

COSC 101 Homework 5: Fall 2024

The due date for this homework is **Thursday, October 17, 11:59pm EDT**

Introduction

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

- Boolean Expressions
- Conditional Statements, including:
 - Nested Conditionals
 - Chained Conditionals

Note: Homeworks build on each other. You should also draw on the skills you used in prior homeworks and labs, including functions and for loops.

! You are not allowed to use constructs/methods/statements, that we have not yet covered in this course—which includes while loops, break and continue statements as well as string methods.

! 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.
- 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.

Your assignment

Your assignment is to complete following steps:

1. Download the hw5.zip file and open it. You will see two python files, hw5_billing.py and hw5_plates.py, in the unzipped folder. Do **not** change the name of these files.
2. Complete hw5_billing.py. This file is used in [Task 1](#).
3. Complete hw5_plates.py. This file is used in [Task 2](#).
4. Review the grading criteria below.
5. 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
- Loops are appropriately used to abstract repeated patterns
- Code is clean (for example, remove code you commented out)
- Code is elegant (for example, limit excessive code, ask yourself, can I do this a simpler way)

Task 1: Electricity Bill

Electricity is an essential resource. In this part of the homework you will write a program (adding to `hw5_billing.py`) to compute and display information for a utility company which supplies electricity to its customers. The program should compute the cost of electric usage and the bill for a customer.

Your program should prompt the user to enter values in the following order:

1. The customer's billing code (one character)
2. The customer's electricity usage in kilowatt-hours (kWh)

The program will compute the bill amount and print some summary information. Below, some detail about the meaning for the above inputs is provided.

! Please use the functions provided in the file as scaffolding. Descriptions of them and sample usage is provided to guide you. You should add relevant parameters, type annotations, and update the docstrings.

Billing code

There are three valid billing codes: `r`, `c`, and `i`. If an invalid code is input, the program should print an error statement.

Computing the bill

The bill amount depends on the billing code used, as follows:

Code `r` (residential):
\$5.00 plus \$0.23 per kWh

Code `c` (commercial):
\$0.12 per kWh for the first 100 kWh then \$0.17 per kWh for the remaining

Code `i` (industrial):

\$0.11 per kWh for the first 200 kWh, then \$0.15 per kWh for the next 100 kWh, finally the remaining is \$0.19 per kWh

Any bills should be rounded to the nearest cent (2 decimal places).

Here is an example of the program's output (x is user input):

```
Billing code for customer: x
Invalid billing code
```

Here is another example of the program's output (r and 300 are user inputs):

```
Billing code for customer: r
kWh's of electricity customer used: 300
Customer Summary:
  Code: r
  Usage: 300
  Bill: $74.0
```

! Precision is important in computer science. Make sure your output **exactly** matches these examples, **including spacing**.

Task 2: Personalized Plate

Your task is to design and implement a program that checks the validity of personalized license plates, in the style of the New York State Department of Motor Vehicles You can explore the [DMV website](#) to realize such a program exists (our rules are slightly different).

For our purpose, a valid plate is made of a combination of numbers and letters,

- is at least 2 characters long and at most 8
- might have no number, but
- must have at least one letter.

Additionally, a personalized license plate has the following restrictions. It **cannot**

- use any symbol or punctuation, such as hyphens, commas, periods or dollar signs. Non-letters, non-numbers and spaces are not allowed.
- use the number zero 0 as the first or last character when all other characters are letters (for example 0RGAN or G0 are invalid).
- use the number zero 0 between two letters (for example B0B is invalid).
- use or contains the substrings FIRE, POLICE, or NYS (ex: TONYSCAR is invalid)
- use the letter I, 0, or S as the first or last character if all other characters are numbers (ex: I23 or 1234S are invalid).
- use the letters I, 0, or S between two numbers to make it appear as a number (ex: 101 or 5I2 are invalid)

- use the following characters twice in a row: the number zero 0, the letter O, the number one 1, or the letter I
- use the number zero 0 and the letter O next to each other, nor the number 1 and I (ex: COOLRIDE, SP00KY, and HI1AMJ0E are all invalid; while N010US is valid).

You should think about how you could write a single function that could cover all the scenarios of the last two bullets for example.

Subtask A:

In `hw5_plates.py` you will find four fruitful function definitions. You should start by adding the docstrings for them and complete with one line the implementation of `isLetterOrNumber`.

Subtask B:

Next you should write code that checks the validity of a personalized plate. Your code **must** include a function `isValidPlate` that takes a string as a parameter and returns `True` if the string is a valid plate, or `False` if the string is an invalid plate. You must first convert the string to its uppercase version using our provided function. Make sure to simplify your code. Check as many conditions as you can with the least code you can.

Subtask C:

Once you have implemented the code to check the validity of a personalized plate, your program should ask first how many plates need to be checked. Your program should request that many plates and process each to create two sets of plates: the valid plates and the invalid ones. Your program finishes by displaying the summary of the simulation as shown below. Your program should match the format of the following three sample outputs.

```
How many plates do you need to enter? 4
Enter your personalized plate request #1: LOD
Enter your personalized plate request #2: B0B
Enter your personalized plate request #3: C0LGATE
Enter your personalized plate request #4: TONYSCAR
You entered the following valid plate(s)
LOD
You entered the following invalid plate(s)
B0B C0LGATE TONYSCAR
```

```
How many plates do you need to enter? 3
Enter your personalized plate request #1: good
Enter your personalized plate request #2: f00dS
Enter your personalized plate request #3: 512A
You entered the following valid plate(s)
512A
You entered the following invalid plate(s)
good f00dS
```

```
How many plates do you need to enter? 6
Enter your personalized plate request #1: HAM1LTON
Enter your personalized plate request #2: UNIGATE
Enter your personalized plate request #3: 13L0D13
Enter your personalized plate request #4: G04IT
Enter your personalized plate request #5: C0SC101
```

Enter your personalized plate request #6: PYTHON
You entered the following valid plate(s)
HAMILTON UNIGATE 13LOD13 G04IT C0SC101 PYTHON
You did not enter any invalid plate

Submission Instructions

Submit the two Python files to the platform indicated in your class section.

- hw5_billing.py
- hw5_plates.py

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