```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Oct  8 19:15:17 2018

@author: mohit
"""

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib.dates import date2num
import matplotlib.pyplot as plt
import requests
import scipy
import lxml
import datetime
from datetime import date
from nsepy import get_history



#retriving tcs and infy data
data = get_history(symbol="TCS", start=date(2015,1,1), end=date(2015,12,31))
data[['Close']].plot()

data1 = get_history(symbol="INFY", start=date(2015,1,1), end=date(2015,12,31))
data1[['Close']].plot()


# there are at least four variables involved for each date (open, high, low, and close),


from matplotlib.dates import DateFormatter, WeekdayLocator,\
DayLocator, MONDAY
from matplotlib.finance import candlestick_ohlc

def pandas_candlestick_ohlc(dat, stick = "day", otherseries = None):

    mondays = WeekdayLocator(MONDAY)
    alldays = DayLocator()
    dayFormatter = DateFormatter('%d')


    transdat = dat.loc[:,["Open", "High", "Low", "Close"]]
    if (type(stick) == str):
        if stick == "day":
            plotdat = transdat
            stick = 1
        elif stick in ["week", "month", "year"]:
            if stick == "week":
                transdat["week"] = pd.to_datetime(transdat.index).map(lambda x: x.isocalendar()
            elif stick == "month":
                transdat["month"] = pd.to_datetime(transdat.index).map(lambda x: x.month) # Ide
            transdat["year"] = pd.to_datetime(transdat.index).map(lambda x: x.isocalendar()[0]
            grouped = transdat.groupby(list(set(["year",stick]))) # Group by year and other ap
            plotdat = pd.DataFrame({"Open": [], "High": [], "Low": [], "Close": []}) # Create
            for name, group in grouped:
                plotdat = plotdat.append(pd.DataFrame({"Open": group.iloc[0,0],
                                            "High": max(group.High),
                                            "Low": min(group.Low),
                                            "Close": group.iloc[-1,3]},
                                           index = [group.index[0]]))
            if stick == "week": stick = 5
            elif stick == "month": stick = 30
            elif stick == "year": stick = 365

    elif (type(stick) == int and stick >= 1):
        transdat["stick"] = [np.floor(i / stick) for i in range(len(transdat.index))]
        grouped = transdat.groupby("stick")
        plotdat = pd.DataFrame({"Open": [], "High": [], "Low": [], "Close": []}) # Create empty
        for name, group in grouped:
            plotdat = plotdat.append(pd.DataFrame({"Open": group.iloc[0,0],
                                        "High": max(group.High),
                                        "Low": min(group.Low),
                                        "Close": group.iloc[-1,3]},
                                       index = [group.index[0]]))

    else:
        raise ValueError('Valid inputs ')


   # Set plot parameters, including the axis object ax used for plotting
    fig, ax = plt.subplots()
    fig.subplots_adjust(bottom=0.2)
    if plotdat.index[-1] - plotdat.index[0] < pd.Timedelta('365 days'):
        weekFormatter = DateFormatter('%b %d')  # e.g., Jan 12
        ax.xaxis.set_major_locator(mondays)
        ax.xaxis.set_minor_locator(alldays)
    else:
        weekFormatter = DateFormatter('%b %d, %Y')
    ax.xaxis.set_major_formatter(weekFormatter)

    ax.grid(True)

# Create the candelstick chart
    candlestick_ohlc(ax, list(zip(list(date2num(plotdat.index.tolist())), plotdat["Open"].tolis
                      plotdat["Low"].tolist(), plotdat["Close"].tolist())),
                      colorup = "black", colordown = "red", width = stick * .4)
# Plot other series (such as moving averages) as lines
    if otherseries != None:
        if type(otherseries) != list:
            otherseries = [otherseries]
        dat.loc[:,otherseries].plot(ax = ax, lw = 1.3, grid = True)

    ax.xaxis_date()
    ax.autoscale_view()
    plt.setp(plt.gca().get_xticklabels(), rotation=75, horizontalalignment='right')

    plt.show()
    pandas_candlestick_ohlc(apple)


stocks.plot(secondary_y = ["INFY", "TCS"], grid = True)

# selecting end and start date

startdate = pd.to_datetime("2015-1-1").date()
enddate = pd.to_datetime("2015-12-31").date()

# df.apply(arg) will apply the function arg to each column in df, and return a DataFrame with
# Recall that lambda x is an anonymous function accepting parameter x; in this case, x will be
stock_return = stocks.apply(lambda x: x / x[0])
stock_return.head()


stock_return.plot(grid = True).axhline(y = 1, color = "black", lw = 2)
```

```
stock_change = stocks.apply(lambda x: np.log(x) - np.log(x.shift(1))) # shift moves dates back
stock_change.head()


data1=data
data1["28d"] = np.round(data["Close"].rolling(window = 28, center = False).mean(), 2)
pandas_candlestick_ohlc(data.loc[startdate:enddate], otherseries = "28d")



tcs["28d"] = np.round(tcs["Close"].rolling(window = 28, center = False).mean(), 2)
pandas_candlestick_ohlc(tcs.loc[startdate:enddate], otherseries = "28d")


nifty = get_history(symbol="NIFTY IT",
                            start=date(2015,1,1),
                            end=date(2015,12,31),index=True)
nifty1=nifty
mix = get_history("TCS", "INFY", start=date(2015,1,1), end=date(2015,12,31))
mix["4w"] = np.round(apple["Close"].rolling(window = 28, center = False).mean(), 2)

pandas_candlestick_ohlc(mix.loc['2015-01-01':'2015-12-31',:], otherseries = "4w")








#part 2 visualization run this command alone

data
stock_fut = get_history(symbol="TCS",
                        start=date(2015,1,1),
                        end=date(2015,12,31),
                        futures=True,
                        expiry_date=date(2015,12,31))
stock_fut.Close.plot()
plt.show()

data1
stock_fut = get_history(symbol="TCS",
                        start=date(2015,1,1),
                        end=date(2015,12,31),
                        futures=True,
                        expiry_date=date(2015,12,31))
stock_fut.Close.plot()
plt.show()
```