Quiz-4

Due Date: 02/05 23:59

Please follow my codes and you will have a good understanding of what is expected of the quiz. Create separate files for each program. **Create a repo called quiz4** and upload all the files there. **Follow the file name convention given with the questions**, so that it will be easier for the TAs to grade. One cannot learn programming without writing it, so please practice. I trust all of you will at least type on their own. Please don't be like this:

"Program, program everywhere, and
Not a program I understand" – Das (The Rime of the Ancient Programmer)

1. Write a program(WAP) in which you will be using a python argparse module to parse the input arguments. There will be three inputs 1) string 2) int 3) float
   Example >>python3 arguments.py hello 007 9.11
   [Name of the file should be **argumnets.py**](1point)
2. WAP to demonstrate ABC. Create a base class and inside of this base class there should be two abstract methods (use @bastractmethod decorator). This base class should be inherited by two other classes which will implement the methods inside of the base class. Come up with your own classes. Please do not create a class called vehicle and inherit it into motorbike and car.
   [Name of the file should be **abc_class.py**](1point)
3. Rewrite the above program using the protocol method
   [Name of the file should be **protocol.py**](1point)
4. WAP a program in which the class is using **@dataclass** decorator. Come up with your own idea of a data class.
   [Name of the file should be **dataclass.py**](1point)
5. Extend the above program (program-4) to include an extra function inside of the @dataclass class.

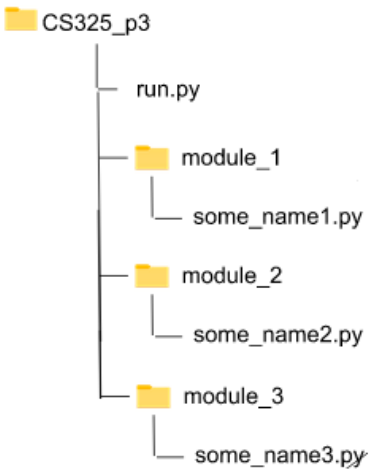
   Example:
   ```
   @dataclass
   class xyz:
       name: str
       price: float
       quantity: int = 0

       def write_your_own_function(self) -> Return_type:
           #your function code here
   ```

   So there will be an extra function. Implement the function to something meaningful.
   [Name of the file should be **dataclass_ext.py**](1point)

6. WAP a program that will demonstrate the @property decorator.
   [Name of the file should be **property.py**](1point)

7. Extend the above program (program-6) by removing the @property decorator and simply use a function. Consult my codes from the class. Come up with a different program.
   [Name of the file should be **no_property.py**] (1point)
8. WAP to demonstrate modules. Follow the following directory structure:



   ONLY FOLLOW THE STRUCTURE NOT THE FOLDER/FILE NAMES. So basically there should be three folders you can name them anything you want. Inside of each folder write a program containing a function. The program run.py (see above) will import all the three functions from the three folders and use them to demonstrate that it indeed used those functions in some meaningful way.
   [Name of the file main file should be **run.py** other files/folders can be named appropriately] (3 points)

If something is missing please assume appropriately.

Send the Github link to the TAs with your name and #800. The subject of the email should be cs325 quiz-4.

Bonus questions:

1. Write a generic function that will accept an array of type string or integer and count the length of each element.

   Example,
   Input: [“abc”, “apple”, “orange”]
   Output: [3,5,6]
   Input: [12, 456, 9000]
   Output: [2,3,4]

No points: But I am going to put your name in my SIUE website under something cool. And if you keep on doing bonus problems for a certain number of times, I will give you an award towards the end of the semester in front of the whole class. It can be for more than one student. Still contemplating the idea but I will do something. Please participate. If you have

attempted the bonus question the subject of the email should be “**cs325 quiz-4_bonus**”. Put everything in the same repo.