

COSC 101C Homework 2: Fall 2024

The due date for this homework is **Thursday, September 19th, 11:59pm EDT**

Introduction

This assignment is designed to give you practice with the following new topics:

- Procedural decomposition and composition
- Function mechanisms: passing arguments and return
- Module functions

Important Tips

- Remember, computer science is a science. Always write code with a prediction in mind. While you can sparingly code to try and see output, you should focus on thinking through what you want your code to do, what you expect as a result, and compare what your code actually does to your result.
- Before you type, trace the execution of the starter code already provided
- Sometimes sub-tasks go together, like computation followed by printing. You should consider them simultaneously if that helps you follow the correctness of your work.
- Use extra steps if needed for your thought process, printing results to test your code. Just remember to remove these extra prints once you know your code is correct.
- Match your output to the output given. Precision is important in computer science.
- Don't change file names. You're generally given `.py` files to work in. Don't change the filenames of those files.

Your assignment

Your task is to complete following steps:

1. Download the `hw2.zip` file and open it. You will see three python files, `hw2_bill.py`, `hw2_house.py`, and `hw2_stock.py` in the unzipped folder. You are expected to write your programs in these files.
2. Complete `hw2_stock.py`. This file is used in **Part 1**.
3. Complete `hw2_house.py`. This file is used in **Part 2**.
4. Complete `hw2_bill.py`. This file is used in **Part 3**.
5. Review the grading criteria at the end of this assignment.
6. Submit your completed programs.

Notice that each starter `.py` file has a header with some information for you to fill in. Please do so. Your feedback helps the instructors better understand your experiences doing the homeworks and where we can provide better assistance.

Grading

When we are assessing your code, higher levels of achievement are demonstrated, in part, by

- Starter code left unmodified
- All lines of output are present
- Lines of output have proper formatting, including spacing and blank lines
- Use spaces and blank lines in code for readability
- Variables are appropriately named and used
- Functions are appropriately named and definitions include type annotations
- Functions are used to capture patterns and minimize repeated code
- Functions are used to break down problems and employ abstraction

Part 1: Stock Investor

Your first task is to write a program (in `hw2_stock.py`) that computes the cost of buying stocks. The program asks the user for their investment budget, a stock symbol, the current share price, and the number of shares desired.

The program informs the user of the maximum number of shares they could buy and how much money would remain if they bought the maximum or chosen number of shares.

A few important details: - When asking the user for a monetary amount, the dollar sign (\$) should be included in the prompt so the user only needs to enter the number. - The stock symbol should be included in the prompts for the share price and number of shares to buy. - All monetary values should be displayed with at most two decimal places.

Example output 1

```
$$$$$ STOCK INVESTOR $$$$$
How much would you like to invest? $987.65

What is the stock symbol? AAPL
What is the share price for AAPL? $221.83
You can buy up to 4 shares and still have $100.33 left to invest
How many shares of AAPL would you like to buy? 3

Buying 3 shares costs $665.49
You will have $322.16 left to invest
```

Example output 2

```
$$$$$ STOCK INVESTOR $$$$$
How much would you like to invest? $123.45

What is the stock symbol? HPE
What is the share price for HPE? $16.41
You can buy up to 7 shares and still have $8.58 left to invest
How many shares of HPE would you like to buy? 7

Buying 7 shares costs $114.87
You will have $8.58 left to invest
```

Example output 3

```
$$$$$ STOCK INVESTOR $$$$$
How much would you like to invest? $100.01

What is the stock symbol? TEST
What is the share price for TEST? $20
You can buy up to 5 shares and still have $0.01 left to invest
How many shares of TEST would you like to buy? 1

Buying 1 shares costs $20.0
You will have $80.01 left to invest
```

Part 2: House drawing

Your task is to design and implement (in `hw2_house.py`) a program that creates a drawing of a house as text art.

Task A: Basic house

Write code to draw a basic house:

```
[_____|_|_]
[  ____  ]
[  |__|  |__| ]
[  |__|  |__| ]
[          ]
[  ____  ]
```

```
[  |__|  |__|  ]
[  |__|  |__|  ]
[                    ]
[  _____  ]
[  |__|  |__|  ]
[  |__|  |__|  ]
[  _____  ]
```

The drawing is made using only five distinct characters: spaces (), underscores (_), vertical bars (|), left square brackets ([), and right square brackets (]).

You **must** use separate functions for distinct features and to increase reusability while reducing repetition in your code. Note that there are distinct segments to the figure as well as *repeated elements* which lend themselves naturally to implementing as separate functions.

Task B: Custom house

Modify your code to allow the user to customize the house. In particular, the user must be able to input: * The character to use on the base and roof of the house * A sequence of four characters to use above each window and door

Example output 1

```
Enter a character to use for the base/roof: .
Enter a sequence of characters to use above the windows and door: vVVv
```

```
[.....|..|..]
[  vVVv  vVVv  ]
[  |__|  |__|  ]
[  |__|  |__|  ]
[                    ]
[  vVVv  vVVv  ]
[  |__|  |__|  ]
[  |__|  |__|  ]
[                    ]
[  vVVv  vVVv  ]
[  |__|  |__|  ]
[  |__|  |__|  ]
[.....|..|..]
```

Example output 2

```
Enter a character to use for the base/roof: #
Enter a sequence of characters to use above the windows and door: +==+
```

```
[#####|##|##]
[  +==+  +==+  ]
[  |__|  |__|  ]
[  |__|  |__|  ]
[                    ]
[  +==+  +==+  ]
[  |__|  |__|  ]
[  |__|  |__|  ]
[                    ]
[  +==+  +==+  ]
[  |__|  |__|  ]
[  |__|  |__|  ]
[#####|##|##]
```

Part 3: Bill splitting

When you go out to dinner with friends, one person pays the entire bill and everyone else sends that person money (e.g., using Venmo) for their share of the bill. However, computing how much each person owes requires accounting for the cost of individual dishes, tax, and tip. You and your friends wish you had a program that could perform the computations for you.

We have provided a partial implementation of this program (in `hw2_bill.py`).

Task A: Add docstrings

Add a docstring to each of the existing functions (in `hw_bill.py`), expect for `main`.

Your docstring should be a short sentence that concisely summarizes what the function does.

Task B: Write `main`

Write the `main` function.

Your `main` function **must** use the functions that are already provided, and **must not** call any other functions (e.g., `print`, `input`, `float`). You **must not** change the provided functions or write additional functions. You **may** use variables and operators (e.g., `+`, `-`, `*`, `/`) in `main`.

Example output 1

```
This program assumes gratuity is not included in the bill
What percentage gratuity should be added? 15
```

```
How much is the total bill before tax? 73.56
How much is the total bill after tax? 79.45
Tax rate: 8.0%
```

```
How much is your dish? 14.99
You owe $18.44
```

Example output 2

```
This program assumes gratuity is not included in the bill
What percentage gratuity should be added? 18
```

```
How much is the total bill before tax? 53.46
How much is the total bill after tax? 56.93
Tax rate: 6.5%
```

```
How much is your dish? 10
You owe $12.45
```

Submission Instructions

Submit three Python files to the platform indicated in your class section:

- `hw2_stock.py`
- `hw2_house.py`
- `hw2_bill.py`

Remember to complete the questions at the top of each file before submitting.