BME69500DL ECE695
Spring 2020

Homework 1

Deadline : Thursday, January 30, 2020 , 11:59 am

1 Introduction

The goal of this homework is to improve your understanding of the Python OO code in general, especially with regard to how it is used in PyTorch.

This is the only homework you will get on general Python OO programming. Future homework assignments will be specific to using PyTorch classes directly or your own extensions of those classes for creating your DL solutions.

2 Tasks

1. Create a class named `People` that has three instance variables named:
   - `first_names`
   - `middle_names`
   - `last_names`

2. Use Python's class `random` to generate an array of 10 random strings of length 5 for each of `first_names`, `middle_names`, and `last_names`. Each string consists of 5 randomly generated lowercase alphabet letters. Make sure to initialize the seed of the random number generator to 0 before generating random strings. To generate a single lowercase alphabet letter, you can use the following Python statement:

   ```
   random.choice(string.ascii_lowercase)
   ```

   make sure to import the Python packages 'random' and 'string' for this statement to execute successfully.

3. Create an instance of your class and store the three arrays in the three instance variables of the `People` instance.

4. Expand your class definition and endow it with an iterator. Iterating through the data stored in the `People` instance should print out the 10 names, with each name in the following order:

   ```
   first_name middle_name last_name
   ```

5. Further expand the definition of the class and endow it with another instance variable that takes one of the following three values:
   - `first_name_first` (that is, a name should appear in the format "first middle last")
   - `last_name_first` (that is, a name should appear in the format "last first middle")
   - `last_name_with_comma_first` (that is, a name should appear in the format "last, first middle")

Note that the last choice is basically the same as the second choice, except that the last name is followed with a comma (a practice that is used in some countries and in some publications when showing author names).

Now show that you can create an instance of your expanded class with a value for the new instance variable. And also show that when you iterate through the instance, it prints out the names in the chosen format. Illustrate that for the 3 different formats in the order given by the above bullets.

6. Further expand the definition of the class and make its instances callable. With this new definition for the People class, when you apply the function-call operator '()' to an instance, it should print out a sorted list of just the last names.

7. Finally, extend your `People` class into a subclass named `PeopleWithMoney`. Endow this class with its own instance variable named `wealth`. Initialize wealth with 10 randomly generated integer between 0 and 1000. Again create an iterator for this class that gets a part of its work done by the iterator of the parent class. Now demonstrate the following:

   - When you iterate through an instance of `PeopleWithMoney`, that should print each individual's name (first, middle, last) and the wealth associated with the individual.
   - Also make the instances of the subclass callable. When you call an instance of `PeopleWithMoney` with the function-call operator '()', that should print out the names of all the individual sorted by the size of their wealth ascendingly.

3 Output Format

The output of your code, that corresponds to the tasks above, will be evaluated using automated scripts. Thus, it is of particular importance that you follow the suggested output format below **precisely**. Note that there isn't a newline character before the first output line nor after the last output line. Each output segment below is separated by a single newline character.

```
first_name1 middle_name1 last_name1
first_name2 middle_name2 last_name2
.
.
.
first_name10 middle_name10 last_name10

last_name1 first_name1 middle_name1
last_name2 first_name2 middle_name2
.
.
.
last_name10 first_name10 middle_name10

last_name1, first_name1 middle_name1
last_name2, first_name2 middle_name2
.
.
.
last_name10, first_name10 middle_name10

sorted_last_name1
```

```
sorted_last_name2
.
.
.
sorted_last_name10

first_name1 middle_name1 last_name1 wealth_amount1
first_name2 middle_name2 last_name2 wealth_amount2
.
.
.
first_name10 middle_name10 last_name10 wealth_amount10

first_name middle_name last_name lowest_wealth_amount
first_name middle_name last_name second_lowest_wealth_amount
.
.
.
first_name middle_name last_name highest_wealth_amount
```

4 Submission

- Make sure to submit your code in Python 3.x and not Python 2.x.
- The first 4 tasks are stepping stones to build your code. The grading script will check the output of task 5 to 7.
- All homework to be submitted on-line through the *turnin* command on the min.ecn.purdue.edu[1] server.
- Log into min.ecn.purdue.edu server using your Purdue career account.
- From min.ecn.purdue.edu, go to the directory where your homework files are. e.g., if your files are at /home/hw1 directory then go to /home/hw1. Obviously, if you don't have your files on the server, move them there using, for example, the *scp* command.
- Type in the following command:

  ```
  turnin -c ecedl -p hw<double digits hw#> <your files separated by a space>
  ```

- As an example, for this homework you will enter the command:

  ```
  turnin -c ecedl -p hw01 file1.py
  ```

- You should get a statement that says "Your files have been submitted to ecedl, hw01 for grading". You can verify your submission by typing:

  ```
  turnin -c ecedl -p hw<double digits homework number> -v .
  ```

[1] If you are an undergraduate student, use the shay.ecn.purdue.edu server instead of the **min** server