Chapter 1

CITC 1301 - A70

D. Blair - Northeast State Community College

Introduction

Python is a mature and robust language. It is also good for learning to program, creating applications that solve problems. In this class, you will learn many exciting ways to interact with computers using software. First, let's define a few parameters we need to understand and work through to be a good programmer.

You don't need to major in programming in order to build programs. Many times folks from different backgrounds that work in different environments need computers to solve problems. Almost anyone can become a programmer with enough effort and practice. So if you are a networking major, systems administrator, mathematician, biologist, chemist, or any number of other occupations, knowing how to program will help you understand your profession better. And, if you are a software engineer, the programming is an absolute necessity to understand thoroughly. More than anything else, learning how to program builds good problem solving skills.

The fundamental question of computer science asks: what can be computed? While computers are at the core of computer science, it is not what computer scientist study. It is like a carpenter has a box of tools that are used to build things. The carpenter knows her tools, but needs a project to work on for the tools to be useful. Much like a carpenter, a programmer needs a computer to build things, but without problems to solve, a computer is useless. So a programmer looks to the world and says: what can be computed?

For everything that can be computed, programmers must solve how to make a computer language to what he need it to do. Sometimes this can be very complex. While the language specifics may be relatively simple, the algorithms are not. What is an algorithm? Imagine yourself making a 5 course meal. All the recipes are complicated and require you to follow the recipes to the tee. The recipes are the algorithms required to make the meal work and taste the way they are meant to taste. If you follow those algorithms next time, the meal should taste the same way.

From time to time, a problem may be so difficult to solve that it is simply to practical to even attempt. Those problems are consider to be intractable. Usually this is caused by lack of resources. Extremely difficult problems may not be economical to solve, for example. It doesn't mean they don't have a solution; instead, it would cost the company too much money to solve the problem. The company may not have the staff that is trained well enough to tackle these problems. In these cases, alternative solutions could be a better option.

Hardware

Just like a carpenter that works on many projects, a programmer needs to know enough about a computer to do the work needed. Understanding the underlying principles of the computer will help you master the steps to go through to put a program into action. It is just like understanding a little about an internal combustion engine helps you maintain your car. You know how to use the accelerator to go faster, out gas in to keep going, and use the brake to stop.

- CPU - central processing unit is the brains of the machine. This is where the basic operations of the computer is carried out. The CPU follows the process of fetch-decode-execute many times per second. All CPU instructions are executed from fast memory call random access memory (RAM). So before the computer can execute any code that is on the disk, it must first be loaded into RAM.
- RAM - random access memory is fast but it is also volatile. Any data that a program is working on that resides in RAM will go away if the computer power is lost before saving that data to permanent storage such as a disk drive. So let's say a program asks you to enter 3 tests scores to average and it calculates the average, unless the program saves that information to a disk drive where a database might reside, then it will be lost when to program ends.
- Input devices - this can be something physical, such as a keyboard, mouse, USB device, or disk drive; but can also be data from a remote database or input from a temperature sensor.
- Output devices - output devices include disk drives, monitors, printers, and USB devices. They can also be a remote database or an RFID writer; for example.

Programming Languages

Programming languages such as Python, PHP, C++, Java, etc. all abstract the complexities of the hardware away from the programmer to make it easier to write software more efficiently with fewer bugs. Programming languages are, however, exact. The syntax must be followed or the program will not compile and run. They all share the property of having well-defined, unambiguous syntax and semantics.

Natural languages like what we speak are not exact and many times can carry many meanings for the same words. This is why natural languages can not be used by programmers to write software.

All programming languages must be compiled down to machine language before they can be executed by the machine. Machine language is the only thing a computer can understand. Machine language is binary, and is a series of 1's and 0's. It is almost impossible for humans to look at machine language and make any sense out of it. This is why we use high-level languages such as Python to code.

There are two ways that high-level languages get turned into machine language. One way is to be compiled all the way down to 1's and 0's by a compiler. This translation is complicated and may require several steps before reaching machine-specific machine language. The second way is through an interpreter. An interpreter is a program that compiles the source code only partially, into object code. Then when it is executed, it requires another application that translates it on down to machine code. This is often called a virtual machine and makes the object code portable. So as long as there is the program installed on a system that runs the object code and interprets it, then the same object code can be compiled once but run anywhere, so to speak. Python is interpreted.

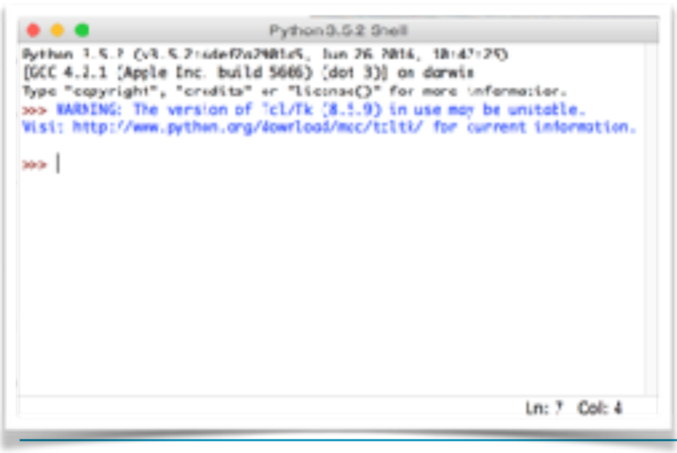
Python

Before you can create and run Python, you must first install it on your machine. You can find the Python download at:

https://www.python.org

Make sure and get version 3.x but you can run both 2.x and 3.x on your computer at the same time. We will discuss how to use the different version later in the text. There are flavors of Python for Windows, Linux, and Mac. I test all the code if this course in all three environments to make sure they all behave the way they are supposed to behave.

There are several good development environments that we will discuss. Python comes with an environment call IDLE shell.

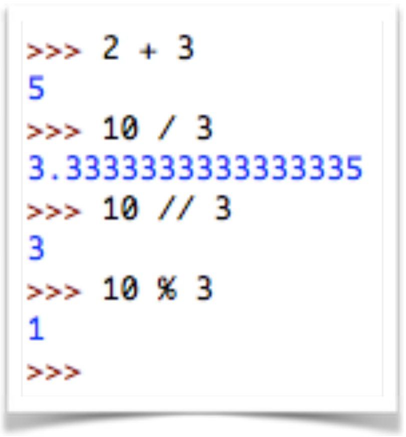


```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 26 2016, 10:47:25)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/mac/tcltk/ for current information.

>>> |
```

It is not very good to create large programs with, but it is useful for trying one-liners. I will explain.

Let’s look at some code. Try some of these:



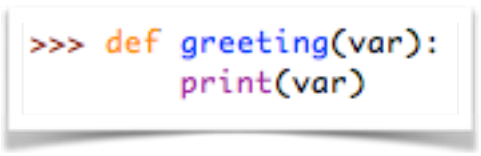
```
>>> 2 + 3
5
>>> 10 / 3
3.3333333333333335
>>> 10 // 3
3
>>> 10 % 3
1
>>>
```

Notice that dividing 10 / 3 results in a “floating point” number.

To get integer division, you use two forward slashes together.

To find just the whole number remainder from division, use the percent sign.

We will get back to math a bit later. First, let’s see how we can execute more than one line of code at a time. Try this:



```
>>> def greeting(var):
        print(var)
```

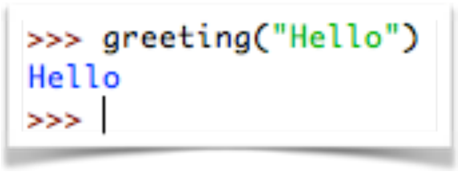
There are only two lines of code here, but there is a lot to explain about what is going on.

First line: **def greeting(var):**

This tells python the you are going to **def**ine a block of code called **greeting**. The name is not important. It could be anything you want to use but you don’t want to clash with a keyword that Python has already defined. Then there is an open and closed parentheses with **var** inside. Here again, var could be any name like “myVar1” or “words” etc. Here, var is what is referred to as a parameter. More on this later, but consider it a place holder for something you will pass it later. Then there is a colon at the end (:). This colon defines a block of code known as a scope. The block of code, or scope, is defined by each line that follows that is tabbed over one table. IDLE automatically tabs over one tab for you but in a development environment, you may need to do this manually.

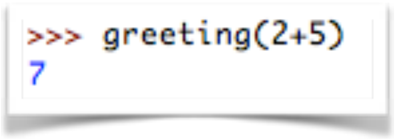
The line that is tabbed in one tab, **just under the definition statement is a print statement**. So, there is a single line of code contained within the scope of this **def**inition (or **function**) that is a print statement. A **statement** is a complete computer command. The print statement tells to computer to print to the console windows, in this case IDLE, the contents of the parameter var. To end the function definition we just created, all that needs to be done is hit the enter key and the cursor goes back to the left margin. So the block of code, or scope of the function, is defined by the one tab indent from the left margin. I could have added any number of other statements to the scope of the function by indenting them one tab stop from the left. We will see many examples of that very soon.

The question now becomes, how do we execute this simple block of code? All that needs to be done is to type the name of the function along with data that goes into the var parameter, like this:



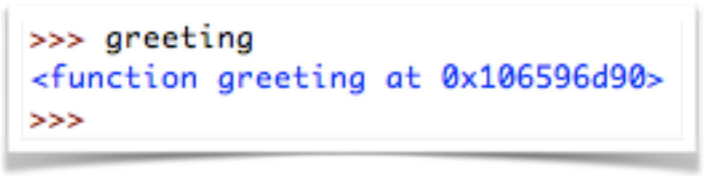
```
>>> greeting("Hello")
Hello
>>> |
```

You just type in the name of the function (**greeting**) along with the string literal "Hello" in this case, and hit enter. The string "Hello" gets passed to the function via the var parameter and then can be inserted into the print statement. If you haven't already tried this, try it now. Play around with passing in different things. You can even pass in something like: 2+5



```
>>> greeting(2+5)
7
```

Note: if you only type the name of the function without the parentheses, you will get function name as Python sees it in memory. So if you see this, you know you forgot to enter the parentheses.



```
>>> greeting
<function greeting at 0x106596d90>
>>>
```

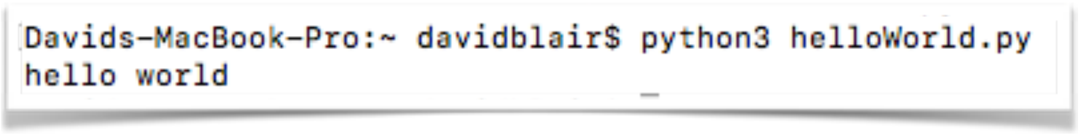
A Python Program

Let's continue on with multiline programs. This is where we need to leave IDLE and move to an IDE (Integrated Development Environment). There are many IDE on the market. Some are free and other are very expensive. At a minimum, you can use Notepad++ for Windows, VIM for MAC and Linux. Another good choice is PyCharm by Jetbrains which works on all these platforms and what we will use in this course. It is free to students but has a bit of a learning curve to use.

Lets use a plain text editor (Notepad or vim for example) and create our first python file. I will use vim, but most of you will use Notepad.

- Create a new text document and call it **helloWorld.py**
- Type the following on the first line: print("hello world")
- Save and open a command shell or window. (Windows, search for "cmd" and enter)

- Move to where you saved the helloWorld.py file. (I will go over how to do this in class)
- Then type in: **python3 helloWorld.py**
- You should see the following:



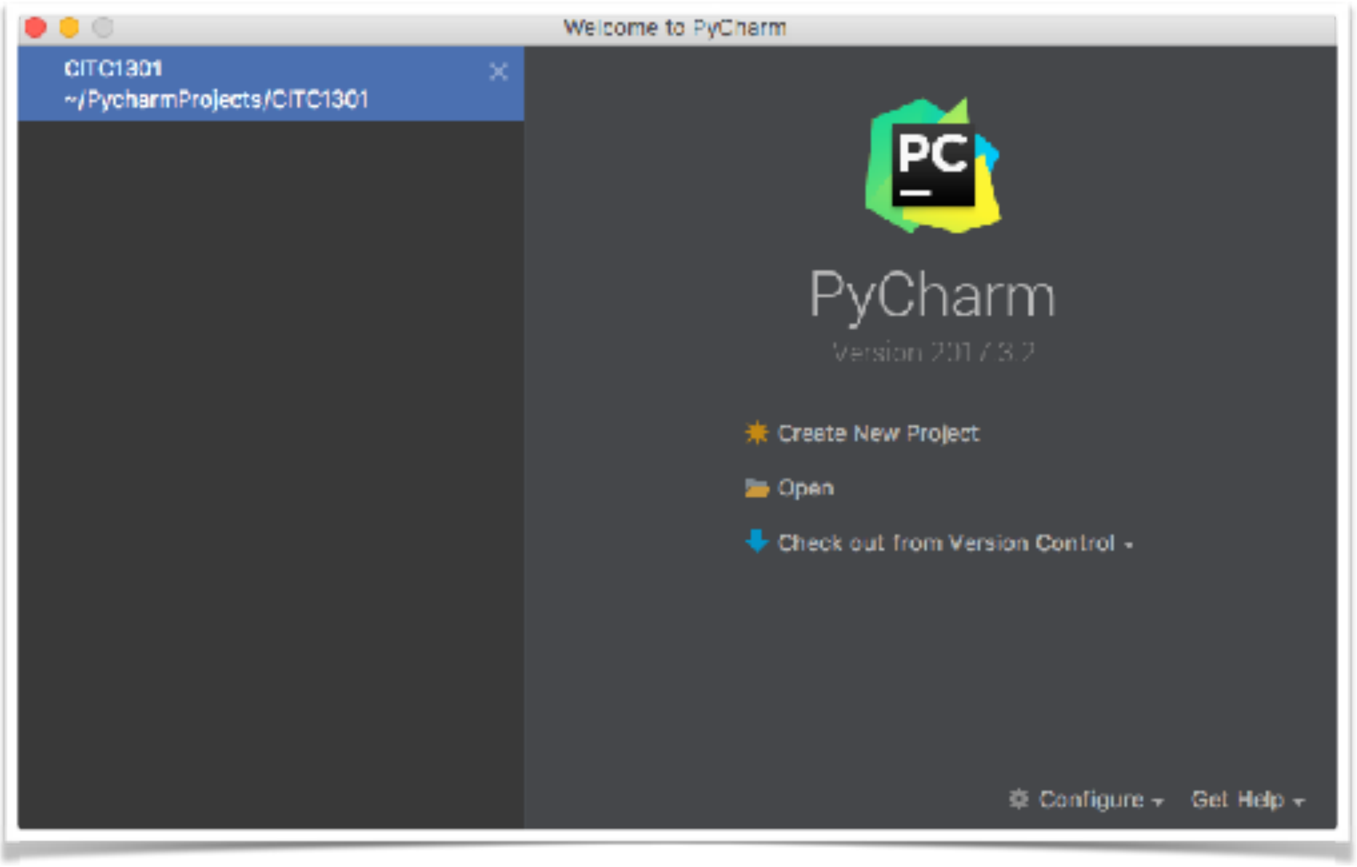
```
Davids-MacBook-Pro:~ davidblair$ python3 helloWorld.py
hello world
```

Now, lets move on to an environment that is a little more robust.

PyCharm

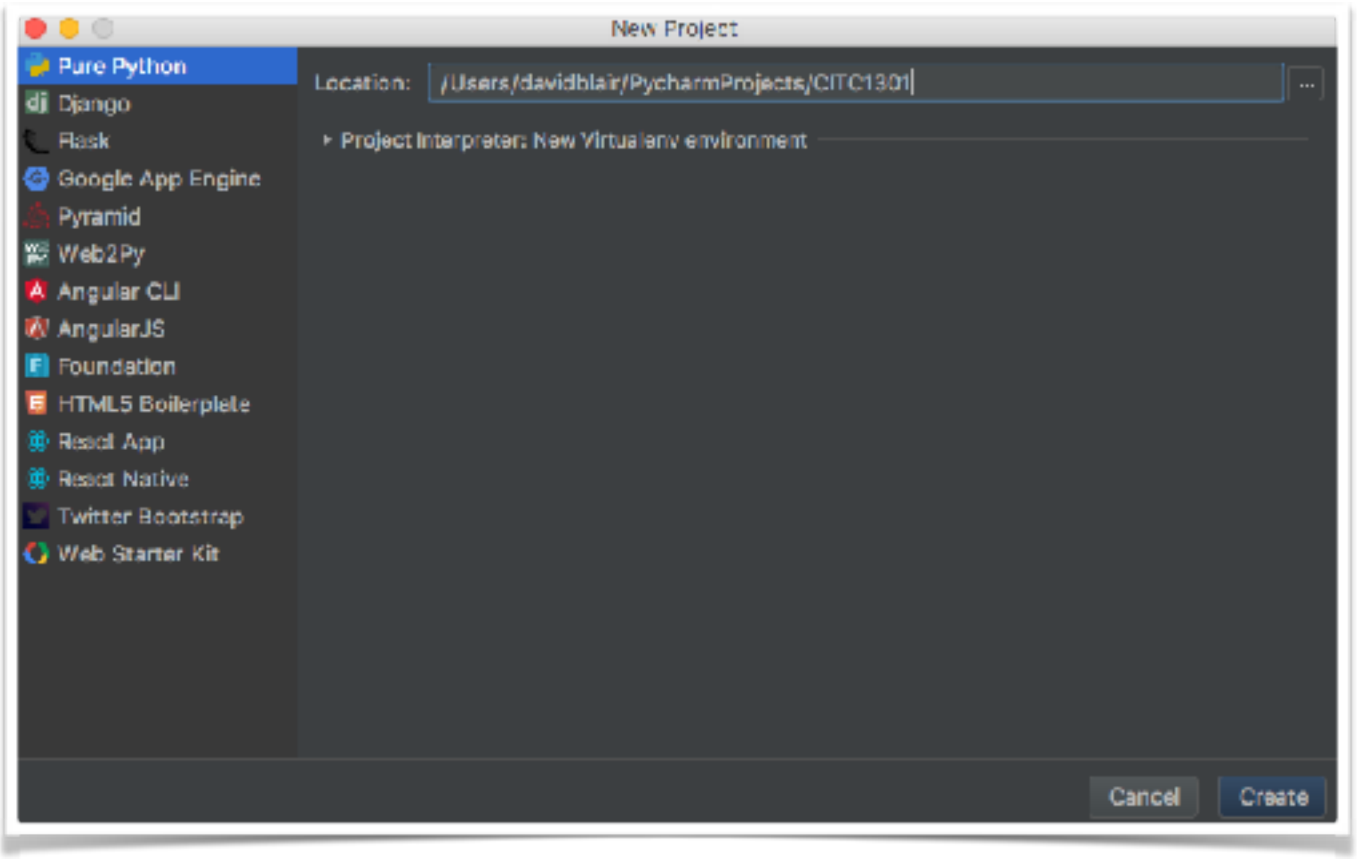
You will need to download pycharm and install it on your personal system; pycharm is already installed in the classroom. Here is the download link for pycharm; you will need to supply JetBrains with your Northeast email address to get the professional version for free. https://www.jetbrains.com/pycharm/

Once you have it installed, then start it up for the first time.



As you can see from the image above, I already have a project created called CITC1301. This is what you will want to do. So, create a new project.

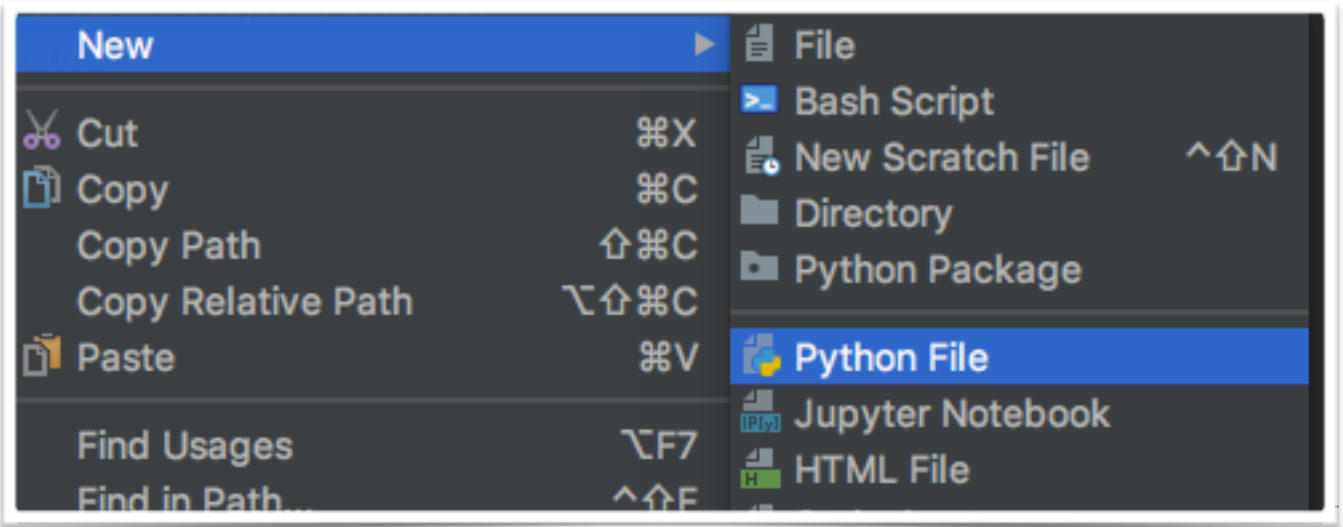
I keep my projects in PycharmProject folder, but you can use another one if you like. Add the name of the project at the end: CITC1301. Also make sure and leave Pure Python selected in the left window pane.



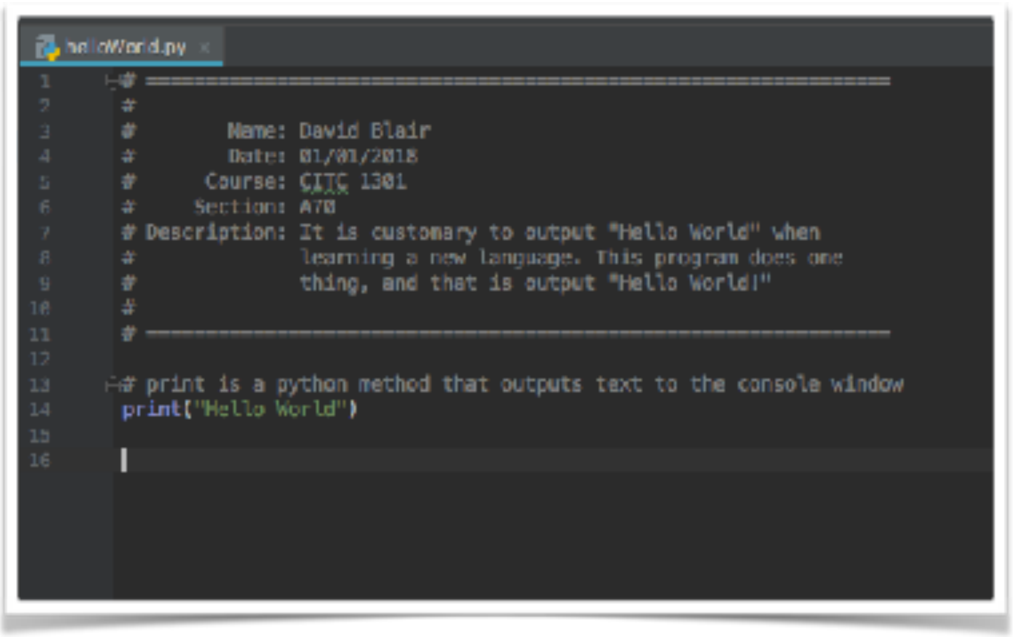
Location: /Users/davidblair/PycharmProjects/CITC1301

▸ Project Interpreter: New Virtualenv environment

Now right click on the project CITC1301 and select New, Python File (see below)



In the design window, you can start typing. Type in the following code:

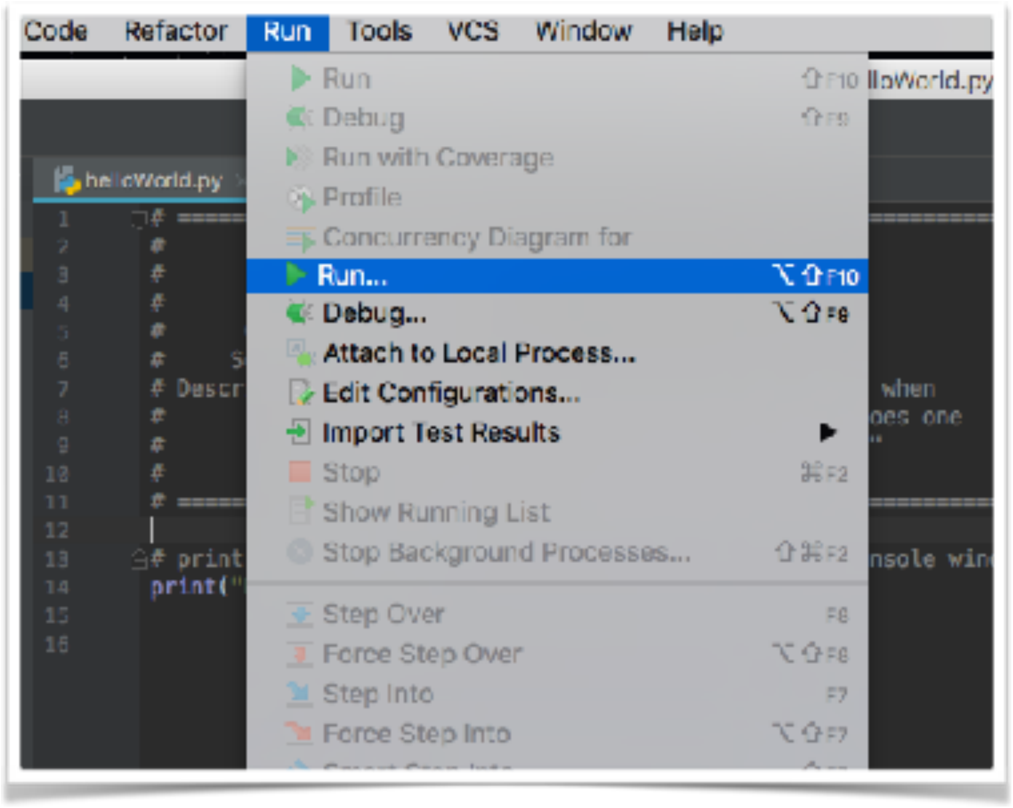


```
# ============================================================
#
#        Name: David Blair
#        Date: 01/01/2018
#      Course: CITC 1301
#     Section: A70
# Description: It is customary to output "Hello World" when
#              learning a new language. This program does one
#              thing, and that is output "Hello World!"
#
# ------------------------------------------------------------

# print is a python method that outputs text to the console window
print("Hello World")

```

Select from the Run menu, Run…



You should see your new program listed, select helloWorld. Execute and see the results.

On the chapter 2. There we will do lots of cool stuff with python!