Lab 12: Python

What is Python?

Python is a high-level programming language that was first introduced in 1991. The most current version is Python3, which was introduced in 2008, but many systems still rely on code that was written using the slightly different Python2.

Most developers think that Python is a much easier language to work in than C++ is. The same task will usually require far fewer statements in Python than it does in C++. The tradeoff is that programmers have less control over the machine in Python than in C++.

Python shares some of its syntax with C++ but its execution is very different because Python is an interpreted language rather than a compiled language. Remember that compiling is the process that replaces all of the high-level statements in a program with executable machine code. Interpreted languages are only read one line at a time. The interpreter translates and executes one line at a time. A Python script that contains a syntax error will continue to run until the interpreter encounters that error. This can make debugging Python a little easier than a compiled program, where it is not always clear where an error occurred.

Variable in Python are not given an exact datatype. The interpreter assumes the datatype based on the value that is used to initialize the variable. Another thing that makes Python look very different from C++ is that code does not have to be written in a function. The example below is a complete python script that will create a string variable named **greeting** and then print the value of that variable on the command line.

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

Libraries

Python has become very popular because of the large number of libraries available that make complex tasks easily accessible. Some of the most valuable libraries for computer scientists include:

- **matplotlib**: a tool for creating charts and graphs.
- **numpy:** a tool for processing numerical arrays.
- **tensorflow:** a package created by Google for building machine learning models using multi-dimensioned collections of values called tensors.
- **scikitlearn:** a package for training and testing simple machine learning models and other tools for data science. Also includes a large collection of easily accessible datasets.
- **pandas:** a tool for creating convenient tables.
- **scipy:** a collection of tools for processing scientific data
- **pygame:** a package for creating simple games using the Python language
- **openCV:** a suite of tools for supporting computer vision

All of these libraries need to be installed on the computer that is running the Python script that needs them. This can be done from the command line using a tool call **pip** or **conda.** Developers who prefer a graphical tool for package (library) magagement can use **anaconda.**

The following script will create a graph of the function $y=x^2$ for a range of x from 0 to 100. Take a moment to consider how many lines of C++ would be required for this task.

```
import numpy
from matplotlib import pyplot as plt
x = np.array(range(0,100))
y = x*x
plt.plot(x,y)
plt.show()
```

Python Syntax

Programmers who are familiar with C++ usually have a fairly easy time learning Python but there are some very important differences that have to be kept in mind. Some of these differences are:

- Whitespace rather than curly brackets is used to show that code is in a particular block. This means that inconsistent indenting will produce code that does not work.
- Variables are not given a datatype explicitly.
- Functions are created using the **def** keyword. This is because functions, like variables, do not have a specified return type.
- Functions can return more than one value.
- There is not **switch** statement. Also **else if** is shortened to **elif.**
- The **cout** statement is replaced with **print**, which is a function that can take any number of strings as arguments.
- Function arguments can be passed out of order. This is done by writing the parameter name and using the assignment operator =. So in Python both of these function calls are legal and will do the same thing:
  - foo(x=0, y=1)
  - foo(y=1, x=0)
- Code does not have to be written in functions.
- The **#include** directive is replaced with the **import** statement, which does not have to be at the top of a script and can be called at any time.
- Comments are separated from regular code by starting a line with # rather than // or **/*...*/**

Have a look at the short script below that defines and uses a simple function.

```
import numpy as np

#This function return the biggest and smallest numbers in an array
def maxAndMin(x):
        min = np.min(x)
        max = np.max(x)
        return max, min

someNumbers = [1, 3, 7, 5, 9, 10, 11, 12, 3, 4, 7, 2]
biggest, smallest = maxAndMin(someNumbers)
print("The biggest number was", biggest)
```

Drawbacks of Python

Although Python is much easier to use and allows developers to focus on a task like data science or machine learning without having to worry to much about how the actual computation are taking place, it does have some drawbacks. The programmer has very little ability to manage the physical memory in Python so it is very difficult to use C-style optimizations when writing a Python script. Interpreted languages also tend to have longer run times than compiled languages do. This the interpreter has to do the work of translating code to machine language at runtime rather than being able to do all of that work before hand.

There are some tools to help Python run more efficiently, but ultimately Python is not the correct choice of language for writing optimized, high-efficiency programs. Rather Python is useful for writing programs when the run time is not really a consideration. Commercial software is usually not written in Python because of this fact, and also because scripted languages expose the source code to the end user which is bad for commercial code.

Engineers, data scientists, mathematicians, and other scientists whose focus is not producing high-speed software packages will get the most benefit out of Python.

Students who are interested in learning more about Python can visit:

- https://www.w3schools.com/python/
- https://docs.python.org/3/tutorial/
- https://www.codecademy.com/learn/learn-python-3