Assignment 1: Python for Analytics

- covers lectures 1-3
- due: October 18th by 6pm.
- Points will be deducted if:
  - Problems are not completed.
  - Portions of problems are not completed.
  - Third party modules where used when the question specified not to do so.
  - The problem was solved in a very inefficient manner. For instance, copying and pasting the same block of code 10 times instead of using a for loop.
  - Each day late will result in a 10% penalty.
  - Not attemping a problem or leaving it blank will result in 0 points for the problem and an additional 5 point deduction.

In [1]:
```
import sys
```

In [2]:
```
sys.version
```

Out[2]:
```
'3.9.12 (main, Apr  4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]'
```

Question 1 (10 points)

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Find the sum of all the multiples of 3 or 5 below 1000. Note, below 1000.

Do this using 2 methods.

- Using an `if` and `else` condition inside a comprehension.
- Using a udf called inside the comprehension.

Print the resulting sum each time.

Solution for Q1

In [2]:
```
def findSums(n):
    sum = 0
    for i in range(1,n):
        if(i%3 == 0 or i%5 == 0):
            sum+=i
            print(sum)
        else:
            print(sum)
            continue

findSums(1000)
```

```
0
0
3
3
8
14
14
14
23
33
33
45
45
45
60
60
60
78
78
```

Question 2 (10 points)

The below snippet of code will download daily closing prices for SPY, which is an ETF that tracks the price S and P 500.

Using a `for` loop, create a `list` of 5 period moving averages. For instance, the first 5 values of the list are `[315.82, 313.43, 314.62, 313.74, 315.41]` and the average is `314.60` . This means the first entry in our new list would be `314.60` .

Print last 5 items of the list and the sum of the list of 5 period moving averges.

In [3]:
```
! pip install yfinance==0.1.77
```

```
Requirement already satisfied: yfinance==0.1.77 in d:\anaconda\lib\site-package
s (0.1.77)
Requirement already satisfied: appdirs>=1.4.4 in d:\anaconda\lib\site-packages
(from yfinance==0.1.77) (1.4.4)
Requirement already satisfied: pandas>=0.24.0 in d:\anaconda\lib\site-packages
(from yfinance==0.1.77) (1.3.5)
Requirement already satisfied: multitasking>=0.0.7 in d:\anaconda\lib\site-pack
ages (from yfinance==0.1.77) (0.0.11)
Requirement already satisfied: numpy>=1.15 in d:\anaconda\lib\site-packages (fr
om yfinance==0.1.77) (1.21.5)
Requirement already satisfied: requests>=2.26 in d:\anaconda\lib\site-packages
(from yfinance==0.1.77) (2.27.1)
Requirement already satisfied: lxml>=4.5.1 in d:\anaconda\lib\site-packages (fr
om yfinance==0.1.77) (4.8.0)
Requirement already satisfied: python-dateutil>=2.7.3 in d:\anaconda\lib\site-p
ackages (from pandas>=0.24.0->yfinance==0.1.77) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in d:\anaconda\lib\site-packages (f
rom pandas>=0.24.0->yfinance==0.1.77) (2021.3)
Requirement already satisfied: six>=1.5 in d:\anaconda\lib\site-packages (from
python-dateutil>=2.7.3->pandas>=0.24.0->yfinance==0.1.77) (1.16.0)
Requirement already satisfied: idna<4,>=2.5 in d:\anaconda\lib\site-packages (f
rom requests>=2.26->yfinance==0.1.77) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in d:\anaconda\lib\sit
e-packages (from requests>=2.26->yfinance==0.1.77) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in d:\anaconda\lib\site-pa
ckages (from requests>=2.26->yfinance==0.1.77) (1.26.9)
Requirement already satisfied: certifi>=2017.4.17 in d:\anaconda\lib\site-packa
ges (from requests>=2.26->yfinance==0.1.77) (2021.10.8)
```

In [4]:
```
! pip install pandas_datareader
```

```
Requirement already satisfied: pandas_datareader in d:\anaconda\lib\site-packag
es (0.10.0)
Requirement already satisfied: requests>=2.19.0 in d:\anaconda\lib\site-package
s (from pandas_datareader) (2.27.1)
Requirement already satisfied: pandas>=0.23 in d:\anaconda\lib\site-packages (f
rom pandas_datareader) (1.3.5)
Requirement already satisfied: lxml in d:\anaconda\lib\site-packages (from pand
as_datareader) (4.8.0)
Requirement already satisfied: numpy>=1.17.3 in d:\anaconda\lib\site-packages
(from pandas>=0.23->pandas_datareader) (1.21.5)
Requirement already satisfied: pytz>=2017.3 in d:\anaconda\lib\site-packages (f
rom pandas>=0.23->pandas_datareader) (2021.3)
Requirement already satisfied: python-dateutil>=2.7.3 in d:\anaconda\lib\site-p
ackages (from pandas>=0.23->pandas_datareader) (2.8.2)
Requirement already satisfied: six>=1.5 in d:\anaconda\lib\site-packages (from
python-dateutil>=2.7.3->pandas>=0.23->pandas_datareader) (1.16.0)
Requirement already satisfied: charset-normalizer~=2.0.0 in d:\anaconda\lib\sit
e-packages (from requests>=2.19.0->pandas_datareader) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in d:\anaconda\lib\site-packa
ges (from requests>=2.19.0->pandas_datareader) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in d:\anaconda\lib\site-packages (f
rom requests>=2.19.0->pandas_datareader) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in d:\anaconda\lib\site-pa
ckages (from requests>=2.19.0->pandas_datareader) (1.26.9)
```

In [6]:
```
import yfinance as yf
import pandas_datareader
```

In [7]:
```
import pandas as pd
```

In [8]:
```
print(yf.__version__)
print(pandas_datareader.__version__)
print(pd.__version__)
```

```
0.1.77
0.10.0
1.3.5
```

In [9]:
```
SPY = yf.download('SPY')
SPY = yf.Ticker('SPY')
spy_lst = SPY.history(start="2020-01-01", end="2020-02-01")["Close"].tolist()
print(spy_lst[0:5])
print(sum(spy_lst[0:5])/5)
```

```
[**********100%**********]  1 of 1 completed
[311.1170349121094, 308.76116943359375, 309.9391174316406, 309.0676574707031, 3
10.71478271484375]
309.9199523925781
```

Solution for Q2

In [14]:
```
spy_movingAve = []
for i in range(0,len(spy_lst) - 5 + 1):
    movingAve = sum(spy_lst[i:i+5])/5
    spy_movingAve.append(movingAve)

print("The sum of 5 period moving average is: ", sum(spy_movingAve))
print("The last 5 elements in moving average is: ", spy_movingAve[-5:])
```

```
The sum of 5 period moving average is:  5333.346777343751
The last 5 elements in moving average is:  [315.38438720703124, 314.53973999023
435, 313.6356994628906, 312.8619018554688, 311.5135131835938]
```

Question 3 (10 points)

The below `transactions` list represents user-item purchases. For instance, the first element in `transactions` is `["A", "item_a"]`, which indicates `"A"` bought `"item_a"`.

Iterate the list and return a dictionary where the key is the user id, (the ids are `"A", "B", "C", "D"`) and the values are a list of items the user bought.

The desired output for `"A"` and `"D"` can be seen in the `sample_output_dct`.

Be sure your solution is scalable, meaning it should be usable for a list of transactions of any size, and not hard coded.

Hits, this can be achieved by initializing an empty `dict`, using a `for` loop, selecting elements from a list, an `if-else` to check if a user key is in the dictionary and `append`ing items to the values of the dictionary.

Print the dictionary out at the end.

In [45]:
```
transactions = [
    ["A", "item_a"],
    ["B", "item_a"],
    ["C", "item_a"],
    ["C", "item_b"],
    ["C", "item_c"],
    ["B", "item_c"],
    ["D", "item_b"],
    ["D", "item_b"]
]

sample_output_dct = {
    "A": ["item_a"],
    "D": ["item_b", "item_b"]
}
```

Solution for Q3

In [49]:
```
dict_transactions = {}
for item in transactions:
    if item[0] in dict_transactions:
        dict_transactions[item[0]].append(item[1])
    else:
        dict_transactions[item[0]] = [item[1]]

dict_transactions
```

Out[49]:
```
{'A': ['item_a'],
 'B': ['item_a', 'item_c'],
 'C': ['item_a', 'item_b', 'item_c'],
 'D': ['item_b', 'item_b']}
```

Question 4 (10 points)

A string can be sliced just like a list, using the bracket notation. Find the 3 consecutive numbers and their index positions that have the greatest sum in the number `352405531243532354352342143234512821925512043 21` .

As an example, the the string `"1324"` has 2 three number windows, `132` and `324` . The first sums to `6` and the later sums to `9` . Thus the 3 numbers would be `[3,2,4]` and the indices would be `[1,2,3]` .

Do not worry about ties. If there is a tie, you can pick the first or last occurence.

Print out the 3 numbers, the 3 indices where they occur and their sum.

In [1]:
```
sample = "1324"

# results should be
numbers = [3,2,4]
max_val = 9
index_vals = [1,2,3]
```

In [37]:
```
a = "352405531243532354352342143234512821925512043 21"
```

Solution for Q4

In [38]:
```
max_val = 0
numbers = []
index_val = []
for i in range(0,len(a) - 2):
    a1 = int(a[i])
    a2 = int(a[i+1])
    a3 = int(a[i+2])
    sum = a1 + a2 + a3
    if(sum > max_val):
        max_val = sum
        numbers = [int(x) for x in (a[i],a[i+1],a[i+2])]
        index_val = [i for i in (i, i+1, i+2)]

print(max_val)
print(numbers)
print(index_val)
```

```
16
[9, 2, 5]
[36, 37, 38]
```

Quesiton 5 (15 points)

The sum of the squares of the first ten natural numbers is

- `1^2 + 2^2 + ... + 10^2 = 385`

The square of the sum of the first ten natural numbers is

- `(1 + 2 + ... + 10)^2 = 3025`

Hence the difference between the sum of the squares of the first ten natural numbers and the square of the sum is

- `3025 - 385 = 2640`

Create a function, or collection of functions, that find the difference between the square of sums and sum of squares from 1 to `n`. Note, to solve the problem you have to:

- find the sum of squares
- the square of sums
- the difference

This can be broken up into smaller functions, with one function making use of the smaller ones, or all be done in one function.

Add an assert statement to your function, to make sure the input `n` is a positive `int`.

Test the function using `n=12` and `n=8` and print the results.

Solution for Q5

In [71]:
```
def findDifference(n):
    if n <= 0:
        raise Exception("x should be a positive number")

    sumOfSquares = 0
    squareSums = 0
    for i in range(1,n):
        sumOfSquares+=i**2
        squareSums+=i

    squareSums = squareSums**2

    squareDiff = squareSums - sumOfSquares

    return squareDiff

print(findDifference(12))
print(findDifference(8))
print(findDifference(-2))
```

```
3850
644

---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
Input In [71], in <cell line: 19>()
     17 print(findDifference(12))
     18 print(findDifference(8))
---> 19 print(findDifference(-2))

Input In [71], in findDifference(n)
      1 def findDifference(n):
      2     if n <= 0:
----> 3         raise Exception("x should be a positive number")
      5     sumOfSquares = 0
      6     squareSums = 0

Exception: x should be a positive number
```

Question 6 (20 points)

Make a function, or group of functions, to find outlier datapoints in a list. The outlier should be based on the standard deviation, giving the user some ability to control the outlier threshold. For instance, setting 2 standard deviations or 3 from the mean should be possible. Note, to solve this problem you will have to:

- find the mean of a list
- find the standard deviation fo a list
- convert the list to zscores using `(x-mean)/std`
- find the indices of where the outlier datapoints are, using the zscores
- return the outlier values and the indicies they occur at.

Test your data using the below stock price data for TSLA. Keep the same data range as is coded in below.

The standard deviation can be calculated as such (https://numpy.org/doc/stable/reference/generated/numpy.std.html (https://numpy.org/doc/stable/reference/generated/numpy.std.html)):

- `std = sqrt(mean(abs(x - x.mean())**2))`

Again, this can be done in one big function or a collection of smaller ones that are then used inside a final function to find the outliers.

NOTE: ASIDE FROM CHECKING WORK, THE ONLY PIECE OF IMPORTED CODE TO BE USED IS `sqrt` from `math` and the imported data from `yfinance`. You may use `numpy` and `scipy` functionality to check your standard deviation and zscore calculations, but only to check your work.

Print out the average, standard deviation, outliers and the index position of where the outliers occur.

In [16]:
```
from math import sqrt
```

In [20]:
```
TSLA = yf.download('TSLA')
TSLA = yf.Ticker('TSLA')
tsla_lst = TSLA.history(start="2019-01-01", end="2020-04-01")["Close"].tolist()
tsla_lst[0:5]
```

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

Out[20]:
```
[20.674667358398438,
 20.02400016784668,
 21.179332733154297,
 22.33066749572754,
 22.356666564941406]
```

Solution for Q6

Since we use zscores to find the outliers, we accept the assumption that the data follows a normal distribution. Therefore, according to the empirical rule of normal distribution, the 2 sigma from the mean takes 95% of the distribution, which indicates that the zscore of the outliers should be either greater than 1.96 or less than -1.96

In [35]:
```
def findOutlier(x):
    pos = []
    my_outlier = []
    my_mean = sum(x)/len(x)
    sse_lst = [abs(a - my_mean)**2 for a in x]
    my_std = sqrt(sum(sse_lst)/len(sse_lst))
    my_zscores = [(i - my_mean)/my_std for i in x]
    for zscore in my_zscores:
        if zscore > 1.96 or zscore < -1.96:
            my_index = my_zscores.index(zscore)
            pos.append(my_index)
            my_outlier.append(x[my_index])

    print("The mean is: ", my_mean)
    print("The standard deviation is: ", my_std)
    print("The outlier is: ", my_outlier)
    print("The index is: ", pos)


findOutlier(tsla_lst)
```

```
The mean is:  22.820205961822705
The standard deviation is:  10.679462063209662
The outlier is:  [52.0, 59.137332916259766, 48.97999954223633, 49.9306678771972
66, 49.871334075927734, 51.41866683959961, 51.62533187866211, 51.1526679992675
8, 53.599998474121094, 53.33533477783203, 57.22666549682617, 61.16133117675781,
59.96066665649414, 60.06666564941406, 55.58599853515625, 53.32733154296875, 51.
91999816894531, 45.266666412353516, 44.53266525268555, 49.574668884277344, 49.7
0066833496094, 49.96666717529297, 48.30266571044922, 46.89866638183594]
The index is:  [273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 28
5, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296]
```

Question 7 (25 points)

Make a class to find the summary statistics of a list of data. Include the below methods:

- `__init__` should take a param `data` , representing the list of data. Be sure to use `self` to bind the `data` attribute to the instance of your class.
- `Returns` an `n` period moving average.
- `Returns` the median.
- `Returns` the mean.
- Finds the min and creates a new attribute called `lst_min` .
- Finds the max and creates a new attribute called `lst_max` .

When calculating summary statistics, be sure to use the `data` attribute created in the `__init__` method.

Make an instance of your class using `tst_lst_of_data` . Run the below tests

- **calculate the 3 period moving average of the list and print the first 3 values of the returned list**
- **get the median and print out the median**
- **run the min and max methods, and print out the attributes `lst_min` and `lst_max`**

In [75]:
```
tst_lst_of_data = [
    62.02,
    60.07,
    63.53,
    66.99,
    67.06,
    67.70,
    68.99,
    69.45,
    66.87,
    68.88,
    69.20,
    69.46,
    60.45,
    59.78,
    57.51
]
```

Solution for Q7

In [109]:
```
import statistics as sts

class summaryStatistics:
    def __init__(self,data):
        self.data = data
        self.lst_min = 0
        self.lst_max = 0

    def movingAve(self,n):
        movingAverage = []
        for a in range(0,len(self.data) - n + 1):
            sumOfn = 0
            for a in self.data[a:a+n]:
                sumOfn += a
            movingAverage.append(sumOfn/n)
        return movingAverage

    def findMedian(self):
        my_median = sts.median(self.data)
        return my_median

    def findMean(self):
        my_mean = sts.mean(self.data)
        return my_mean

    def findMin(self):
        self.lst_min = min(self.data)

    def findMax(self):
        self.lst_max = max(self.data)


p = summaryStatistics(tst_lst_of_data)
print("First 3 elements in moving average is: ", p.movingAve(3)[0:3])
print("Median is: ", p.findMedian())
print("Mean is: ", p.findMean())
p.findMin()
print("Min is: ", p.lst_min)
p.findMax()
print("Max is: ", p.lst_max)
```

```
First 3 elements in moving average is:  [61.873333333333335, 63.52999999999994, 65.86]
Median is:  66.99
Mean is:  65.19733333333333
Min is:  57.51
Max is:  69.46
```