Homework 2

Name: < Jason Lubrano >

This assignment is due on Moodle by **9:30am on Thursday, September 19th 2019**. Submit only this Jupyter notebook to Moodle. Do not compress it using tar, rar, zip, etc. Your solutions to analysis questions should be done in Markdown directly below the associated question. Remember that you are encouraged to discuss the problems with your instructors and classmates, but **you must write all code and solutions on your own**.

The rules to be followed for the assignment are:

- Do **NOT** load or use any Python packages that are not available in Anaconda 3.6.
- Some problems with code may be autograded. If we provide a function or class API **do not** change it.
- Do not change the location of the data or data directory. Use only relative paths to access the data.

```python
import pandas as pd
import numpy as np
import pickle
from pathlib import Path
import math
import matplotlib.pyplot as plt
```

[50 points] Problem 1

There are two functions that need to be completed:

normalization(fname, attr, normType)

- This function takes in the location of the data file, the attribute that has to be normalised (one of the values from 'open','high','low','close','volume') and the type of normalization to be performed('min_max' or 'z_score')
- Based on the normalisation type that is mentioned, you will have to apply the appropriate formula and return a dictionary where key = original value in the dataset, value = normalised value

correlation (fname1, attr1, fname2, attr2)

- This function takes in the location of the first data file, the attribute that has to be used in the first file, the location of the second data file and the attribute that has to be used in the second file.
- This function has to calculate the correlation coefficient between the two attributes mentioned in the two files.

Note:

- Download both the data files and store it in the location "data/dataset_1.csv" and "data/dataset_2.csv" . Please maintain this as it would be necessary while grading.
- Do not change the variable names of the returned values.
- If the test case fails, one way to debug is to see the output of the testing data and comparing it to your output.
- Initially the test case will be failed as there is no code in the below two functions.

```python
# dataset_1.csv -> AMZN 5 YR
# dataset_2.csv -> FB 5 yr
```

```python
def normalization (fname, attr, normType):
    '''
    Input Parameters:
        fname: Name of the csv file contiaining historical quotes
        attr: The attribute to be normalized
        normType: The type of normalization
    Output:
        a dictionary where each key is the original column value and each value is the normalised column value.
    '''

    result = {}

    #TODO: Write code given the Input / Output Paramters.

    # making data frame from csv file
    data = pd.read_csv(fname)
    df = pd.DataFrame(data)
    df.dropna()

    # colum
    dfAttr = df[attr]
    # print(dfAttr.head())

    if(normType == "min_max"):
        # print("MIN MAX")
        dfAttrNorm=(dfAttr-dfAttr.min())/(dfAttr.max()-dfAttr.min())
        # print(dfAttrNorm)
        result = dict(zip(dfAttr, dfAttrNorm))

    elif (normType == "z_score"):
        # print("Z SCORE")
        dfAttrZscr = (dfAttr - dfAttr.mean())/dfAttr.std(ddof=0)
        # print(dfAttrZscr)
        result = dict(zip(dfAttr, dfAttrZscr))
    else:
        result = {0,0}

    return result
```

```python
# normalization ("data/dataset1.csv", "high", "min_max")
```

```python
# normalization ("data/dataset1.csv", "high", "z_score")
```

```python
def correlation (fname1, attr1, fname2, attr2):
    '''
    Input Parameters:
        fname1: name of the first csv file containing historical quotes
        attr1: The attribute to consider in the first csv file (fname1)
        fname2: name of the second csv file containing historical quotes
        attr2: The attribute to consider in the second csv file (fname2)

    Output:
        correlation coefficient between attr1 in fname1 and attr2 in fname2
    '''

    correlation_coefficient = 0.0

    #TODO: Write code given the Input / Output Paramters.

    # making data frame from csv file
    data1 = pd.read_csv(fname1)
    df1 = pd.DataFrame(data1)
    df1.dropna()
    dfAttr1 = df1[attr1]
    # print(dfAttr1.count())

    data2 = pd.read_csv(fname2)
    df2 = pd.DataFrame(data2)
    df2.dropna()
    dfAttr2 = df2[attr2]
    # print(dfAttr2.count())

    corrcoef = np.corrcoef(x=dfAttr1, y=dfAttr2, rowvar=True)
    correlation_coefficient = corrcoef[0][1]

    return correlation_coefficient
```

```python
correlation ("data/dataset1.csv", "volume", "data/dataset1.csv", "high")
```

```
0.12368429735885123
```

```python
import unittest

class TestKnn(unittest.TestCase):
    def setUp(self):
        self.loc1 = "data/test1.csv"
        self.loc2 = "data/test2.csv"
        file = open('data/testing_normalization', 'rb')
        self.data_normalization = pickle.load(file)
        file.close()
        file = open('data/testing_correlation', 'rb')
        self.data_correlation = pickle.load(file)
        file.close()

    def test0(self):
        """
        Test the label counter
        """
        result = normalization(self.loc2,"open","min_max")
        for key,value in self.data_normalization.items():
            self.assertAlmostEqual(result[key],value, places = 1)

    def test1(self):
        """
        Test the label counter
        """
        result = correlation(self.loc1,"close",self.loc2,"close")
        self.assertAlmostEqual(result,self.data_correlation, places = 1)

tests = TestKnn()
tests_to_run = unittest.TestLoader().loadTestsFromModule(tests)
unittest.TextTestRunner().run(tests_to_run)
```

```
..
----------------------------------------------------------------------
Ran 2 tests in 0.026s

OK

<unittest.runner.TextTestResult run=2 errors=0 failures=0>
```

[50 points] Problem 2

There are 4 functions that need to be completed:

1. For each of the graphs, the input function parameters and the expected output has been mentioned below.

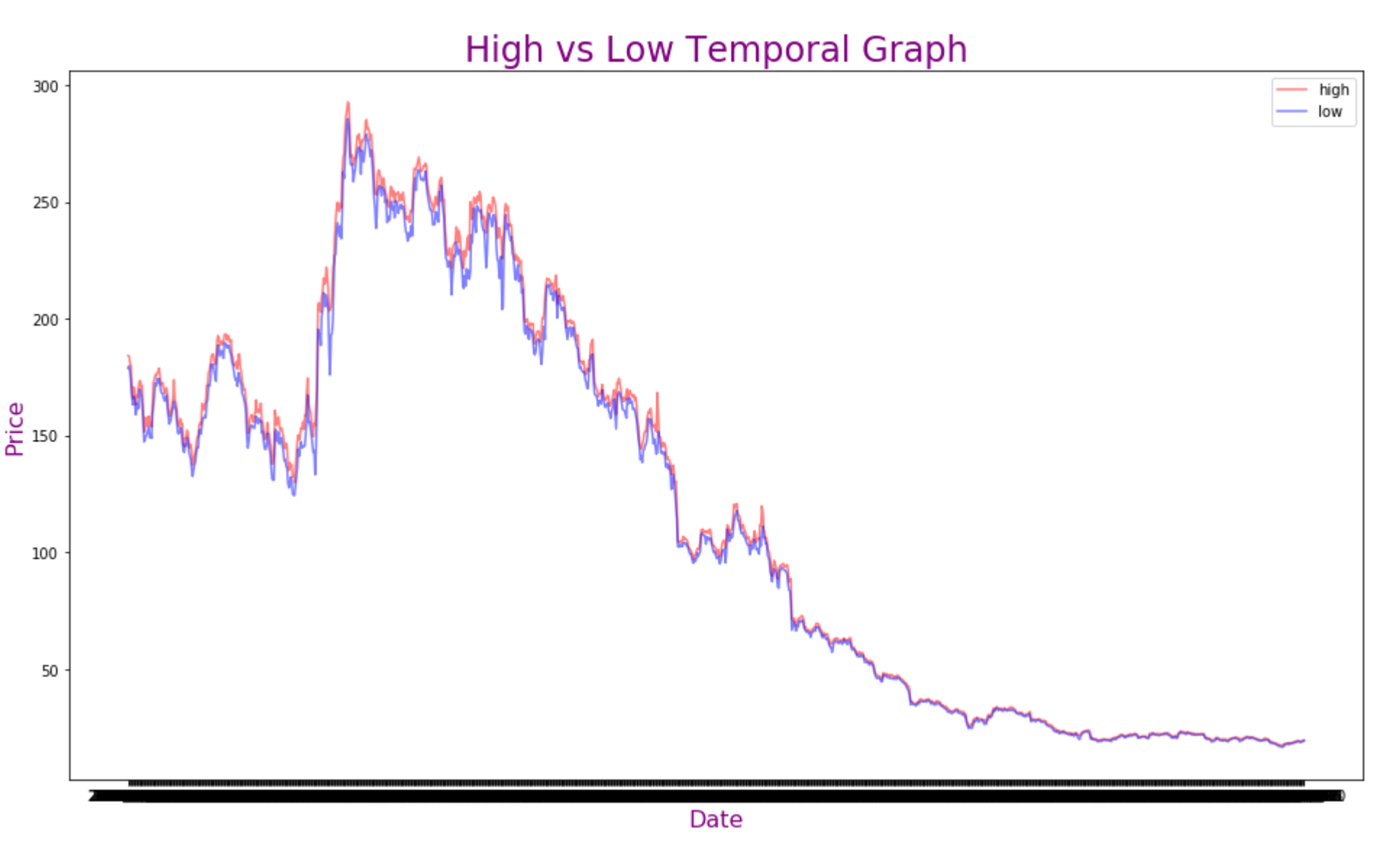
Note:

- Make sure the dataset you are using is the one mentioned in the problem statement in moodle. The link has been provided.
- After defining your functions. Create another block to call these functions by passing the attributes mentioned in moodle.

```python
import matplotlib
import matplotlib.dates as mdates
# Plot size to 14" x 7"
matplotlib.rc('figure', figsize = (14, 7))
# Font size to 14
matplotlib.rc('font', size = 14)
# Do not display top and right frame lines
matplotlib.rc('axes.spines', top = False, right = False)
# Remove grid lines
matplotlib.rc('axes', grid = False)
# Set backgound color to white
matplotlib.rc('axes', facecolor = 'white')
```

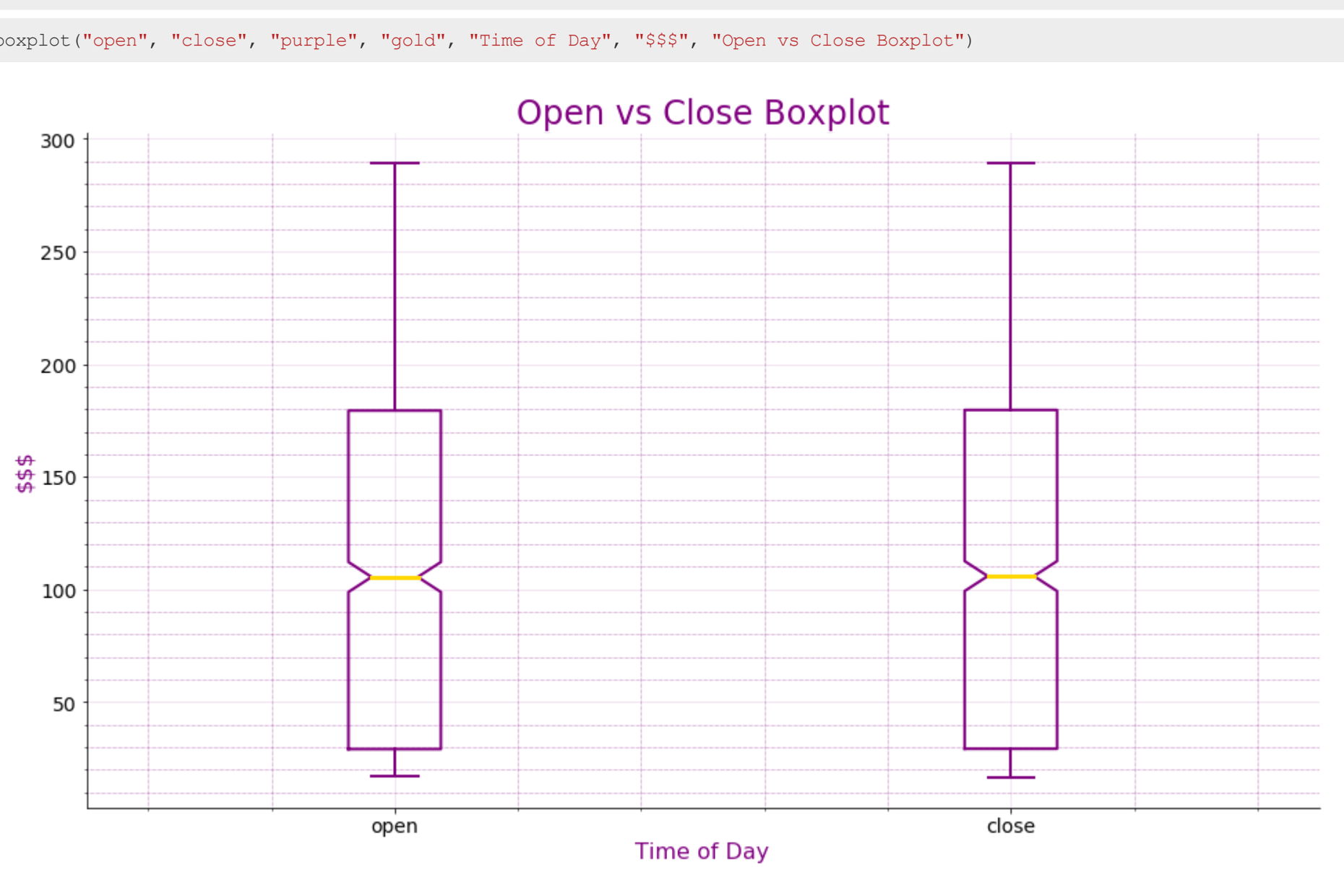
```python
def temporal_graph(x_data,y_data,xlabel,ylabel,title):
    '''Input : x_data and y_data are the lists containing the data points for x and y axis
    xlabel and ylabel are the labels that should be given to the corresponding axes
    title contains the title of the graph

    Output : A temporal graph displayed'''

    # making data frame from csv file
    data = pd.read_csv("data/NVDA-HistoricalQuotes.csv")
    df = pd.DataFrame(data)
    df.dropna()
    # Used for testing purposes
    # print(df.head())

    dfhigh = df[[x_data]]
    dflow = df[[y_data]]


    # print(df_tgraph.head())

    fig, ax = plt.subplots(
        nrows=1,
        ncols=1,
        figsize=(16,9)
    )

    plt.plot(
        df["date"],
        dfhigh[x_data],
        alpha=0.5,
        color="red",
        label=x_data
    )
    plt.plot(
        df["date"],
        dflow[y_data],
        alpha=0.5,
        color="blue",
        label=y_data
    )

    ax.set_xlabel(
        xlabel,
        fontsize=16,
        color="purple"
    )

    ax.set_ylabel(
        ylabel,
        fontsize=16,
        color="purple"
    )

    ax.set_title(
        title,
        fontsize=24,
        color="purple"
    )

    ax.grid(
        alpha=0
    )

    ax.set_axisbelow(
        True
    )

    plt.legend()

    plt.show()
```

```python
temporal_graph("high", "low", "Date", "Price", "High vs Low Temporal Graph")
```



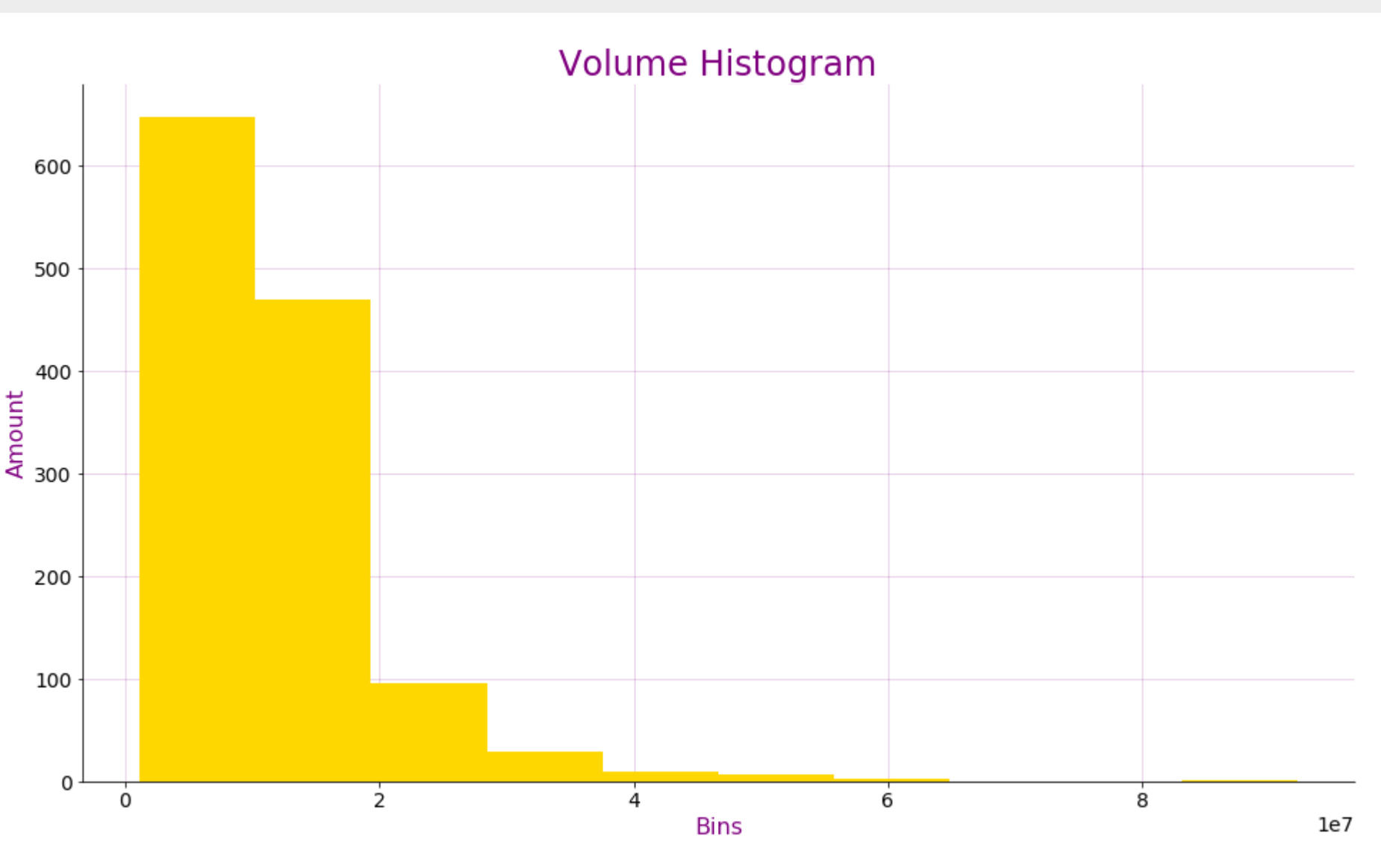

```python
def boxplot(x_data,y_data, base_color, median_color, x_label, y_label, title):
    '''Input : x_data and y_data are the lists containing the data points for x and y axis
    base_color and median_color can be used to set colors in the graph.
    xlabel and ylabel are the labels that should be given to the corresponding axes
    title contains the title of the graph.

    Output : A boxplot displayed'''
    '''Input : x_data and y_data are the lists containing the data points for x and y axis
    xlabel and ylabel are the labels that should be given to the corresponding axes
    title contains the title of the graph

    Output : A temporal graph displayed'''

    # making data frame from csv file
    data = pd.read_csv("data/NVDA-HistoricalQuotes.csv")
    df = pd.DataFrame(data)
    df.dropna()
    # Used for testing purposes
    # print(df.head())

    df_tgraph = df[[x_data, y_data]]
    # print(df_tgraph.head())

    fig, ax = plt.subplots(
        nrows=1,
        ncols=1,
        figsize=(16,9)
    )

    bp = df_tgraph.boxplot(
        column=[x_data, y_data],
        showfliers=True,
        notch=True,
        return_type='dict'
    )

    for element in ['boxes', 'whiskers', 'fliers', 'means', 'caps']:
        plt.setp(bp[element], color=base_color, linewidth=2)

    for element in ['medians']:
        plt.setp(bp[element], color=median_color, linewidth=3)

    ax.set_xlabel(
        x_label,
        fontsize=16,
        color=base_color
    )

    ax.set_ylabel(
        y_label,
        fontsize=16,
        color=base_color
    )

    ax.set_title(
        title,
        fontsize=24,
        color=base_color
    )

    ax.grid(
        color=base_color,
        alpha=0.2
    )

    ax.set_axisbelow(
        True
    )

    ax.minorticks_on()

    ax.grid(which='major', linestyle='-', linewidth='0.5', color=base_color)

    ax.grid(which='minor', linestyle=':', linewidth='0.5', color=base_color)
    #plt.legend()

    plt.show()
```

```python
boxplot("open", "close", "purple", "gold", "Time of Day", "$$$", "Open vs Close Boxplot")
```




```python
def histogram(data, x_label, y_label, title):
    '''Input : data is the list containing the data points for histogram buckets
    xlabel and ylabel are the labels that should be given to the corresponding axes
    title contains the title of the graph

    Output : A histogram displayed

    '''

    # making data frame from csv file
    filer = pd.read_csv("data/NVDA-HistoricalQuotes.csv")
    df = pd.DataFrame(filer)
    df.dropna()
    # Used for testing purposes
    # print(df.head())

    # print(df_tgraph.head())
    df_tgraph = df[[data]]

    fig, ax = plt.subplots(figsize=(16,9))

    #hist = df_tgraph.hist(bins=10, color="purple")

    plt.hist(df_tgraph[data], bins=10, color='gold')

    ax.set_xlabel(x_label, fontsize=16, color="purple")

    ax.set_ylabel(y_label,fontsize=16, color="purple")

    ax.set_title(title, fontsize=24, color="purple")

    ax.grid(color="purple", alpha=0.2)

    ax.set_axisbelow(True)

    #plt.legend()

    plt.show()
```

```python
histogram("volume", "Bins", "Amount", "Volume Histogram")
```




```python
def amzn_new_plot(x_data,y_data,a_data,b_data,xlabel,ylabel,title):
    '''Define this function as you would seem fit to display the plot that interests you using
    the same dataset. Define your function parameters and display the resulting plots'''

    data = pd.read_csv("data/NVDA-HistoricalQuotes.csv")
    df = pd.DataFrame(data)
    df.dropna()
    # Used for testing purposes
    # print(df.head())

    timemask = (df["date"] > '2018/12/31')

    YTD = timemask
    dfhigh = timemask
    dflow = timemask
    dfopen = timemask
    dfclose = timemask

    # print(df_tgraph.head())

    fig, ax = plt.subplots(
        nrows=1,
        ncols=1,
        figsize=(16,9)
    )

    plt.plot(
        df.loc[YTD]["date"],
        df.loc[dfhigh][x_data],
        alpha=0.5,
        color="red",
        label=x_data
    )
    plt.plot(
        df.loc[YTD]["date"],
        df.loc[dflow][y_data],
        alpha=0.5,
        color="blue",
        label=y_data
    )

    plt.plot(
        df.loc[YTD]["date"],
        df.loc[dfopen][a_data],
        alpha=0.5,
        color="orange",
        label=a_data
    )

    plt.plot(
        df.loc[YTD]["date"],
        df.loc[dfclose][b_data],
        alpha=0.5,
        color="green",
        label=b_data
    )

    ax.set_xlabel(
        xlabel,
        fontsize=16,
        color="purple"
    )

    ax.set_ylabel(
        ylabel,
        fontsize=16,
        color="purple"
    )

    ax.set_title(
        title,
        fontsize=24,
        color="purple"
    )

    ax.grid(
        alpha=0
    )

    ax.set_axisbelow(
        True
    )

    plt.legend()

    plt.show()
```

```python
amzn_new_plot("high", "low", "open", "close", "Date", "Price", "High, Low, Open, Close YTD Temporal Graph")
```