GitHub Link: https://github.com/nehaz1515/Assignment_10.1

Neha Zope
CSE-20-03
3 December 2021

Assignment 10.1 README

Class Documentation:

The purpose of this class is for the user to build or create an object, in this class specifically a cup, based on size and type. From there, they can then figure out what drinks are allowed and the cup and how much amount of liquid can fit in the cup. The purpose of the class, which is called Cup, is used to help define what cup is. Since it consists of methods and variables, the main purpose of the class, Cup, is just to define the object cup so that the user can build their own cup. It essentially acts as a prototype from which objects are created, in this case the class Cup is acting as a prototype from which the cup can be created.

There is one class variable and two data variables in this class. The class variable is titled has_opening and it is a class variable since every cup, no matter size or type, has an opening to drink from and it's the same for every cup. Therefore, it does not need to be a data variable but rather a class variable that is shared by all object instances of the class). The first data variable(which is private) is called self.__size. This data variable represents or stores the size of the created cup, which can only be small, medium, or large. The second data variable(which is also private) is called self.__type. This data variable represents or stores the type of the created cup, which can only be plastic, wood, china, steel, or glass.

There are technically six total methods for this class. First one is set_size. This method needs the input of a size(only small, medium, or large) and it essentially sets or changes the self.__size variable to store the new size. If the initial size of a cup was small, and then the user entered set_size("large") then the output should be stored as large now. The next method is called get_size. This method does not need any input, it will just return whatever is stored in the self.__size variable, so it will return the size of the cup. It will however return the latest version of whatever is stored in the variable, meaning if the user did change the size using set_size, get_size will return that newly changed size. These same two methods are used for the type of cup as well. Set_type needs the input of a type(plastic, wood, china, steel, or glass) and it will set or change the self.__type variable to store the new type. If the initial type of a cup was plastic, and the user then entered set_type("china), then the output will be stored as china. The get_type method does not need any input either, it will just return whatever is stored in the self.__type variable, so it will return the type of the cip. Again, it will return the latest version of whatever is stored in the variable, so if the user changed the type using set_type, get_type will return that newly changed type. The next method is called cup_volume, which will essentially tell the user how much oz of liquid they can have based on the size of their cup. This method does not need any input, since it is using the self.__size variable that was already set or stored. What this method does is based on whatever size was initially inputted for self.__size, this method will return a value of how many oz of liquid the cup can hold. Small can hold 8oz, medium can hold 12oz, and large can hold 16oz. The last method is called which_drinkswork and what this method does is tell the user what types of drinks will work or be compliant with the type of cup(since certain drinks don't work with certain materials). This method has no input either since it uses the self.__type variable and whatever was stored in that variable. Essentially this method will return a statement that will say what types of drinks do or do not work with the cup. If the cup is made of wood or plastic, the method will say no hot liquids will work and if the cup is made from china, steel, or glass, the method will say any liquids, hot or cold, will work with the cup.

Demo Program Instruction:

Description: In the demo program, multiple things are happening. The demo program is building three different types of cups. The first cup is called own_cup. All that is happening here is the cup is being created from the class Cup. It is created as the input of a size("large") and the type("glass") is inputted and therefore the class is helping to create this object, which is the cup or own_cup instead. There are not any methods being used on it as it is just an example of how the user can build a cup. The next cup in the demo program is called x. This cup is being built from the class Cup and having the size as medium and the type as plastic. Now the first thing being called in the demo program is the class variable, titled has_opening. Since this is a class variable because every cup has an opening, the output should return True(and we are printing the output to show that it says True). Then the methods get_type, get_size, cup_volume, and which_drinks work are all being called on the created cup, x. The third cup being created is called make_cup with the size small and type china. Again this cup is being created from the class Cup. For this cup, the methods set_size, set_type, get_size, get_type, cup_volume, and which_drinkswork are all being called on the created cup, make_cup.

Directions on how to run the program: A user can either run this program on terminal or on Visual Studio Code itself. If they want to run it on Visual Studio code itself, all they have to do is open the program on VSC, make sure the file/program is saved as MyOwnClass.py, and then press the right-sided triangle to the top right which will run the Python file. If the user wants to open the terminal and run it, the user should first make sure the directory is correct(the directory should match where the file is), and then enter python3 before entering MyOwnClass.py. So essentially it should look like on one line/command python3 MyOwnClass.py. Also if the user wants to build their own cup or play around with the own_cup(that was created in the demo program), they can first enter python3, and then on a new line make sure to import the class Cup from MyOwnClass module, and then continue to create or test whatever cup they want to.