QUTTIC: Workshop
Semester 2 2018

Introduction to Python for Finance

Beginner to Intermediate

Name: Andrew Collison
Email: collison@qut.edu

Contents

1 What Is Python and Why Do We Use It

1.1 Introduction

Python is a multipurpose programming language that has an emphasis on readability, syntax and accomplishing tasks in fewer lines of code. Python is an object oriented language with inbuilt memory management, thus small python projects can be scaled into large projects relatively easily.

1.2 Object Oriented Programming

Object oriented programming is using "Objects" such as classes and functions to manipulate and control data. This is a key part of the python programming language as it allows developers to write modules that can be used throughout many projects and makes the design and scalling of projects much easier.

1.3 Applications

Python has grown to be one of the most popular programming languages in recent times due mostly to it's simplicity and small learning curve. Python can be applied to everything from cybersecurity, robotics, machine learning, automation and finance.

2 Getting Started

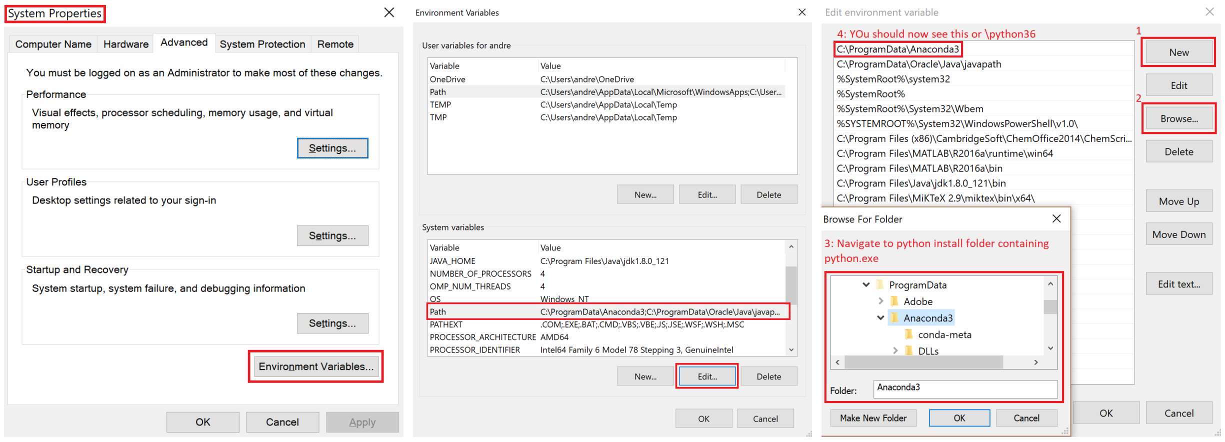
2.1 Installation

Python can be installed in two ways, the most popular method is directly downloading the latest version from python.org. Alternatively there is the Anaconda python distribution that includes a large number of python packages.

1. Downloading
   As of writing this the most recent python version is python 3.6.4 and is available at
   https://www.python.org/downloads/release/python-364/
   Select the download version that is applicable to your computer.
   The Anaconda install is available at https://anaconda.org/

2. Installing
   Installation is a simple process and it is best to go with the default settings.
   First hand experience: Do not tick ”add to path” on the install menu.

3. Adding python to the system path



You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your sign-in

System startup, system failure, and debugging information

Variable	Value
OneDrive	C:\Users\andre\OneDrive
Path	C:\Users\andre\AppData\Local\Microsoft\WindowsApps;C:\User...
TEMP	C:\Users\andre\AppData\Local\Temp
TMP	C:\Users\andre\AppData\Local\Temp

Variable	Value
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_121
NUMBER_OF_PROCESSORS	4
OMP_NUM_THREADS	4
OS	Windows_NT
Path	C:\ProgramData\Anaconda3;C:\ProgramData\Oracle\Java\javap...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 78 Stepping 3, GenuineIntel

4: YOu should now see this or \python36

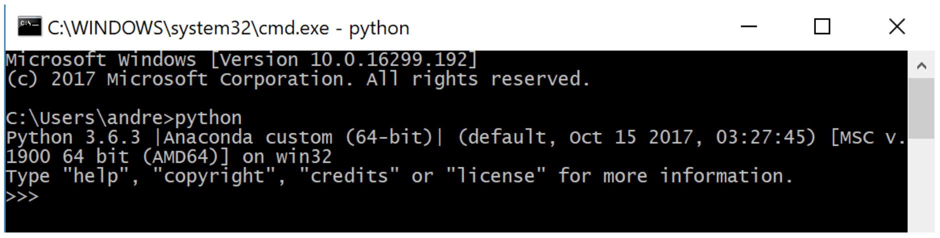
```
C:\ProgramData\Anaconda3
C:\ProgramData\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
C:\Program Files (x86)\CambridgeSoft\ChemOffice2014\ChemScri...
C:\Program Files\MATLAB\R2016a\runtime\win64
C:\Program Files\MATLAB\R2016a\bin
C:\Program Files\Java\jdk1.8.0_121\bin
C:\Program Files\MiKTeX 2.9\miktex\bin\x64\
```

3: Navigate to python install folder containing python.exe

Folder: Anaconda3

Figure 1: Adding Python to the system path

4. Testing install After adding to the system path you should now be able to execute python scripts from the command prompt. A new command prompt can be opened by pressing windows + R and typing cmd.exe. Once in this window, typing ”python” and pressing enter should produce the following or similar output.



```
C:\WINDOWS\system32\cmd.exe - python
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\andre>python
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct 15 2017, 03:27:45) [MSC v.
1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 2: Confirming that python is correctly installed

2.2 Choosing an Interactive Development Environment (IDE)

An interactive development environment is where you write your python scripts. There are a number of excellent IDEs available but the two most common and easy to use are PyCharm (recommended) and the ol faithful NotePad++. PyCharm is managed by Jetbrains and is an excellent development environment that is easy to use and set up. There is a free community version of PyCharm available.

2.3 Your First Python Script

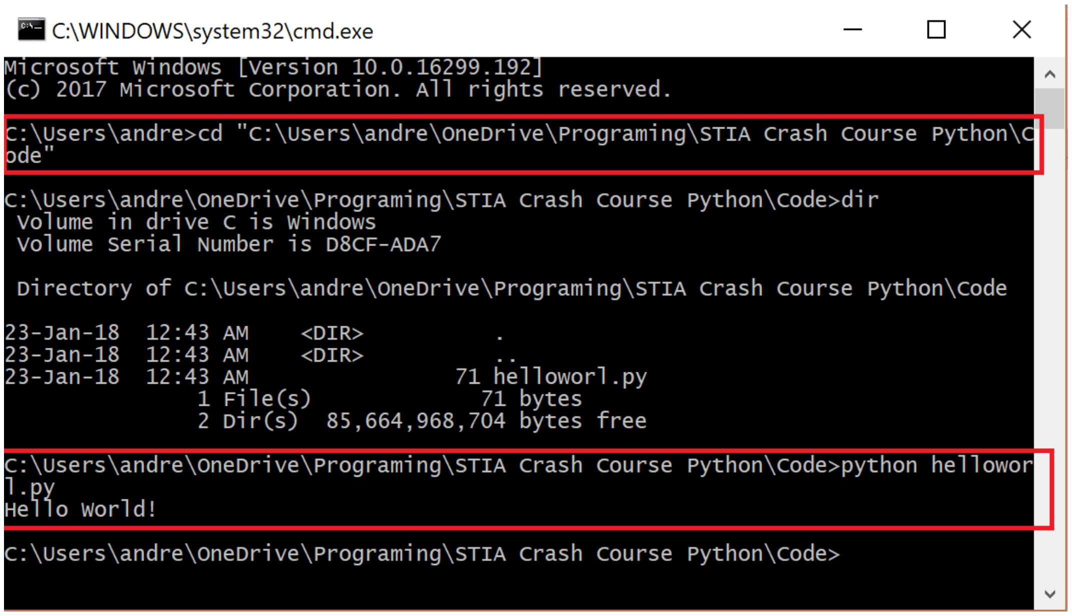
It is a requirement as a new developer that your first script be the famous "Hello World!" program, an incredibly sophisticated program.

1. Opening a new script file.
   This is easily done by creating a new text file and naming it "helloworld.py" (remove the .txt from the end). This is now a script that is executable by the python compiler. Open this file using your IDE of choice.

2. Editing the script file and printing to the terminal.
   Inside your IDE write the following code.

```python
x = "Hello World!" # Variable to be printed
print(x) # Print variable to terminal
```

3. Running the script.
   If you are in the PyCharm IDE the script can be run by simpling clicking on the "Run" button or pressing ctrl+shft+F10.
   However the OG method for running python is to run the script through the command window. This is done by opening a new command prompt window and using "cd" to navigate to the directory containing the "helloworl.py" script file. typing python followed by the file name will execute the script.



```
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\andre>cd "C:\Users\andre\OneDrive\Programing\STIA Crash Course Python\C
ode"

C:\Users\andre\OneDrive\Programing\STIA Crash Course Python\Code>dir
 Volume in drive C is Windows
 Volume Serial Number is D8CF-ADA7

 Directory of C:\Users\andre\OneDrive\Programing\STIA Crash Course Python\Code

23-Jan-18  12:43 AM    <DIR>          .
23-Jan-18  12:43 AM    <DIR>          ..
23-Jan-18  12:43 AM                71 helloworl.py
               1 File(s)             71 bytes
               2 Dir(s)  85,664,968,704 bytes free

C:\Users\andre\OneDrive\Programing\STIA Crash Course Python\Code>python hellowor
l.py
Hello World!

C:\Users\andre\OneDrive\Programing\STIA Crash Course Python\Code>
```

Figure 3: First python script "Hello World!"

3 Python Basics

3.1 Variables and Data Types

In python data is assigned to variables. These variables can be called throughout the program to be used or updated as the program progresses. Variables are assigned as follows

```
x = 1 # integer
y = "STIA" # string
z = 3.14 # float
# Alternatively
x, y, z = 1, "STIA", 3.14
```

These variables are assigned as either numeric or alphabetic (string) datatypes.

Table 1: Numeric Datatypes in Python

Data Type	Format	Description
Int	x = 10	Signed Integer
Long	x = 345L	Long integers
Float	x = 21.9	Floating Point real values
Complex	x = 3.14J	Contains integers in the range of 0-255

Strings are denoted using quotation marks, "xxxx". Tuples are similar to strings in that they are list of letters, however tuples can not be changed in length after they have been assigned.

3.2 Basic Operations

The basic operations in python are form the basis of any data manipulation and calculations that will be performed. Most notable is the "=" and "==" characters. Where a single "=" symbol is to assign a variable a value a double "==" is returns a True of False value based on the two variables being equal to each other.

Table 2: Python Basic Operations

Operation	Symbol	Description
is assigned to	=	assigns a value to a variable
is equal to	==	variables are equal to each other
is not equal to	!=	variables are not equal to each other
addition	+	adds two values together
subtraction	-	subtracts two values from each other
multiplication	*	multiplies two values together
division	/	divide two variables
greater than	>	greater than a certain value
less than	<	less than a certain value
greater than or equal	>=	greater than or equal to a certain value
less than or equal	<=	less than or equal to a certain value

3.3 Lists, Tuples and Dictionaries

Variables can take on a number of values. This is done in the form of arrays, lists and dictionaries. These are methods of storing values and forms the backbone of any python program.

3.3.1 Lists

Python lists are the simplest method for data storage in python. python lists are defined by "[x1, x2, x3]". Python lists are indexed at 0, so the values x1 is in position 0 where x2 is in position 1, this is a key point to remember as it is important for list indexing during iterations and modifying the list.

```
x = [] #empty list
x = [1, 2, 3, 4, "Andrew"] # list of values
```

There are a number of operations that can be performed on lists

- .append(x): adds an item to the end of a list.
- .insert(i, x): Insert an item at a given position. The first argument is the index of the element before which to insert the value.
- .remove(x) Remove the first item from the list whose value is x
- .clear(): remove all items from the list
- .index(x): Return zero-based index in the list of the first item whose value is x.
- .count(x): return the number of times x appears in the list.
- .sort(key = none, reverse = false): Sort the items of the list in place (the arguments can be used for sort customization.
- .reverse() reverses the elements of the list.
- del : the del statement can remove an item from a list given its index. listname[x], will remove an item from the list at location x.

```
>>> fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
>>> fruits.count('apple')
2
>>> fruits.count('tangerine')
0
>>> fruits.index('banana')
3
>>> fruits.index('banana', 4)  # Find next banana starting a position 4
6
>>> fruits.reverse()
>>> fruits
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange']
>>> fruits.append('grape')
>>> fruits
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange', 'grape']
>>> fruits.sort()
>>> fruits
['apple', 'apple', 'banana', 'banana', 'grape', 'kiwi', 'orange', 'pear']
>>> fruits.pop()
'pear'
```

These operations become crucial later on when iterating over lists and manipulating data.
Lists can also be made into two dimensional arrays as follows x = [[1,2,3],[4,5,6]]

3.3.2 Tuples and Sequences

Although tuples and sequences are not as widely used they are still usefull to understand when working with date-time data. A tuple consists of a number of values separated by commas.

```
>>> t = 12345, 54321, 'hello!'
>>> t[0]
12345
>>> t
(12345, 54321, 'hello!')
>>> # Tuples may be nested:
... u = t, (1, 2, 3, 4, 5)
>>> u
((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
>>> # Tuples are immutable:
... t[0] = 88888
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> # but they can contain mutable objects:
... v = ([1, 2, 3], [3, 2, 1])
>>> v
([1, 2, 3], [3, 2, 1])
```

Tuples can not change in length or value once they have been set.

3.3.3 Dictionaries

Dictionaries are perhaps the most useful means of storing datasets in python. Dictionaries can be thought of as a list containing another list or even a list containing another dictionary. This allows for data to be intuitively stored and retrieved when required. Where lists are indexed with a range of numbers, dictionaries are indexed by keys that can be strings or numbers. The key must be a tuple (immutable).

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
>>> list(tel.keys())
['irv', 'guido', 'jack']
>>> sorted(tel.keys())
['guido', 'irv', 'jack']
>>> 'guido' in tel
True
>>> 'jack' not in tel
False
```

For finance applications dictionaries are useful for storing all the available data about an asset in one easily accessible location. This makes wrangling all the data much easier than having an individual list for every category of data spread throughout the script.

3.4 Pandas Module

The pandas module is an expansion of python dictionaries and has number of extremely useful functions built into it. I'll cover some of the key pandas tools but there are many more available in the pandas documentation.
This module is aimed at making the manipulation of complex datasets much easier and more intuitive for developers. The module helps in streamlining development and reducing the overall complexity of your program.

3.4.1 Creating a Pandas Dataframe

The dataframe is the crux of the pandas module and is an extremely powerful tool for data wrangling. Once into a pandas dataframe rows and columns can easily be accessed.

```python
# import pandas module
import pandas as pd

# Values to go in dataframe
x = [1,2,3,4,5,6]
y = [2,4,8,16,32,64]

# Create the dataframe object
df = pd.DataFrame({'x values':x, 'y values':y})
print(df) # print the full dataset
print(df['x values']) # print x column
print(df.iloc[[2]]) # print the second index row
```

3.4.2 Importing CSV Files Using Pandas

CSV files are a very common techniques for storing large datasets and are easily loaded and exported to and from pandas dataframes.

```python
# import pandas module
import pandas as pd
# import data from csv into pd datafram
df = pd.read_csv('filepath', index_col = '' , header =, parse_dates = )
```

the read_csv() function takes in a number of variables. These are just a few that I find useful when dealing with financial datasets.

- 'filepath': this is the path to the target file. If the csv is in the same directory as the script file you only need to put the file name. This must e a string value.
- index_col: this sets the index of the dataframe and is important for row iterations that index is logical and ordered as you would expect. It is common to index date-time data by the column containing the date-time data, however this is necessary. The default settings index from 0 to the length of the dataset.
- parse_dates: This can be set as a boolean value of "True" or "False". Setting this to True, pandas will check the data for any formatted date-strings and parse them into date-time objects that can be interpreted in python. This is a massive time saver when importing data that contains date-time information.

The pandas module is massive and these are just two of the 100s of tools that are available. Whenever a data wrangling problem arises the pandas documentation is always my first port of call. https://pandas.pydata.org/pandas-docs/stable/whatsnew.html

4 Intermediate

The core of any programming language are control statements and loops. Through using control statements and loops there isn't a task that can't be accomplished. Put simply, control statements are simply "if this value *insert operation* relative to another value then do this". Loops allow these control statements to be repeated until the end objective is completed.

4.1 Control Statements and Loops

4.1.1 If Statements

The if statement compares two values. The "else" and "elif" statements are used to add additional functionality that will be covered further.

```
# values to compare
x = 3
y = 4
if x <= y: # first control statement
  print('No shit mate')
elif y >= x: # second control statement
    print("When does the fun begin?")
```

4.1.2 For loops

For loops are used to iterate over datasets, for example

```
>>> x = ['Mislav', 'Andrew', 'Red', 'Nich', 'Taber', 'Locky']
>>> for i in range(len(x)):
    print(i, x[i])
0 Mislav
1 Andrew
2 Red
3 Nich
4 Taber
5 Locky
```

Here we are setting the for loop to loop from range 0 to the length of the list. This is a typical structure for a python loop. As the index is increased each loop the value at index point "i" in list x is printed out.

4.1.3 While Loops

While loops continue looping through the lines of code until some condition is met.

```python
x = 1
loop = True
while loop:
  x = x+1
  print(x)
  if x > 5:
    loop = False
```

A common loop structure is the ”while True” loop. The loop will continue until the loop is interrupted or is told to stop. Above the loop is initially set to True, but after x > 5 loop is set to false and the loop is broken.

4.2 Functions

Functions are the bread and butter of python scripting. Functions enable the developer to produce an output from a function based on input arguments. These functions can be used repeatably throughout the program and are major part of designing scale-able programs.

```python
x = 3
y = 7

def sum_numbers(x, y):
  result = x + y
  return result

output = sum_numbers(x, y)
print(output)
```

Functions can incorporate control statements and loops to expand their functionality. You can also import functions from previous scripts you have written to be used in other programs, these are known as modules.

4.3 Matplotlib and Visualization

Matplotlib is a data visualization module that has a number of chart types and functions built in to create excellent visual representations of data. Matplotlib is definitely a module all users should become familiar with. matplotlib can be used for pie charts, histograms, candlestick charts, line graphs, scatter graphs, heatmaps, geographic data and heaps more. It is the one stop shop for all things visualization.

4.3.1 Creating a Simple Line Graph

```python
import matplotlib.pyplot as plt

x = [1,2,3,4,5,6,7,8]
y = [2, 15, 32, 92, 200, 400, 1000, 500]

plt.plot(x, y, 'r')
plt.title('Data Values')
plt.xlabel('X values')
plt.ylabel('Y values')
plt.legend(('Data',), loc = 'upper left')
plt.show()
```

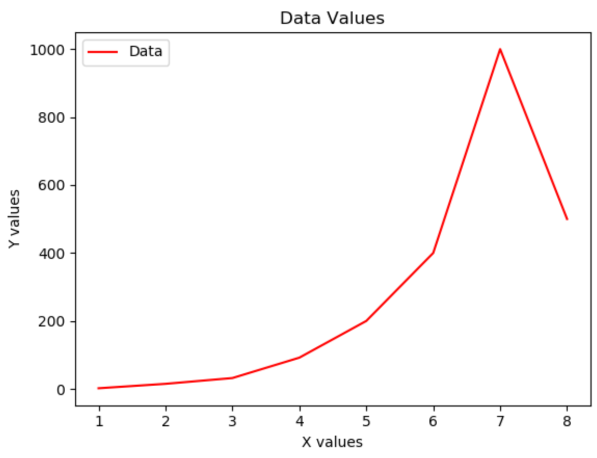


Figure 4: Simple graph using matplotlib