Lab 4: Random Objects

Objectives

- Practice making and using classes and objects.

Preliminary Setup

For this lab, you will be creating an object-oriented (OO) program to play a simple dice game. Start with the following:

1. Create a PyCharm project.

2. Create **main.py**, where you will put the **main()** function.

3. Create a **dice.py** module where you will put the **Dice** class.[1]

Step I: Write the Dice class

In the **dice** module, write the **Dice** class. We are going to be modeling dice with different numbers of sides. Therefore, a Dice object should have two properties/attributes:

- the number of sides the dice has and
- the current value showing on the dice.

These properties should be *attributes* in your class (also known as *instance variables*).

Add the following methods:

- A constructor that (in addition to `self`) takes ***only*** a parameter that indicates the number of sides that the particular dice should have. Be sure that the constructor gives each of the instance variables an appropriate starting value. Make sure that each of the instance variables is **private**.
    - **NOTE**: In general, a constructor should be sure to assign values to each of the attributes/instance variables, even if a given instance variable does not correspond to a parameter passed to the constructor. In this case, the current

1 I've checked, and "Dice" is an acceptable singular form. If this bothers you because you consider "Dice" as **plural** only, then you should instead create a "Die" class. Classes should be named with singular nouns.

value of the dice is not passed as a parameter, but should still be assigned by the constructor. What's a good value to assign this attribute?

- A `roll()` method that rolls the dice (causes the current value to be a new random number from 1 to the number of sides).
    - Note: you can use **randint** from the **random** module; see the docs.
    - Note: **roll** is a *setter method* -- it changes the attributes of the given Dice object. It ***doesn't need to return*** anything -- it just changes the attribute(s).
- A `get_value()` method that returns the value currently showing on the dice.
    - Such a method is called a *getter method*.

Step II: Write main

In the **main** module, write the **main()** function to play a simple game according to the following rules:

1. The game uses a D6 and a D12 (6-sided and 12-sided dice).
2. On each turn, we roll both dice.
3. The game is won if one of the dice shows a value that's exactly twice the value of the other.
    a. Note: we win if the D12 shows twice the D6 OR if the D6 shows twice the D12.

Your **main** function should create variables for the two dice: each should be an object of type **Dice**. On each turn/round of the game, the two dice objects should be rolled -- don't create new dice objects on each round.

Your program should use the **input** function to ask the user to hit return before each turn. Recall that the input function is used to get input from the user, but here we don't really need to use the input the user types -- we are just using it to pause the game between rounds to control the pace of the game. So, you can do something like:

```
input("Press return to continue")
```

without putting the result in a variable.

When the user presses return, the program should roll the dice and print out their new values. After each turn, the program should check if we've won. If we've won, print a message saying so and end the program.

Step III: Add a default [Extra Credit]

Now, go back and modify the **Dice** class so that the default number of sides is 6, so that if we ask to create a Dice object without specifying the number of sides, we will get a D6. **We still want to be able to create other sizes** of Dice, we just want the default size to be 6.

Modify **main()** so it uses this default when we want a D6, and specifies a non-default size when we want a D12.

How to turn in this lab

Before turning in any program in this class, remember this mantra:

> ***Just because it works doesn't mean it's good.***

Part of your grade will also come from things like how understandable and readable your code is. You will also be graded on the neatness, presentation, and style of your program code.

Turn in the lab via gradescope by 4pm the day after the lab.

Ask for help if you're having problems!