

## Introduction to Python Getting Started

### Contents

<b>0 Getting started</b>	<b>1</b>
0.1 Introduction	1
0.2 Installing Python and Jupyter Notebook	2
0.3 A first example	4

## 0 Getting started

### 0.1 Introduction

#### Introduction

This mini-course will help you learn to:

- navigate in a Jupyter Notebook
- be able to code simple Python code (variables + lists, conditions, flow control)
- know how to find help in the Python universe / zoo
- basics of `numpy` and `scipy`
- basics of `pandas`
- basics of plotting

I will also create a set of videos for self-learning on “Statistics with Python” to build on the material from your first session.

A great reference is **Python for Finance (2nd edition)** by Yves Hilpisch. (It uses Python version 3.7.)

#### Background

- Python is an open-source programming language that can be downloaded and used for free.
- Python was created by Guido van Rossum and first published in 1991.
- Today the language is largely developed by the Python Software Foundation, a nonprofit organization.
- It is named after the British comedy group “Monty Python”.

```
[1]: import this
```

The Zen of Python, by Tim Peters

```
Beautiful is better than ugly.  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
Sparse is better than dense.  
Readability counts.  
Special cases aren't special enough to break the rules.  
Although practicality beats purity.  
Errors should never pass silently.
```

Unless explicitly silenced.  
In the face of ambiguity, refuse the temptation to guess.  
There should be one-- and preferably only one --obvious way to do it.  
Although that way may not be obvious at first unless you're Dutch.  
Now is better than never.  
Although never is often better than *\*right\** now.  
If the implementation is hard to explain, it's a bad idea.  
If the implementation is easy to explain, it may be a good idea.  
Namespaces are one honking great idea -- let's do more of those!

Many examples and extended information can be found on the following websites:

- [Beginners' Guide](#)
- [Python.org](#)
- [Scipy Lectures](#)
- [The Hitchhiker's Guide to Python](#)

## Pros

- **Universal:** Python runs on any operating system.
- **Easy to learn:** Although Python is highly versatile (e.g. can be used for scientific computing), it is relatively easy to learn.
- **Readable code:** Python is a high-level programming language, making it easy to read and work with.
- **General purpose:** The language can be applied to solve different problems at hand.
- **Open source and free.**
- **Cross-platform**
- **Indentation aware:** indentation is used instead of braces to mark code blocks.

## Cons

- **Speed:** While Python is not slow, it cannot keep up with compiled languages such as C, C++, Fortran, COBOL, etc.

## 0.2 Installing Python and Jupyter Notebook

### Setting up Python for this class

- We will use **Python 3.7** in this class.
- Easiest method to get started:
  - Install [Anaconda](#); this will setup up **Python** as well as the **Jupyter Notebook** environment that we are going to use in class.
  - Make sure to install a version of Anaconda that installs Python 3.7 (as opposed to the current version 3.8).
  - For this, go to the archive <https://repo.anaconda.com/archive/> and install the version named Anaconda3-2020.02 appropriate for your operating system.
- The code will most likely work with a newer version of Python as well.

### Jupyter Notebook

- **Jupyter Notebook** is a browser-based application used for creating and sharing documents, containing live code, visualizations, equations, plain text, and many other features.
- Launch Jupyter Notebook:
  - from Anaconda or
  - from the command line (**Terminal** in MacOS or **cmd** in Windows) using the command

```
jupyter notebook
```

- Jupyter notebooks run in a local webserver.

## Jupyter Notebook

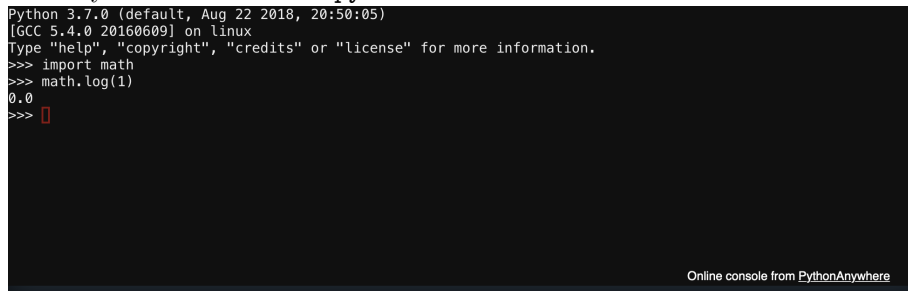
- Before launching Jupyter Notebook from the command line you may wish to navigate to the directory where the notebooks are stored or where you want to store them.
- To open a new notebook click on “New” and “Python 3” in the Jupyter Notebook main window.
- To learn to use Jupyter Notebook before, click on “Help” and “User Interface Tour”.

## Using Colab

- If you don’t have Jupyter Notebook installed yet, you can use the online service <https://colab.research.google.com>.
- However, I strongly recommend to install you own version at some point.

## Other ways to run Python

- The Python **shell**: call `python` from a shell or start an online version at [online shell](#)



```
Python 3.7.0 (default, Aug 22 2018, 20:50:05)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import math
>>> math.log(1)
0.0
>>> 
```

Online console from [PythonAnywhere](#)

## Other ways to run Python

- Run Python scripts (suffix `.py`) from a command line using the `python` command, e.g. `python file.py`.
- Python’s **IDLE** (Integrated Development and Learning Environment) is another basic shell to run Python commands.
- An **IDE** (Integrated Development Environment) is an application that integrates programming, running code, debugging, etc.
- **IPython**: Interactive Python shell.

## The Python Ecosystem

- Aside from the programming language, there is a large number of packages, modules and other tools available to support specific tasks.
- For example, various plotting libraries are available and can be readily used using `import`.

## Popular packages (“The scientific stack”)

- **NumPy**: multidimensional array objects
- **SciPy**: functionality often needed in science or finance
- **matplotlib**: plotting
- **pandas**: times series and tabular data
- **scikit-learn**: machine learning package
- **PyTables**: data storage package

## Navigating Jupyter Notebook

- Cells in Jupyter Notebook are either “text cells”, where you can write Markdown code, or they are “code cells” containing... well, code.
- Markdown cheat sheet: <https://www.markdownguide.org/cheat-sheet/>
- To type in a cell, click on it until the bar on the left turns green.
- To exit typing mode, hit the ESC button (the bar turns blue).
- To run a cell (text or code), type: SHIFT-RETURN.

How code is run:

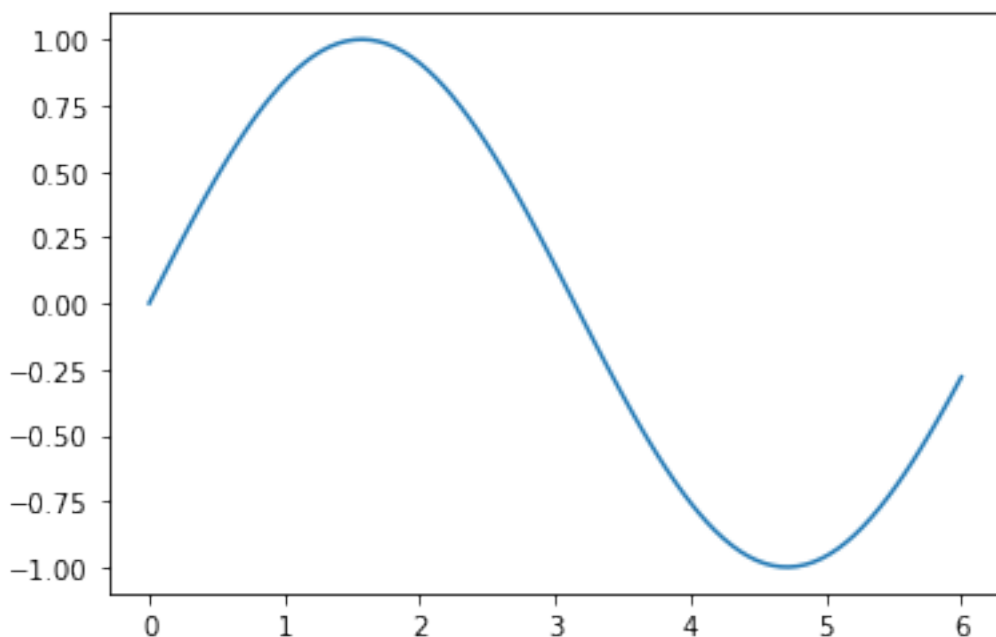
- A Python interpreter (“the kernel”) is running in the background waiting for work.
- Whenever you hit SHIFT-RETURN, the Python interpreter will run the code in the cell.
- It will also update its internal state, for example if you load packages, introduce variables or assign values to variables.
- Closing the kernel erases the internal state of the session. (Note how this is different from saving the file.)

### Navigating Jupyter Notebook

- Spend a moment to make yourself familiar with the functionality in a Jupyter Notebook.
- The little keyboard icon below the menu opens a dialog box with all possible Notebook commands.
- You can type to narrow down the commands.
- It also shows keyboard shortcuts associated with commands.
- A couple of commands are used so often (e.g. opening a new cell below the current one), that it pays off to memorise their shortcuts.
- Aside from the usual “File”, “Edit”, “View” entries, the menu lets you operate the kernel.

### Navigating Jupyter Notebook

```
[2]: import numpy as np
import math
import matplotlib.pyplot as plt
x = np.linspace(0, 6, 100);
y = np.sin(x);
plt.plot(x, y);
```



### Popular packages (“The scientific stack”)

- **NumPy**: multidimensional array objects
- **SciPy**: functionality often needed in science or finance
- **matplotlib**: plotting
- **pandas**: times series and tabular data

- `scikit-learn`: machine learning package
- `PyTables`: data storage package

### 0.3 A first example

#### Example loading and plotting financial data

- The following code imports the packages that will be used and sets up the plotting library.

```
[3]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn')
plt.rcParams['font.family'] = 'serif'
```

#### Example loading and plotting financial data

- The following code imports a time series of S&P 500 index data from a csv file (csv=comma separated values).
- It also shows information about the time series.

```
[4]: data = pd.read_csv('data/tr_eikon_eod_data.csv', index_col=0, parse_dates=True)
data = pd.DataFrame(data['.SPX'])
data.dropna(inplace=True)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2138 entries, 2010-01-04 to 2018-06-29
Data columns (total 1 columns):
#   Column  Non-Null Count  Dtype
---  -
0    .SPX    2138 non-null         float64
dtypes: float64(1)
memory usage: 33.4 KB
```

#### Example loading and plotting financial data

- Transform the index level data to log-returns, estimate volatility (=standard deviation of returns) and produce plots of index and volatility.

```
[5]: data['rets'] = np.log(data / data.shift(1))
data['vola'] = data['rets'].rolling(252).std() * np.sqrt(252)
data[['.SPX', 'vola']].plot(subplots=True, figsize=(10, 6));
```

