CSI2100-01 Spring 2020 Lab 12

Contents

1 General Information

1.1 Coding Style Guide

Starting with Lab 12, we employ a **coding style guide** to enhance the readability and clarity of our source code. You find the guide in file `StyleGuide.pdf` provided with the Lab 12 specification.

- Please read the style guide and have a look at the provided examples. The style guide comes with a link to our web site where you will find additional examples.
- For every lab problem, Hyeongjae Python will run a style checker to test whether your code conforms to the style guide. You are kindly asked to fix style issues in your code befor you submit on YSCEC.

1.2 Rules for Comments and Docstrings

Starting with Lab 12, the following rules for comments and docstrings are in place:

1. Every function and method (including code provided to you) must contain a docstring that describes its purpose.

2. Every function and method (including code provided to you) must be commented. As previously, you should comment on the important/complicated parts on your code.

   As an exception, you are not required to provide comments with functions and methods that contain less than 3 non-comment LOC.

   Example:

   ```
   1  def shortFunction():
   2      """
   3      This function is just a placeholder.
   4      """
   5      pass
   ```

   The above function consist of only 2 non-comment LOC (line 1 and line 5) and therefore does not require comments.

3. Every class (including code provided to you) must include a docstring that describes the purpose of the class.

4. Every file that you submit must contain a docstring that contains your name, student ID and problem name.

   Example:

   ```
   """
   Name: Hwango Lee
   Student ID: 202067584
   Lab problem: lab12_p1.py
   """

   ...
   ```

1.3 Administration and Notation

- **Because of our coronavirus precautions, you are provided with an excerpt of the relevant sections of the textbook together with this lab assignment.**
- In some places and to facilitate reading, we use “␣” to depict a space (blank) character, and “¤” for a “\n” (newline) character.

2 Programming Problems

Problem 1:

Write a Python function that receives three arguments:

1. A (possibly empty) dictionary which will be used to store the average temperature for each day of the week.

2. A weekday “day”.
3. The average temperature of weekday “day”.

The function should add the temperature to the dictionary only if it does **not** already contain a temperature for that day. The function should return the resulting dictionary (updated or not). Please use the following stub for your function:

```
def addDailyTemp(mydict, day, temperature):
    """
    Add key 'day' and value 'temperature' to the dictionary 'mydict',
    only if key 'day' does not already exist in the dictionary.
    The resulting dictionary is returned.
    """
```

Hint: You are **not allowed** to make any assumption on how a value of the function parameter “day” looks like! (It could be any immutable value, like a string ’Monday’ or ’Mon’ or ’Montag’ (in German), or an integer 1, to name only a few examples. The important point is that for a given key that does not already exist in the dictionary, your function must store the given temperature value. Thus, you must determine whether the given key is already contained in the dictionary. If so, do not add the value and just return the dictionary. Otherwise, add the key-value pair to the dictionary and return the dictionary.

Please submit only your function plus the `import` declarations that your function implementation requires. Do not submit any other code (main program, test code, . . .).

Problem 2:

Exercise P2 from page 380 of the textbook.
You can assume that the keys in the dictionary are stored as strings `'Sun'`, `'Mon'`, `'Tue'`, `'Wed'`, `'Thu'`, `'Fri'`, and `'Sat'`.

Please use the following stub for your function:

```
def moderateDays(mydict):
    """
    Returns a list [...] of the days for which the average
    temperature was between 70 and 79 degrees.
    """
```

For example, if Sunday and Monday were the only days in the asked temperature range, the returned list should be `['Sun', 'Mon']`. If no day matches the temperature range, an empty list should be returned.

The temperature range is *inclusive*, which means that a day with temperature 70 or 79 should be included in the returned list.

There is no particular order required for the strings in your result list. No temperature, just the days must be returned.

Please submit only your function plus the `import` declarations that your function implementation requires. Do not submit any other code (main program, test code, . . .).

Problem 3:

Problem D3 from page 334 of the textbook.
Your program should ask the user for a filename. Depending on the filename extension (.txt or .enc), the program should perform the corresponding action (encryption or decryption). For example:

```
Enter␣a␣filename:␣foo.txt¤
```

The program will generate a random key and store it in file foo.key. Then file foo.txt will be encrypted using the generated key, and stored in file foo.enc.

Decrypting works as follows:

```
Enter␣a␣filename:␣foo.enc¤
```

The program will open file foo.key to load the key, decrypt file foo.enc, and store the result in file foo.txt. If a filename provided by the user does not exist, you can terminate the program with any given error message or exception.

(*H1*) You can assume that the character-set from the input-file is restricted to lower-case letters, upper-case letters, digits (0-9), spaces (' '), and new-line characters ('\n').

(*H2*) With this problem, it is not allowed to use the identity function $f(x) = x$ for encryption and decryption of a character x. The only exception is the new-line character: **do not encrypt the new-line character!** Copy the new-line character verbatim (as is) to the output.

(*H3*) You must make sure that you do not encrypt two input-characters to the same encrypted character. For example, it is not allowed to use both $f(x) = i$ and $f(y) = i$. Because you then have no way to decrypt character "i". Should it be decrypted to "x" or "y" ? In mathematical terms, your key-file corresponds to an *injective* function from characters to characters.

(*H4*) We use the CSV file format to store a random key. For the example in the textbook, the file would have the following contents:

```
A,F¤
B,G¤
C,L¤
...
```

(*H5*) The CSV file format is a text file format. You can open such a file with PyCharm's editor to see what your program generated (select `File` ⇒ `Open` and navigate to your `.key` file). Being able to view a CSV file in our editor is useful for debugging.

(*H6*) Although we use the CSV file format for key files, we do **not** use the `.csv` file-name extension! We use extension `.key` instead, as specified above.

(*H7*) Your key-file must contain one entry for each character that may appear in the input file (as specified with Hint (*H1*) above). Because we do not encrypt new-line characters, your key-file should not contain an entry for new-line.

Problem 4:

Please download the source-code of the `Fraction` class in file `fraction.py`. This class is a basic implementation of mathematical fractions that we discussed in Lecture 10. For example: $\frac{1}{2}, \frac{3}{7}, \ldots$

Please extend this class by adding the following two methods.

Method 1:

```
def reduce(self):
    """
    Reduces self to simplest terms. This is done by dividing both
    numerator and denominator by their greatest common divisor (GCD).
    Also removes the signs if both numerator and denominator are
    negative. Whole numbers (1, 2, ...) are represented as
    1/1, 2/1, 3/1, ...
    """
```

Method 2:

```
def adjust(self, factor):
    """Multiplies numerator and denominator by factor."""
```

You should submit only the code of your `Fraction` class and any import declarations that your class requires. Do not submit any other code with your class (no test code, etc.).

Below you find an example how we might test your class:

```
 1  from lab12_p4 import Fraction
 2
 3  f1 = Fraction(2, 12)
 4  f2 = Fraction(1, 6)
 5  f2.num = 2
 6  print('f1:', f1)    # f1: 1/6
 7  print('f2:', f2)    # f2: 2/6
 8  f2.reduce()
 9  print('f2:', f2)    # f2: 1/3
10  f2.adjust(3)
11  print('f2:', f2)    # f2: 3/9
```

Note from the above listing (Line 6) that newly-created fraction objectes are automatically reduced. To make this work, the `__init__` method of the `Fraction` class has to call the `reduce()` method. You must add this to the `__init__` method.

Note on Hyeongjae Python: Hyeongjae Python will display the test code with its diagnostic output. The test code makes use of the **`format`**`()` method provided by Python string (**`str`**) objects. Please consider the very accessible example here for understanding how the **`format`**`()` method from the test code works.

3 Marking Criteria and Plagiarism

Marking Criteria

- Programming style: code must adhere to our style guide (Section 1.1) and our documentation guidelines (Section 1.2), otherwise points will be deducted.

- Score is only given to programs that compile and produce the correct output with Python version 3.
- Points are deducted for programs that are named wrongly. See the list of deliverables for the required file names.
- Points are deducted for programs that produce warnings.
- Please pay particular attention to the requested output format of your programs. Deviating from the requested output format results in points deductions.
- Use of modules:
  - You are allowed to use the `sys` module with all programing problems.
  - Please always use an exit code of 0 with the `exit()` function of the `sys` module: `sys.exit(0)`.
  - You are allowed/asked to use modules stated with individual programming problems and contained in provided code (if any).
  - No other modules are allowed.

Plagiarism (Cheating)

- This is an individual assignment. All submissions are checked for plagiarism.
- Once detected, measures will be taken for **all** students involved in the plagiarism incident (including the "source" of the plagiarized code).

Deliverables and Due Date

In the below table, column "Style Check" refers to the coding style guidlines from Section 1.1.

Column "Correctness" refers to the correctness check that was already provided with previous labs.

Please note that at this stage no check for the documentation guidelines from Section 1.2 is provided. Please make sure to document your code as requested.

		Hyeongjae Python	
Problem	File name	Style Check	Correctness
1	lab12_p1.py	✓	✓
2	lab12_p2.py	✓	—
3	lab12_p3.py	✓	—
4	lab12_p4.py	✓	✓

- Please submit all files in a ZIP-file named lab12_<student_id>.zip
- Please submit your archive on YSCEC by Wednesday, June 17, 2020, 23:00.