FBA | QUANTITATIVE FINANCE RESEARCH GROUP

Financial AI

Homework 3

Due at 06:00 pm (Korea Standard Time) on Saturday, September 3.

Submit one file: written solutions with executable Python code

Problem 1. Download the daily adjusted closing price for the stocks in constituents.csv from June 11, 2012, through June 9, 2022. You can download this from Yahoo Finance in Python.

(a) Remove the symbols that has NA values more than 5 % of 2521 daily prices. Additionally, remove the dates with NA values

(b) Assume these are daily prices on the trading days from June 12, 2012, through June 9, 2022. Plot the close price series and a histogram of the close prices of AAPL, GOOG, and TSLA.

Problem 2. Employees and Managers

(a) For this part, you will need the Employee3.py and Manager.py modules and the problem2.py source code file.

In problem2.py, you will see that part (a) is written for you: it creates two Employee objects and two Manager objects, displays these four objects, then creates a company (a list) containing these four objects, and finally displays the items in the company.

Execute this code to confirm that it works as you would expect.

(b) Parts (b) and later are "commented out" using a pair of triple single quotes. Move the upper triple single quotes below (b) and above (c).

What we want to do in part (b) is to give raises to all of company's Employees (including Managers: a Manager *is a kind of* Employee). But we want to give a 10% raise to each Manager and a 3% raise to each "ordinary" Employee.

Learn about the isinstance() function, and replace the commented pseudocode with working Python code. Then, test that your code for part (b) works as you would expect.

(c) You can enable equality and relational operator symbols (==, !=, <, <=) to work with objects of a class by defining the special methods __eq__(), __ne__(), __lt__(), and so forth within the Employee class.

Define these methods to compare Employee objects by rate (ignore the name and id). Once you have added these methods to your Employee class, move the upper triple single quotes below (c) and above (d). Test that the code for part (c) works as you would expect.

(d) We use a *class variable* within Employee to count the number of Employee objects (including Manager objects, since a Manager is a kind of Employee) that have been created so far.

We would also like to know how many Employee objects *currently exist*. That is, if we create an Employee object, the number of Employee objects that have been created should be incremented, *and* the number of Employee objects that currently exist should also be incremented.

If we then use del to delete that Employee object, the number of Employee objects that have been created so far will remain unchanged, *but* the number of Employee objects that currently exist should be *decremented*.

If an object is created *locally* within a function definition, then that object will automatically be deleted when that function returns, even if we do not explicitly use a del statement. So in the temp_job() function definition in part 1.d, four Employee objects are created when the function is called, and those four Employee objects are automatically deleted when the function returns.

Learn about the __del__() special method that automatically gets called when an object is deleted. Add a class variable to keep track of the number of Employee objects that currently exist. Add code to increment this variable within __init__() and to decrement this variable within __del__(). Add a get method that returns the value of this class variable.

Eliminate the triple single quotes that enclose the code for part (d), then test that the code for part (d) works as you would expect.