

PythonTutorial

September 3, 2021

1 Python Quickstart

This is called a jupyter notebook meant to give a quick introduction to Python. You can execute the code in each cell by clicking in the cell and hitting “Shift-Enter”. Restart the notebook using the small square double arrow button above.

```
[1]: # Comments start with a hash
```

```
[2]: # First thing to do in a new program is import all your libraries
# To install new libraries, just do a "pip install packagename"
#     or "conda install packagename", but
#     anaconda comes with a lot of stuff
#     so you should be good with the base install

import numpy as np           # Numpy is a numerical library.
                              # It is built in C which gives it
                              # significant performance benefits over
                              # pure python code

import pandas as pd          # Pandas is great when you are working
                              # with datafiles.

import matplotlib.pyplot as plt # Plotting library

# The "as" command tells python that "np" is short for "numpy"
```

```
[3]: # Setting variables

a = 4           # integer
b = 500         # integer
c = 10.1        # float
d = 1.0e6       # float
e = "a string" # string
```

```
[4]: # Printing variables

print(a)
print(b)
```

```
print(a,b)
print("%d %d"%(a,b))      # Here both are formatted as integers
print("%3.2f %e"%(a,b))   # Formatted as float and scientific notation
```

```
4
500
4 500
4 500
4.00 5.000000e+02
```

[5]: *# For loop (notice for loops start with 0, end with n-1)*

```
for n in range(10):
    print("%d^2 = %d"%(n,n**2))
```

```
0^2 = 0
1^2 = 1
2^2 = 4
3^2 = 9
4^2 = 16
5^2 = 25
6^2 = 36
7^2 = 49
8^2 = 64
9^2 = 81
```

[6]: *# While loop (notice we need to increment n manually
whereas the for loop does it for you)*

```
n = 0
while n < 10:
    print("%d^2 = %d"%(n,n**2))
    n += 1
```

```
0^2 = 0
1^2 = 1
2^2 = 4
3^2 = 9
4^2 = 16
5^2 = 25
6^2 = 36
7^2 = 49
8^2 = 64
9^2 = 81
```

[7]: `print("-- Python Lists --")`
Python uses lists to store multi-dimensional data

```

a = []                                # Empty list
print(a)
a = [0, 1, 2, 3]                      # List with 4 elements
print(a)
a = [n for n in range(10)]            # List with a loop in it
print(a[5])                          # Get fifth element in list
print(len(a))                        # Get size of lists with len()

print("\n-- Dictionaries --")
# Dictionaries are similar to arrays,
# but you can set the index to anything you want, like a string
a = {"Mike": 1, "Steve": 2}
print(a)
print(a["Mike"])

print("\n-- Numpy Arrays --")
# An alternative to lists are numpy arrays.
# Operations with numpy arrays are generally much faster.
# Use them whenever possible
a = np.array([])                      # Empty array
print(a)
a = np.array([0, 1, 2, 3])            # Similar to lists, but better
print(a)
a = np.arange(10)                     # Initialize an array [0, 10)
print(a)
a = np.zeros((1000,1000))             # Really big array of all zeros
print(a)
print(a.shape)                       # Get size of arrays with shape

```

-- Python Lists --

```

[]
[0, 1, 2, 3]
5
10

```

-- Dictionaries --

```

{'Mike': 1, 'Steve': 2}
1

```

-- Numpy Arrays --

```

[]
[0 1 2 3]
[0 1 2 3 4 5 6 7 8 9]
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...

```

```
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]
(1000, 1000)
```

```
[8]: # Import a year's worth of financial data for apple, amazon, and walmart

aapl = pd.read_csv("AAPL.csv") # Read a csv (comma separated value) file
amzn = pd.read_csv("AMZN.csv")
wmt = pd.read_csv("WMT.csv")

# These are all now pandas dataframes.
# You can take a look at the data by using the head (top five lines)
# or tail (bottom five lines)
aapl.head()
```

```
[8]:
```

	Date	Open	High	Low	Close	Adj Close	\
0	2019-06-04	175.440002	179.830002	174.520004	179.639999	177.521378	
1	2019-06-05	184.279999	184.990005	181.139999	182.539993	180.387146	
2	2019-06-06	183.080002	185.470001	182.149994	185.220001	183.035568	
3	2019-06-07	186.509995	191.919998	185.770004	190.149994	187.907410	
4	2019-06-10	191.809998	195.369995	191.619995	192.580002	190.308762	

	Volume
0	30968000
1	29773400
2	22526300
3	30684400
4	26220900

```
[9]: # You can also get some statistical data using the describe function
aapl.describe()
```

```
[9]:
```

	Open	High	Low	Close	Adj Close	\
count	253.000000	253.000000	253.000000	253.000000	253.000000	
mean	254.951423	258.222055	252.456522	255.601304	254.169563	
std	42.722106	43.435319	42.304440	43.076031	43.563057	
min	175.440002	179.830002	174.520004	179.639999	177.521378	
25%	213.190002	214.419998	211.070007	212.639999	210.752029	
50%	257.260010	260.440002	255.380005	259.429993	258.117004	
75%	289.929993	297.880005	286.320007	292.649994	291.859924	
max	324.739990	327.850006	323.350006	327.200012	326.316681	

	Volume
count	2.530000e+02
mean	3.347581e+07
std	1.718790e+07

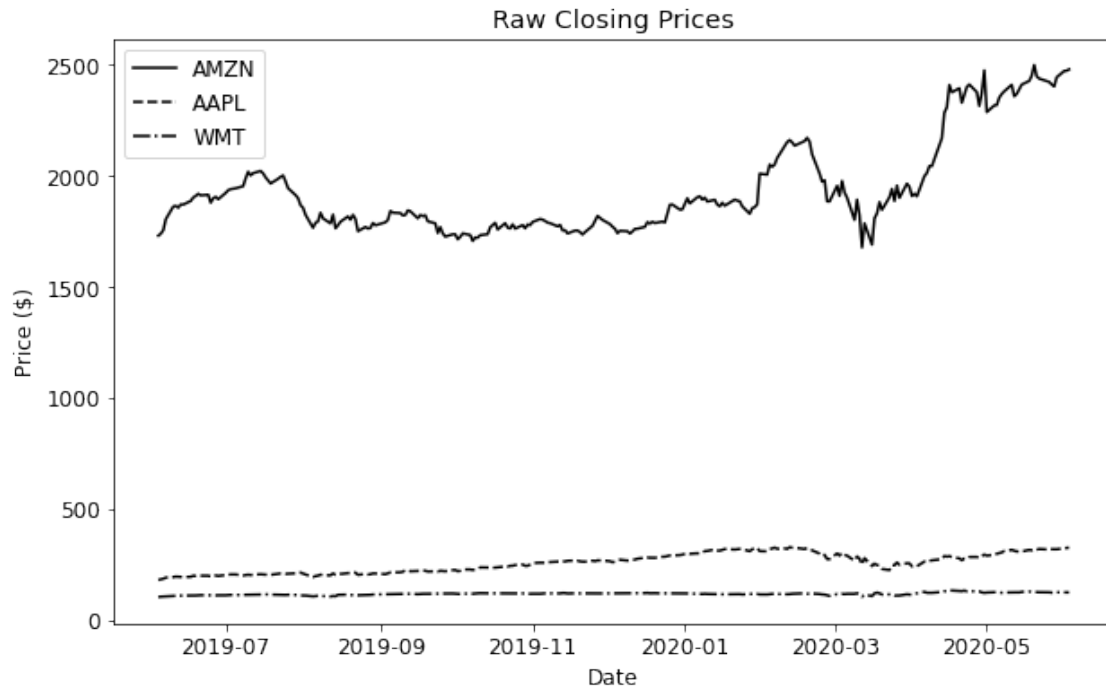
```
min    1.136200e+07
25%    2.184000e+07
50%    2.843260e+07
75%    3.813280e+07
max    1.067212e+08
```

```
[10]: # The dataframes all have a "Date" column,
#      but Python doesn't know it's a date yet.
#      You need to define it as a date
```

```
amzn["Date"] = pd.to_datetime(amzn["Date"])
aapl["Date"] = pd.to_datetime(aapl["Date"])
wmt["Date"] = pd.to_datetime(wmt["Date"])
```

```
[11]: # Plot the raw close prices
```

```
plt.figure(figsize=(10,6))    # This sets the size of the plot,
                                #    10 units wide by 6 units high
plt.rcParams['font.size'] = 12
plt.title("Raw Closing Prices")
plt.xlabel("Date")
plt.ylabel("Price ($)")
plt.plot(amzn["Date"], amzn["Close"], 'k-', label="AMZN")
# The arguments here from left to right are x-variable,
#    y-variable, line type (k for black, - for solid line),
#    and line label for the legend
plt.plot(aapl["Date"], aapl["Close"], 'k--', label="AAPL")
plt.plot(wmt["Date"], wmt["Close"], 'k-.', label="WMT")
plt.legend()    # Add a legend
plt.show()
```



```
[12]: # That's not a very good plot because amazon
#      is so much higher than the other two
#      Try plotting daily price change instead

amzn_pc = amzn["Close"].pct_change() # Compute daily percent change
aapl_pc = aapl["Close"].pct_change()
wmt_pc = wmt["Close"].pct_change()

amzn["Close_change"] = amzn_pc # Add a column in the dataframe
aapl["Close_change"] = aapl_pc
wmt["Close_change"] = wmt_pc

# Clip off the first entry since the percent change
# has one less row than the others
amzn = amzn.iloc[1:]
aapl = aapl.iloc[1:]
wmt = wmt.iloc[1:]

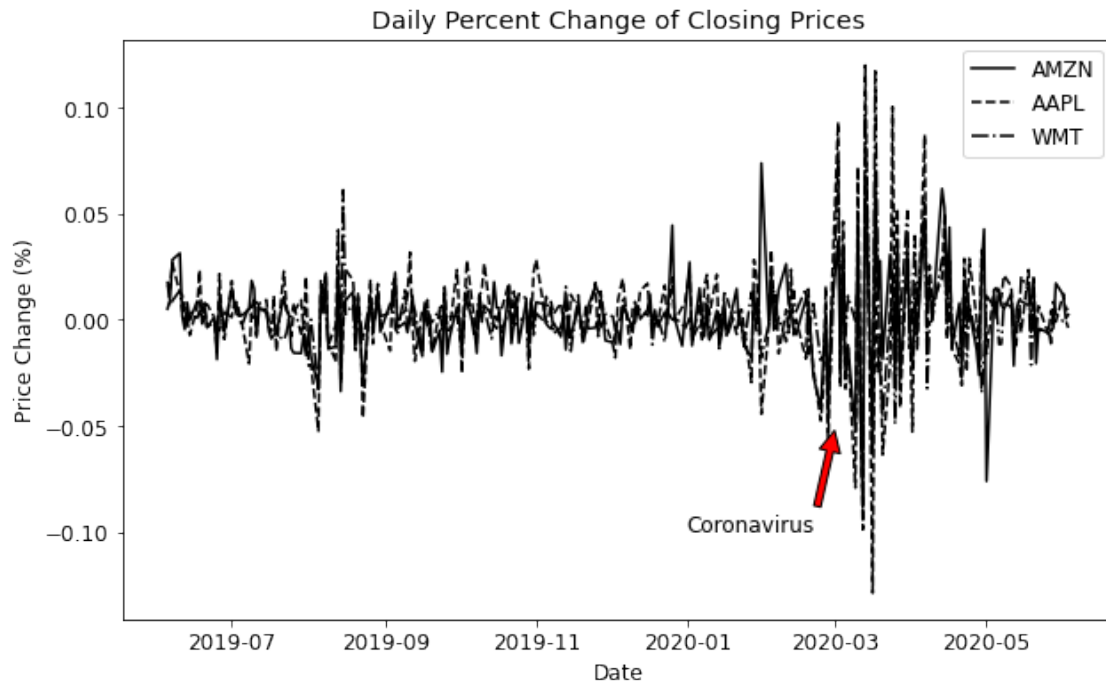
plt.figure(figsize=(10,6))
plt.rcParams['font.size'] = 12
plt.title("Daily Percent Change of Closing Prices")
plt.xlabel("Date")
plt.ylabel("Price Change (%)")
plt.plot(amzn["Date"], amzn["Close_change"], 'k-', label="AMZN")
```

```

plt.plot(aapl["Date"], aapl["Close_change"], 'k--', label="AAPL")
plt.plot(wmt["Date"], wmt["Close_change"], 'k-.', label="WMT")
plt.gca().annotate('Coronavirus', xy=(pd.Timestamp(2020,3,1), -0.05),
                    xytext=(pd.Timestamp(2020,1,1), -0.1),
                    arrowprops=dict(facecolor='red', shrink=0.05),
                    )

plt.legend()
plt.show()

```



```

[13]: # Now assume we invested 1 dollar in each stock a year ago.
      # How much do we have today?

      # Add one to each row
      amzn_c = amzn_pc + 1
      # First row gets $1 investment
      amzn_c.iloc[0] = 1.0
      # Multiply each row with the row prior and sum
      amzn_val = np.cumprod(amzn_c)

      aapl_c = aapl_pc + 1
      aapl_c.iloc[0] = 1.0
      aapl_val = np.cumprod(aapl_c)

      wmt_c = wmt_pc + 1

```

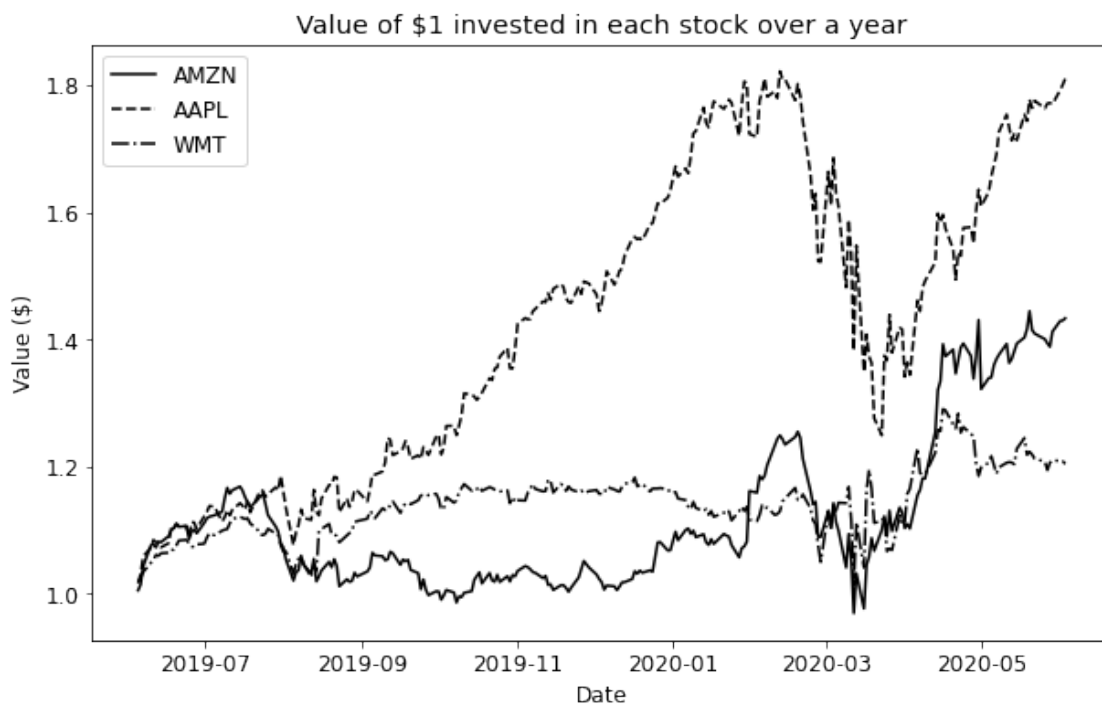
```

wmt_c.iloc[0] = 1.0
wmt_val = np.cumprod(wmt_c)

amzn["Value"] = amzn_val
aapl["Value"] = aapl_val
wmt["Value"] = wmt_val

plt.figure(figsize=(10,6))
plt.rcParams['font.size'] = 12
plt.title("Value of $1 invested in each stock over a year")
plt.xlabel("Date")
plt.ylabel("Value ($)")
plt.plot(amzn["Date"], amzn["Value"], 'k-', label="AMZN")
plt.plot(aapl["Date"], aapl["Value"], 'k--', label="AAPL")
plt.plot(wmt["Date"], wmt["Value"], 'k-.', label="WMT")
plt.legend()
plt.show()

```



```

[14]: # Value of the whole portfolio

port_val = amzn["Value"] + aapl["Value"] + wmt["Value"]

plt.figure(figsize=(10,6))
plt.rcParams['font.size'] = 12

```



```
plt.title("Value of portfolio over a year")
plt.xlabel("Date")
plt.ylabel("Value ($)")
plt.plot(amzn["Date"], port_val, 'k-')
plt.savefig("portfolio.png")
```

