Introduction To Python For Finance

Andrew Collison & Mislav Ilich

Student Trading and Investment Association

February 2018

Learning Outcomes

- Key concepts behind python.
- Install Python on your computer
- Starting and running your python scripts
- Understand variables and their different forms.
- Building and loading dataframes using the pandas module.
- Functions and Classes.
- Loops and control statements.
- Basic data visualization using MATPLOTLIB.

Introduction

Introduction

What Is Python

- Python is a multipurpose programing language, used in areas such as;
  - Defense
  - Security
  - Machine learning
  - and most importantly the finance sector.
- Python is an object oriented programing language that uses functions and classes to operate.
- Python is a space terminated language. That is it uses spaces to separate blocks of code.

Introduction

Why do we use python

- Python is one of the most widely adopted programing languages that you can learn. In recent years it has often often been quoted as the 2nd/3rd most popular programing language in use.
- Python is also an easy language to learn an adapt to current problems. The language is designed around a streamlined work flow to enable for quick implementation of ideas.
- Often python is used to prototype code before adopting it to more efficient languages such as C#.

Introduction

Why do we use python

Figure 1: Current percentage of trades executed using algorithms in equity markets. Source: Multi-Asset Risk Modeling: Techniques for a Global Economy in an Electronic

Getting Started

Getting Started

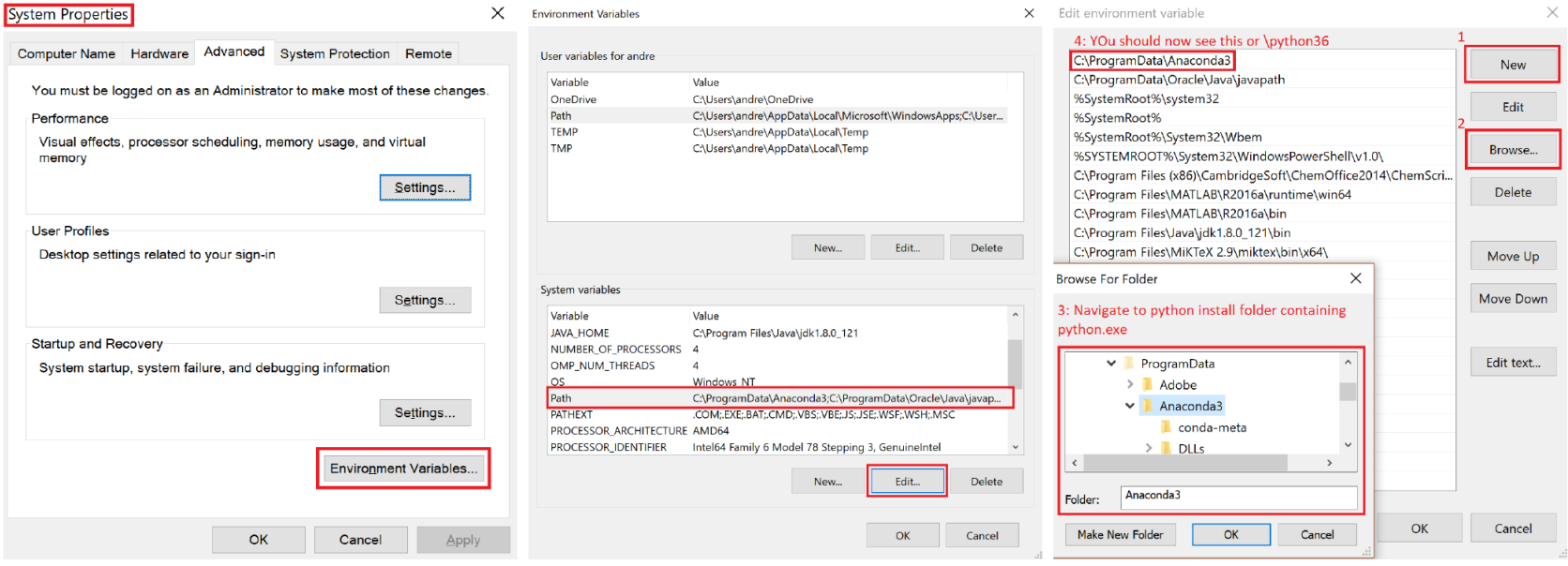
Installing Python3.6 and PIP

1. Download the software. There are two main python installations;
   - From the Python foundation at www.python.org (standard python)
   - A repackaged version of python compiled by Anaconda Inc that includes a number of machine learning modules from anaconda.org (modified to support 64-bit processing)
2. Install the software. This is as simple as running the python-3.6.4.exe file.
   In this step there is a tick box that asks to add python to your system path DO NOT check this box. To avoid over writing your operating system path we will do this after the installation.

Getting Started

Installing Python3.6 and PIP

3. Adding to system path



4. Checking Installation: On windows open a command prompt (windows+r $\rightarrow$ "cmd") and type python into the prompt, press enter.

Getting Started

Installing Python3.6 and PIP

5. Choosing an Interactive Development Environment (IDE):
   - Python comes with a standard text editor called idle. This is a bare-bones program that offers little in the way of features but is a great place to start.
   - Some alternatives to IDLE are NotePad++, PyCharm (excellent), VIM, Visual Studio, iPython Notebook and hundreds more.
   - I'll be using NotePad++ throughout this tutorial, as I find it to be quick and easy to work with.
6. Installing python modules: Modules allow us to extend python, think of them like DLC. They are installed using pip. Eg: typing *pip install pandas* will install the pandas module.

Getting Started

Touching on the command line interface

- The command line interface is a powerful tool that is often over looked by many. It removes the bulky and time consuming methods of pointing and clicking to operate a computer.
- The two commands we will use today
  - dir (windows) and ls (Linux & Mac): this command lists all the files that are in your current directory.
  - cd : used to change directory, cd . and cd.. can be used to jump up levels from your current directory.

Getting Started

First Python Script: "Hello World"

1. Open the hello_world.py file.
2. From here assign a variable to be equal to the string "hello world"
3. Print the variable to the command line.

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

Python Basics

Python Basics

Variables and data types

- Variables: are the locations in which values are stored. They can be called and manipulated throughout the program.

Table 1: Key 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
String	x = "STIA"	Contains a string of letters

```
1 x = 1 # integer
2 y = "STIA" # string
3 z = 3.14 # float
4 # Alternatively
5 x, y, z = 1, "STIA", 3.14
```

Python Basics

Basic Operations

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

Python Basics

Lists

- Python lists are the simplest method for data storage in python.
- Python lists are indexed at 0, such that in list "[x1, x2, x3]", x1 is at position 0.

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

Python Basics

List Operations

- .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.

Python Basics

Dictionaries

- Very useful means of storing data in python.
- Can be thought of as a list of lists
- Rather than being indexed by an integer, they are indexed by a key word known as a tuple.

```
>>> 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']
```

The Pandas Module

The Pandas Module

Introduction to the Pandas module

- What is Pandas?
  Pandas is a module that is popular in data science as it enables for data to be compiled into manageable objects data frame objects.
- Why use pandas?
  Pandas makes managing data easy. Pandas data frames are easy to manipulate and the module also comes with a number of math operations built into it.

The Pandas Module

Creating data frames

- The data frame is the crux of the pandas module and is an extremely powerful tool for data wrangling.

They can be easily created as follows

```
# 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
```

The Pandas Module

Loading data from a .csv file

- Data is commonly stored in a comma separated file known as a csv file. Being able to import this data is a useful skil to have. Pandas also supports many other formats such as json and excel files.

```
# 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 Pandas Module

Loading data from a .csv file

The read_csv() function takes in a number of variables, such as;

- 'filepath': this is the path to the target file.
- 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.
- 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.

Advanced Beginner

Control Statements

The "if" statement

- The if statement compares two values.
- The "else" and "elif" statements are used to add additional functionality.

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

Control Statements

The "for" loop

- 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.

Control Statements

While Loops

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

```
1 x = 1
2 loop = True
3 while loop:
4   x = x+1
5   print(x)
6   if x > 5:
7     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.

Code Structure

Functions

- Functions are a key component to structuring your programs.
- They can be used throughout your code and on different projects.

```
1 x = 3
2 y = 7
3
4 def sum_numbers(x, y):
5     result = x + y
6     return result
7
8 output = sum_numbers(x, y)
9 print(output)
```

- Functions can incorporate control statements and loops
- The "return" statement returns some variable from the function.

Data Visualization using MatPlotLib

MatPlotLib

What is matplotlib?

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.

MatPlotLib

Creating a simple line graph

```
1  import matplotlib.pyplot as plt
2
3  x = [1,2,3,4,5,6,7,8]
4  y = [2, 15, 32, 92, 200, 400, 1000, 500]
5
6  plt.plot(x, y, 'r')
7  plt.title('Data Values')
8  plt.xlabel('X values')
9  plt.ylabel('Y values')
10 plt.legend(('Data',), loc = 'upper left')
11 plt.show()
```

MatPlotLib

Creating a simple line graph