KEY CONCEPTS:

We discussed the idea of objects as a way to write reusable code.

We use CLASSES as way to define a blueprint - a collection of properties as attributes (think variables) and methods what can act on that object (think functions)

OBJECTS are instances of this CLASS - each can (and usually do) have different properties or attributes, but all have the same methods available to them

You can refer to documentation here:
https://www.w3schools.com/python/python_classes.asp

Class Code:
https://colab.research.google.com/drive/10oYk8I5gxBnCAD3GrlmeX5AgCfd3Vehz?authuser=0#scrollTo=PN0czNvIip3H

Homework:

This assignment was originally created by Dr. Oleg Smirnov for School Nova

Task 1
Create a class Monster with four attributes (all integer values): a unique numerical ID, original power, current power, and a level. The ID and original power should be the arguments passed by the user. The original power is fixed (it never changes). The initial level is equal to 1. The current power is calculated on the basis of the following formula: current power = initial power * level.

Task 2
Add a basic introduction method for the monster, which should look something like this: “Monster 101, initial power 4, current power 4, level 1.”

Task 3
Generate 10 monsters. Using the instance method from Task 2, display information about each monster.

Task 3 Challenge Version:
Generate 10 monsters using a for loop and random values for the original power (between 1 and 5, included). Hint: use ‘i’ from the for loop to generated the unique ID values for your monsters. The instances should be saved as elements of a list. Using the instance method from Task 2, display information about each monster.

Task 4

In the actual game, the monsters interact with a hero. Create a simple class Hero. For our current purposes, the class Hero should have just three attributes: an ID (integer), a level (integer), and a difficulty level (string: “easy”, “normal”, or “hard”). Create a hero who is level 20, playing on “hard” difficulty.

Task 5
Let us now update the class Monster. Implement an instance method that adjusts the level of the monster on the basis of the level of the hero and difficulty level. Therefore, there are two arguments: (1) an integer - the level of the hero, and (2) a string – the difficulty, which is either “easy”, “normal”, or “hard”. The level of the monster is equal to the level of the hero. It is further adjusted given the difficulty. For “easy” it decreases by 2, for “hard” it increases by 2 (and no change for “normal” difficulty”). IMPORTANT: The method should also update the current power of the monster given the monster’s original power and the level of the monster.

Task 6
Use instance method from Task 5 to adjust the level of each monster. Using the instance method from Task 2, display information about each monster to verify that each monster is now level 22. Also verify that the current power of each monster is now equal to the original power * 22.

Task 6 Challenge version: (this step is only possible if you did Task 3 using the challenge version.)
Use for loop and the instance method from Task 5 to adjust the level of each monster. Using the instance method from Task 2, display information about each monster to verify that each monster is now level 22. Also verify that the current power of each monster is now equal to the original power * 22.