**Object-Oriented Programming in Python**

**Youtube videos:**

Python OOP 1 - Classes and Instances - https://youtu.be/ZDa-Z5JzLYM

Python OOP 2 - Class Variables - https://youtu.be/BJ-VvGyQxho

Python OOP 3 - Classmethods and Staticmethods - https://youtu.be/rq8cL2XMM5M

Python OOP 4 - Inheritance - https://youtu.be/RSl87lqOXDE

Python OOP 5 - Special (Magic/Dunder) Methods - https://youtu.be/3ohzBxoFHAY

Python OOP 6 - Property Decorators - https://youtu.be/jCzT9XFZ5bw

**Reading:**

Guttag. Introduction to Computation and Programming using Python. *Chapter 8, Classes and Object Oriented Programming*, p.90 - 110.

**Assignment:**

1. **Rolling multi-sided dice**

**Description of the task**

A normal die (the singular of dice) is a cube, and each face shows a number from one to six. Some games employ nonstandard dice that may have fewer (e.g. four) or more (e.g. thirteen) sides. Let’s design a general class MSDie to model multi-sided dice. We could use such an object in any number of simulation or game programs.

Each MSDie object will know two things:

1. How many sides it has
2. Its current value

When a new MSDie is created, specify how many sides it will have (e.g *n*). The die can be operated on through three provided methods: *roll,* to set the die to a random value between 1 an *n,* exclusively; *setValue*, to set the die to a specific value (i.e cheat); and *getValue* to see what current value is.

**Steps**

1. Use the skeleton script msdie.py.
2. Create class MSDie with attributes *number of sides* and *value.*
3. Create class methods for rolling the die, getting the value of the die and setting the value of the die.

**2. Dice roller widget**

Build an application that displays and rolls a pair of six-sided dice. Display the buttons graphically and provide two buttons, one for rolling and one for quitting the program.

Break the application down into the following programs:

**Building the Button**

The buttons you will be using are not the modern ones that have a 3D look and feel. So the best that can be done, is to find out where the mouse clicked after the click has already been done. The buttons will be rectangular regions in a graphic window where user clicks can influence the behaviour of the running application. Create a class, Button in the script *button.py*, containing the following:

1. *Constructor -* This initialises all the instance variables. Create a button in a window. Specify the window in which the button will be displayed, the location/size of the button, and the label that will be on the button.
2. *Activate -* Set the state of the button to active.
3. *Deactivate* - Set the state of the button to inactive.
4. *Clicked -* Indicate when the button is clicked. If the button is active, this method will determine if the point clicked is inside the button region. The point will have to be sent as a parameter to the method.
5. *getLabel* - Return the label string of the button. This is provided so that we can identify a particular button.

**Building Dice**

The purpose of this class is to display the value of a die in a graphical fashion. The face of the die will be a square (via Rectangular) and the pips (i.e. the dots on the die) will be circles. Create a class, DieView, in the script *dieview.py* containing the following:

1. *Constructor -* Create a die in a window. Specify the window, the center point of the die, and the size of the die as parameters.
2. *setValue -* Change the view to show a given value. The value to display will be passed as a parameter.

HINT

* The main thing in DieView is turning various pips “on” and “off” to indicate current value of the die.

**The Main Program**

This script, *roller.py,* imports the Button and DieView classes from their respective modules. It creates the application window, draws the interface widgets, defines an event and can close the window. The program should get mouse clicks and process them until the user clicks Quit.