

Functional Programming

1.0 Lambda Function

Syntax : lambda p1, p2: expression

Here, p1 and p2 are the parameters which are passed to the lambda function. You can add as many or few parameters as you need.

However, notice that we do not use brackets around the parameters as we do with regular functions. The last part (expression) is any valid python expression that operates on the parameters you provide to the function.

```
In [9]: adder = lambda x, y: x + y
        print(adder(1, 2))
```

3

```
In [11]: mult = lambda x,y : x*y
         print(mult(2,7))
```

14

2.0 Map

Python map() applies a function on all the items of an iterator given as input. An iterator, for example, can be a list, a tuple, a set, a dictionary, a string, and it returns an iterable map object. Python map() is a built-in function

Syntax: map(function, iterator1, iterator2 ...iteratorN)

Parameters

Here are two important

function:

A mandatory function to be given to map, that will be applied to all the items available in the iterator.

iterator:

An iterable compulsory object. It can be a list, a tuple, etc. You can pass multiple iterator objects to map() function.

```
In [1]: intList = [1,2,3,4,5,6,7,8,9]
```

```
In [1]: def square_me( x ):
        return x * x
```

```
In [6]: squareList = map( square_me, intList)
```

```
In [56]: list(squareList)
```

```
Out[56]: [1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
In [57]: squareList = map(lambda x: x*x, intList)
        list(squareList)
```

```
Out[57]: [1, 4, 9, 16, 25, 36, 49, 64, 81]
```

3.0 Filter

The filter() function returns an iterator where the items are filtered through a function to test if the item is accepted or not.

Syntax : filter(function, iterable)

```
In [58]: evenInts = filter( lambda x : x % 2 == 0, intList )
```

```
In [59]: list( evenInts )
```

```
Out[59]: [2, 4, 6, 8]
```

4.0 Modules and Packages

```
In [60]: import math
        ## Taking square root of a value
        math.sqrt(16)
```

```
Out[60]: 4.0
```

```
In [61]: from random import sample
```

```
In [62]: sample( range(0, 11), 3)
```

```
Out[62]: [8, 0, 1]
```

5.0 Other Features

```
In [63]: import random
        randomList = random.sample( range(0, 100), 20)
        randomList
```

```
Out[63]: [92,
          77,
          63,
          22,
          33,
          20,
          84,
          82,
          54,
          97,
          93,
          31,
          61,
          30,
          10,
          55,
          96,
          32,
          36,
          14]
```

```
In [65]: from statistics import mean, median
        def getMeanAndMedian( listNum ):
            return mean(listNum), median(listNum)
```

```
In [66]: mean, median = getMeanAndMedian( randomList )
```

```
In [67]: print( "Mean: ", mean, " Median: ", median)
```

Mean: 54.1 Median: 54.5

```
In [68]: pwd
```

Out[68]: 'C:\\Users\\thyagu'

Bitcoin Analysis

1.0 Import Packages

```
In [6]: import datetime as dt
import pandas as pd
import pandas_datareader.data as web
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.dates as mdates
```

```
In [7]: import plotly.express as px
```

```
In [8]: #Data Source
import yfinance as yf
#Data viz
import plotly.graph_objs as go
```

1.2 Dowloading Data From YFinance

```
In [9]: data = yf.download(tickers='BTC-USD', period = '24h', interval = '15m')
```

[*****100%*****] 1 of 1 completed

```
In [10]: print(data)
```

| | Open | High | Low \ |
|---------------------------|--------------|--------------|--------------|
| Datetime | | | |
| 2021-11-08 09:30:00+00:00 | 66197.968750 | 66216.015625 | 66089.140625 |
| 2021-11-08 09:45:00+00:00 | 66149.656250 | 66268.078125 | 66114.054688 |
| 2021-11-08 10:00:00+00:00 | 66149.945312 | 66216.203125 | 66026.335938 |
| 2021-11-08 10:15:00+00:00 | 66072.406250 | 66072.406250 | 65803.414062 |
| 2021-11-08 10:30:00+00:00 | 65896.937500 | 66037.625000 | 65888.617188 |
| ... | ... | ... | ... |
| 2021-11-09 08:45:00+00:00 | 67994.687500 | 68161.351562 | 67994.687500 |
| 2021-11-09 09:00:00+00:00 | 68138.726562 | 68138.726562 | 67987.085938 |
| 2021-11-09 09:15:00+00:00 | 67986.656250 | 67993.210938 | 67935.109375 |
| 2021-11-09 09:30:00+00:00 | 67964.789062 | 68069.476562 | 67932.382812 |
| 2021-11-09 09:37:02+00:00 | 67959.414062 | 67959.414062 | 67959.414062 |
| | Close | Adj Close | Volume |
| Datetime | | | |
| 2021-11-08 09:30:00+00:00 | 66196.773438 | 66196.773438 | 52944896 |
| 2021-11-08 09:45:00+00:00 | 66170.492188 | 66170.492188 | 108097536 |
| 2021-11-08 10:00:00+00:00 | 66026.359375 | 66026.359375 | 5142528 |
| 2021-11-08 10:15:00+00:00 | 65903.562500 | 65903.562500 | 30717952 |
| 2021-11-08 10:30:00+00:00 | 66008.078125 | 66008.078125 | 24825856 |
| ... | ... | ... | ... |
| 2021-11-09 08:45:00+00:00 | 68161.351562 | 68161.351562 | 39538688 |
| 2021-11-09 09:00:00+00:00 | 67987.085938 | 67987.085938 | 0 |
| 2021-11-09 09:15:00+00:00 | 67941.382812 | 67941.382812 | 41369600 |
| 2021-11-09 09:30:00+00:00 | 67982.046875 | 67982.046875 | 78712832 |
| 2021-11-09 09:37:02+00:00 | 67959.414062 | 67959.414062 | 0 |

[98 rows x 6 columns]

1.3 Setting Start and End Time For Analysis

```
In [19]: start = dt.datetime(2021,10,1)
end = dt.datetime(2021,11,9)
```

1.4 Reading BTC Data from Start time to End Time

```
In [12]: btc = web.DataReader("BTC-USD", 'yahoo', start, end) # Collects data
btc.reset_index(inplace=True)
btc
```

| Out[12]: | Date | High | Low | Open | Close | Volume | Adj Close |
|----------|------------|--------------|--------------|--------------|--------------|-------------|--------------|
| 0 | 2021-09-29 | 44092.601562 | 41444.582031 | 41551.269531 | 43790.894531 | 31141681925 | 43790.894531 |
| 1 | 2021-09-30 | 48436.011719 | 43320.023438 | 43816.742188 | 48116.941406 | 42850641582 | 48116.941406 |
| 2 | 2021-10-01 | 48282.062500 | 47465.496094 | 48137.468750 | 47711.488281 | 30614346492 | 47711.488281 |
| 3 | 2021-10-02 | 49130.691406 | 47157.289062 | 47680.027344 | 48199.953125 | 26638115879 | 48199.953125 |
| 4 | 2021-10-03 | 49456.777344 | 47045.003906 | 48208.906250 | 49112.902344 | 33383173002 | 49112.902344 |
| 5 | 2021-10-04 | 51839.984375 | 49072.839844 | 49174.960938 | 51514.812500 | 35873904236 | 51514.812500 |
| 6 | 2021-10-05 | 55568.464844 | 50488.191406 | 51486.664062 | 55361.449219 | 49034730168 | 55361.449219 |
| 7 | 2021-10-06 | 55338.625000 | 53525.468750 | 55338.625000 | 53805.984375 | 36807860413 | 53805.984375 |
| 8 | 2021-10-07 | 55922.980469 | 53688.054688 | 53802.144531 | 53967.847656 | 34800873924 | 53967.847656 |
| 9 | 2021-10-08 | 55397.945312 | 53735.144531 | 53929.781250 | 54968.222656 | 32491211414 | 54968.222656 |
| 10 | 2021-10-09 | 56401.304688 | 54264.257812 | 54952.820312 | 54771.578125 | 39527792364 | 54771.578125 |
| 11 | 2021-10-10 | 57793.039062 | 54519.765625 | 54734.125000 | 57484.789062 | 42637331698 | 57484.789062 |
| 12 | 2021-10-11 | 57627.878906 | 54477.972656 | 57526.832031 | 56041.058594 | 41083758949 | 56041.058594 |
| 13 | 2021-10-12 | 57688.660156 | 54370.972656 | 56038.257812 | 57401.097656 | 41684252783 | 57401.097656 |
| 14 | 2021-10-13 | 58478.734375 | 56957.074219 | 57372.832031 | 57321.523438 | 36615791366 | 57321.523438 |
| 15 | 2021-10-14 | 62757.128906 | 56868.144531 | 57345.902344 | 61593.949219 | 51780081801 | 61593.949219 |
| 16 | 2021-10-15 | 62274.476562 | 60206.121094 | 61609.527344 | 60892.179688 | 34250964237 | 60892.179688 |
| 17 | 2021-10-16 | 61645.523438 | 59164.468750 | 60887.652344 | 61553.617188 | 29032367511 | 61553.617188 |
| 18 | 2021-10-17 | 62614.660156 | 60012.757812 | 61548.804688 | 62026.078125 | 38055562075 | 62026.078125 |
| 19 | 2021-10-18 | 64434.535156 | 61622.933594 | 62043.164062 | 64261.992188 | 40471196346 | 64261.992188 |
| 20 | 2021-10-19 | 66930.390625 | 63610.675781 | 64284.585938 | 65992.835938 | 40788955582 | 65992.835938 |
| 21 | 2021-10-20 | 66600.546875 | 62117.410156 | 66002.234375 | 62210.171875 | 45908121370 | 62210.171875 |
| 22 | 2021-10-21 | 63715.023438 | 60122.796875 | 62237.890625 | 60692.265625 | 38434082775 | 60692.265625 |
| 23 | 2021-10-22 | 61743.878906 | 59826.523438 | 60694.628906 | 61393.617188 | 26882546034 | 61393.617188 |
| 24 | 2021-10-23 | 61505.804688 | 59643.343750 | 61368.343750 | 60930.835938 | 27316183882 | 60930.835938 |
| 25 | 2021-10-24 | 63729.324219 | 60691.800781 | 60893.925781 | 63039.824219 | 31064911614 | 63039.824219 |
| 26 | 2021-10-25 | 63229.027344 | 59991.160156 | 63032.761719 | 60363.792969 | 34878965587 | 60363.792969 |
| 27 | 2021-10-26 | 61435.183594 | 58208.187500 | 60352.000000 | 58482.386719 | 43657076893 | 58482.386719 |
| 28 | 2021-10-27 | 62128.632812 | 58206.917969 | 58470.730469 | 60622.136719 | 45257083247 | 60622.136719 |
| 29 | 2021-10-28 | 62927.609375 | 60329.964844 | 60624.871094 | 62227.964844 | 36856881767 | 62227.964844 |
| 30 | 2021-10-29 | 62330.144531 | 60918.386719 | 62239.363281 | 61888.832031 | 32157938616 | 61888.832031 |
| 31 | 2021-10-30 | 62406.171875 | 60074.328125 | 61850.488281 | 61318.957031 | 32241199927 | 61318.957031 |
| 32 | 2021-11-01 | 62419.003906 | 59695.183594 | 61320.449219 | 61004.406250 | 36150572843 | 61004.406250 |
| 33 | 2021-11-02 | 64242.792969 | 60673.054688 | 60963.253906 | 63226.402344 | 37746665647 | 63226.402344 |
| 34 | 2021-11-03 | 63516.937500 | 61184.238281 | 63254.335938 | 62970.046875 | 36124731509 | 62970.046875 |
| 35 | 2021-11-04 | 63123.289062 | 60799.664062 | 62941.804688 | 61452.230469 | 32615846901 | 61452.230469 |
| 36 | 2021-11-05 | 62541.468750 | 60844.609375 | 61460.078125 | 61125.675781 | 30605102446 | 61125.675781 |
| 37 | 2021-11-06 | 61590.683594 | 60163.781250 | 61068.875000 | 61527.480469 | 29094934221 | 61527.480469 |
| 38 | 2021-11-07 | 63326.988281 | 61432.488281 | 61554.921875 | 63326.988281 | 24726754302 | 63326.988281 |
| 39 | 2021-11-08 | 67673.742188 | 63344.066406 | 63344.066406 | 67566.828125 | 41125608330 | 67566.828125 |
| 40 | 2021-11-09 | 68530.335938 | 67321.320312 | 67380.914062 | 67959.414062 | 40595714048 | 67959.414062 |

1.4 Displaying the Adjusted Closing Prices of Bitcoin

```
In [13]: #bitcoin
crypto= btc[['Date', 'Adj Close']]
crypto= crypto.rename(columns = {'Adj Close': 'BTC'})
```

```
In [14]: crypto
```

```
Out[14]:
```

| | Date | BTC |
|----|------------|--------------|
| 0 | 2021-09-29 | 43790.894531 |
| 1 | 2021-09-30 | 48116.941406 |
| 2 | 2021-10-01 | 47711.488281 |
| 3 | 2021-10-02 | 48199.953125 |
| 4 | 2021-10-03 | 49112.902344 |
| 5 | 2021-10-04 | 51514.812500 |
| 6 | 2021-10-05 | 55361.449219 |
| 7 | 2021-10-06 | 53805.984375 |
| 8 | 2021-10-07 | 53967.847656 |
| 9 | 2021-10-08 | 54968.222656 |
| 10 | 2021-10-09 | 54771.578125 |
| 11 | 2021-10-10 | 57484.789062 |
| 12 | 2021-10-11 | 56041.058594 |
| 13 | 2021-10-12 | 57401.097656 |
| 14 | 2021-10-13 | 57321.523438 |
| 15 | 2021-10-14 | 61593.949219 |
| 16 | 2021-10-15 | 60892.179688 |
| 17 | 2021-10-16 | 61553.617188 |
| 18 | 2021-10-17 | 62026.078125 |
| 19 | 2021-10-18 | 64261.992188 |
| 20 | 2021-10-19 | 65992.835938 |
| 21 | 2021-10-20 | 62210.171875 |
| 22 | 2021-10-21 | 60692.265625 |
| 23 | 2021-10-22 | 61393.617188 |
| 24 | 2021-10-23 | 60930.835938 |
| 25 | 2021-10-24 | 63039.824219 |
| 26 | 2021-10-25 | 60363.792969 |
| 27 | 2021-10-26 | 58482.386719 |
| 28 | 2021-10-27 | 60622.136719 |
| 29 | 2021-10-28 | 62227.964844 |
| 30 | 2021-10-29 | 61888.832031 |
| 31 | 2021-10-30 | 61318.957031 |
| 32 | 2021-11-01 | 61004.406250 |
| 33 | 2021-11-02 | 63226.402344 |
| 34 | 2021-11-03 | 62970.046875 |
| 35 | 2021-11-04 | 61452.230469 |
| 36 | 2021-11-05 | 61125.675781 |
| 37 | 2021-11-06 | 61527.480469 |
| 38 | 2021-11-07 | 63326.988281 |
| 39 | 2021-11-08 | 67566.828125 |
| 40 | 2021-11-09 | 67959.414062 |

1.5 Displaying the 7 Days Moving Average of Bit Coin

```
In [16]: # 7 day moving average
crypto[ 'BTC_7DAY_MA' ] = crypto.BTC.rolling(7).mean()
crypto[ 'BTC_7DAY_MA' ]
```

```
Out[16]: 0          NaN
1          NaN
2          NaN
3          NaN
4          NaN
5          NaN
6    49115.491629
7    50546.218750
8    51382.062500
9    52418.738839
10   53357.542411
11   54553.526228
12   55200.132812
13   55491.511161
14   55993.731027
15   57083.174107
16   57929.453683
17   58898.316406
18   59547.071987
19   60721.491071
20   61948.882254
21   62647.260603
22   62518.448661
23   62590.082589
24   62501.113839
25   62645.934710
26   62089.049107
27   61016.127790
28   60789.265625
29   61008.651228
30   61079.396205
31   61134.842076
32   60844.068080
33   61253.012277
34   61894.106585
35   62012.691406
36   61855.221540
37   61803.599888
38   62090.461496
39   63027.950335
40   63704.094866
Name: BTC_7DAY_MA, dtype: float64
```

1.6 Visualization of BTC Price Varitaions

```
In [17]: fig = px.line(crypto, y=["BTC"] )
fig.show()
```





1.7 Visualization of BTC Moving 7 DaysAverage

```
In [18]: fig = px.line(crypto, y=['BTC_7DAY_MA'] )  
fig.show()
```

```
In [ ]:
```