

Assignment 2 - COMP2152

Part 1 - Coding Questions (50%)

You have been given function.py and main.py, but you will also need to create files monster.py, hero.py, and character.py. At the end, you will need to submit all 5 files.

1. Make a Hero class

- Add **method** hero_attacks() inside the class.
- Add combat_strength and health_points as **properties** of the class.
- Create an init method (**constructor**) that rolls the dice for combat strength and health points.
- Create a del method (**destructor**) that just prints the following: "The Hero object is being destroyed by the garbage collector".
- Instantiate** a hero object and make it work with for the rest of the fight sequence and overall game

2. Make a Monster class

- Add **method** monster_attacks() inside the class.
- Add m_combat_strength and m_health_points as **properties** of the class.
- Create an init method (**constructor**) that rolls the dice for combat strength and health points.
- Create a del method (**destructor**) that just prints the following: "The Monster object is being destroyed by the garbage collector".
- Instantiate** a monster object and make it work with for the rest of the fight sequence and overall game

- Convert to using the complex getters and setters for ALL of your classes. Add complex getters (use **@property**) and setters (**@propertyName.setter**) for all private properties of all classes. When using @propertyName.setter, remember to replace propertyName with the name of the property in your class you want to set.

- Move your class definitions into specific python files **hero.py** and **monster.py**

- Make a class called Character to serve as a **parent** class.

- In the constructor, add the combat_strength and health_points as properties so that class **Hero** and class **Monster** can both use these properties as the same name. (Note, technically, these properties will *not* be inherited, because they are private. But these properties will *pretend* to be public due to the complex getters and setters, so the Hero and Monster classes *pretend* to inherit these properties)
- Make these properties private



6. Refactor the Hero class and the Monster class so that they only **inherit** the properties `combat_strength` and `health_points` from the parent class. Make sure to call the parent destructor and parent constructor in your child destructors and child constructors.
7. Change from using variables to using object oriented programming for the hero and the monster
 - a. `m_combat_strength` → change to `monster.combat_strength`
 - b. `m_health_points` → change to `monster.health_points`
 - c. `health_points` → change to `hero.health_points`
 - d. `combat_strength` → change to `hero.combat_strength`
8. Add a **try-except block** around the dream levels function when accepting user input. Validate that the number is an integer between 0-3.
9. Use the **os library/module** from python to print out the operating system name of the computer it's running on.
10. Use the **platform library/module** from python to find a function that helps you print out the version of python. Put this at the top of `main.py`, before Testing Functions.
11. When saving the game, **write** another line that represents the number of monsters killed (including all past games)
12. When loading the game, **read** the line that represents the number of monsters killed (including all past games). (ie. Read what you wrote in the previous question.)

Note: These classes should work with `main.py` to have the regular game sequence



Part 2 - Explain your Code (50%)

Write the answers in the doc provided and save and submit Part 2 as a PDF. You do not have to write or submit any new code for this section. I want you to understand how you could work on a piece of code that already exists (as is the case when working with Open Source code), and how to improve it. You can **type** in your answers *or* complete it **by hand** (handwriting **MUST** be legible) and then scan your submission.

1. How have we used classes for our project to reuse code?

2. Provide 1 line of code, as one of many examples, where code is shared between the monster class and the hero class?

3. What is the benefit of using complex getters and setters?

4. If we didn't use try-except blocks, what would be the problem?

5. How could we use the name of the **operating system** or the **version of python** in your game to prevent errors? Choose just 1 of the above.

6. What's another piece of information we could save inside of the save.txt file?
(Remember, we load this information every time we start a new game, so that we can keep track of all of the games you have played so far.)



7. New Feature:

- a. Think of **1 new feature** you can add to the game that could use list comprehension and nested if-statements. For now just write 1 sentence that describes the feature:

Now add your new feature description here:

Examples:

Below are the examples to show you that you can be very creative, and you should have fun with this exercise. You must use an idea that is NOT directly on the list below:

eg a) Add another monster so that the hero can fight 2 monsters at once

eg b) Create a digital board game, that shows the hero moving around to different towns on a map

eg c) Add a dog that runs in front of the hero and discovers features about the world

-
- b. Give the new feature you created a short 2-3 -word a title:

Now write your Title here:

Examples:

eg a) Multiple Monsters

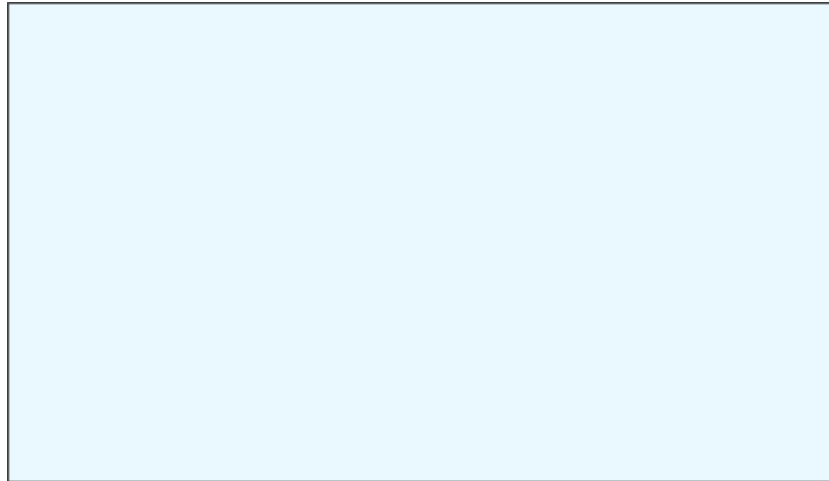
eg b) Roam Towns

eg c) Dog Scout

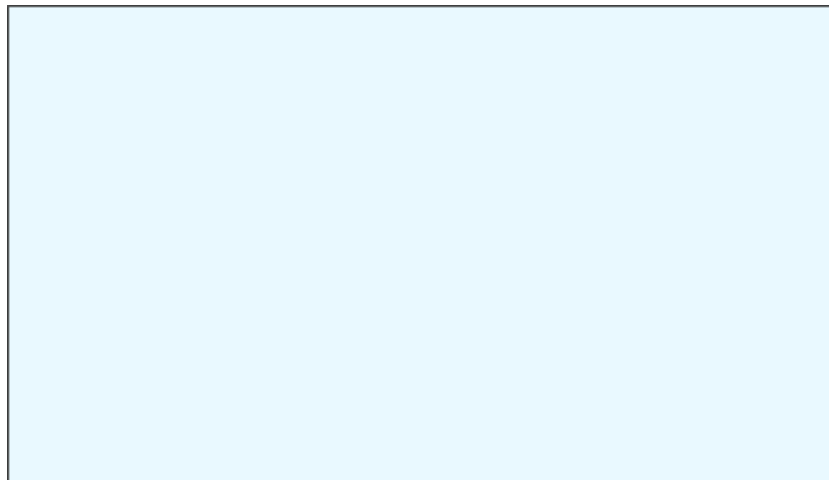


- c. Explain how you could implement the idea you chose. You must explain how you would use both of the control structures below. Draw a diagram, map, sketch for each (you can use any software for this, e.g. Draw.io). You don't have to match the style of diagram I have here, just use a visual to describe your idea. Note, you must have loops and conditional statements diagrammed below as needed:

i. **Using a list comprehension loop**



ii. **Using nested conditional statements**



Example:

eg b) Roam Towns

i. Using a list comprehension loop

Every time in the loop, move one square in 1 direction, (N, E, S, W). Have a variable that keeps track of the Hero's location by saving values of the board. We can have 2 nested for loops and store the map as a 2D array.

Eg.

Hero location is currently at Row 3, Column D.

Town 2 location is at Row 4, Column G.

Town 1 location is at Row 1, Column A .

Diagram:

	A	B	C	D	E	F	G	H	I
1	(town1_loc)								
2									
3				(player_loc)					
4							(town2_loc)		

ii. Using a nested Conditional Statement

If the hero is in Town 2, **then** allow the hero to buy armor but not sell.
Otherwise, the hero can sell armor but cannot buy.

Create an array of armor options available in Town 2. He could also trade some of his loot based on the value of the loot he has.

