lesson 03

Working with Pandas

Python for Financial Analysis

Rajah Chacko

elvtr

Syllabus Review

1. Introduction to Python: Python in Finance
2. Python Basic Syntax: Importing Libraries
3. **Working with Pandas**
4. Pandas Underneath the Hood: Working with NumPy
5. Data Wrangling and Visualization
6. Extracting Financial Insights from Charts and Graphs
7. Financial Calculations with Python: Part 1
8. Financial Calculations with Python: Part 2
9. CAPM and Portfolio Management
10. Linear Regression
11. Time Series Analysis
12. Algorithmic Trading

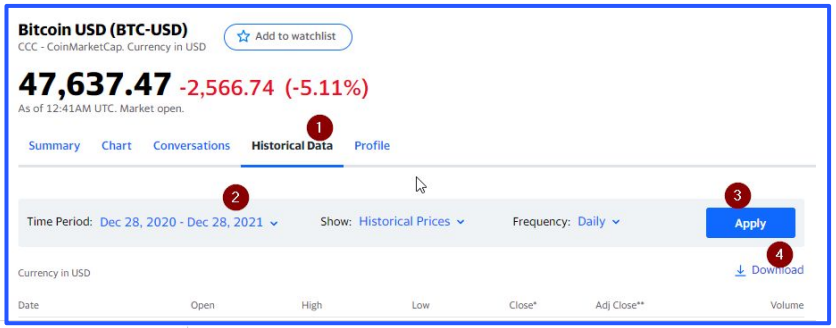
Bonus Class: Cryptocurrency Beyond the Basics with a Fintech Guest Speaker

Class agenda

- Importing stock data
- Working with DataFrames
- Importing from Excel or .csv files.
- Powerful filters and indexes
- Pythonic: Functions, class, and subclass

Importing stock data

- Let's download a Comma-Separated Variable (CSV) file.
  - A. Start at finance.yahoo.com
  - B. Search bar: enter a favorite (e.g., AAPL, TSLA, AMZN, BTC-USD, ^SPX)
  - C. Select the Historical Data tab, change time to 1Y, Click Apply,, and then Download
  - D. Save the CSV file to the directory where you're working. Feel free to peek!



Importing stock data using an API

- The Application Program Interface: skip the browser and straight to Python
- Will use tiingo as an example.
- https://readthedocs.org/projects/tiingo-python/downloads/pdf/latest/
    A. Sign up on tiingo for a (free) account
    B. Pip install tiingo (at the Anaconda prompt)
    C. Follow the pattern in the above PDF document

Working with Pandas DataFrames

- The import statement
  A. import pandas as pd
- Using Pandas to read the csv into a dataframe
  A. tsla = pd.read_csv('TSLA.csv')
  B. Reads the first line as a header.
- Printing the head, the tail, or the whole thing
  A. With head(), tail(), or just the df name.

But will it work for Excel?

- You bet.
- Excel files also have worksheets. You need to tell it which worksheet to import.
- Note that it'll read .xls and .xlsx files even if you don't have MS Excel installed.
- Can also read from OpenOffice (odf, ods and odt files).

More fun with pandas

- Coercing data types
- len(df) tells us how many rows
- Df.columns tells us the column names
- Powerful filters and indexes
    A. Df[df['col_name'] == x]
    B. Df.col_name as an alternative
- Indexing
    A. Often one level.
    B. Indexing by datetime

Functions

- Building Functions
  - A. Encapsulation
    - a. Well-defined parameters and return values
    - b. You've got one job
    - c. What happens in functions stays in functions
    - d. Avoids cut & paste (and code smell)
  - B. Build simple functions greet and double
  - C. Three bond prices: does this look easier?
- Calling functions
  - D. We've already called functions with like sqrt(), head(), and set_index()
  - E. We set parameters on the way in and get a return value out

Classes, and Subclasses

- Classes are factories for creating Python objects
    A. The objects get initialized
    B. They have state (like a name and an age)
    C. They have methods (the functions we just learned about)
    D. Self keyword
    E. Change of perspective
        a. The original syntax for a function call, print_time(current_time), suggests that the function is the active agent. It says something like, “Hey, print_time! Here’s an object for you to print.”
        b. In object-oriented programming, the objects are considered the active agents. An invocation like current_time.print_time() says “Hey current_time! Please print yourself!”
- Subclasses inherit from their parent class
    F. They inherit methods and state (and can add or change their own)
    G. They help us add new items without breaking existing ones
    H. (Example: a ConvertibleBond inherits from Bond)

Assignment #3

From a website, read in a year's stock price data for three stocks that interest you. Write the price data to three CSV files (or an Excel or OpenOffice file with three worksheets.)
Read the prices into a dataframe and calculate some stats about it, such as its high, low, and average. I recommend that you read in one stock's data and calculate the stats, and then modify your code into a function so that you call it three times. Don't copy and paste code!

Take home Python (optional):

Create a method that takes reads from the API based on the ticker (instead of creating the spreadsheets manually). Print the stats for the same tickers (and time period). If the stats differ, analyze why.

Resources
(part 1)

- Reading CSV and Excel files

  https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html

  https://pandas.pydata.org/docs/reference/api/pandas.read_excel.html

- Tiingo

  https://readthedocs.org/projects/tiingo-python/downloads/pdf/latest/

- Writing to a file

  https://www.geeksforgeeks.org/writing-to-file-in-python/

Resources
(part 2)

- Coercing columns to datetime

  https://stackoverflow.com/questions/26763344/convert-pandas-column-to-datetime

  https://www.delftstack.com/howto/python-pandas/how-to-convert-dataframe-column-to-datetime-in-pandas/

- Titanic CSV

  https://github.com/datasciencedojo/datasets/blob/master/titanic.csv

  https://www.geeksforgeeks.org/difference-between-loc-and-iloc-in-pandas-dataframe/

- Classes

  https://openbookproject.net/thinkcs/python/english3e/classes_and_objects_I.html

Q&A