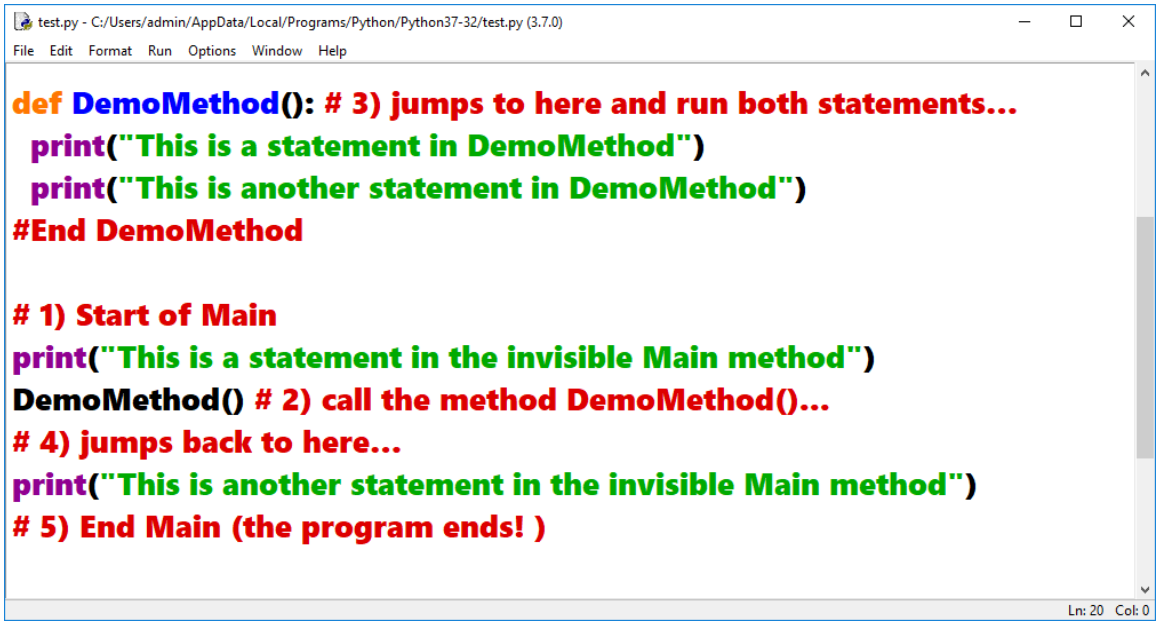

Figure 18. A script file with a custom function

LAB 1-4. Create a Console Application

1) Create a new python script file in Idle called Lab1-2.py

2) Add the code shown in Figure 18 to the file.

3) Run the file and note what it does.

Script Headers

Whenever you create a script on your computer, it should **include a header the describes that it does and what changes** have occurred to the script over time.

Here is an example of a “Script Header” included at the top of a script (Figure 14)

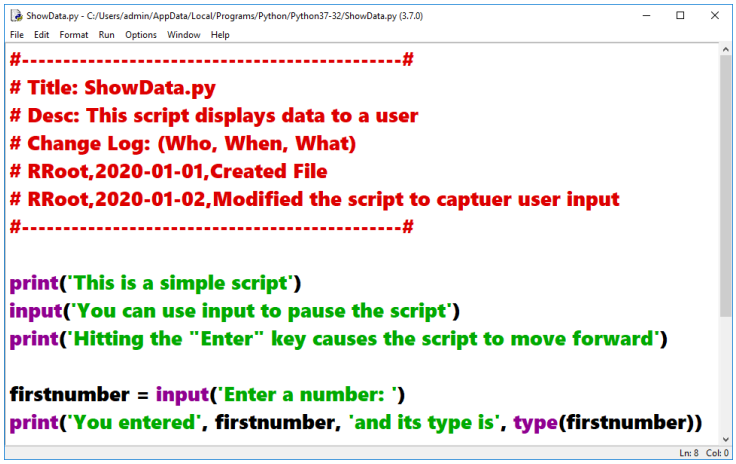

Figure 14. The results of the script file while paused

Summary

In this module, we looked at what the Python language is, how to install it and how to use it. We also covered the different components that make up a Python script.

At this point, you should try and answer as many of the following questions as you can from memory, then review the subjects that you cannot. Imagine being asked these questions by a co-worker or in an interview to connect this learning with aspects of your life.

- What is Python?
- How is it installed?
- How do you run a Python program?
- What are Console applications?
- What are Python script files?
- What are the basics of programming?
- What are Python coding standards?
- Name some common Python functions?
- How do you create custom functions?
- What is the "Main" body of a script?
- What is a Script Header?

When you can answer all of these from memory, it is time to complete the module's assignment and move on to the next module.