1. Provide example code and necessary elaborations for demonstrating advantages of Dynamic Typing as specified in Task 2.

The advantage of Dynamic Typing can be concluded to 2 main points: Flexibility and Writability.

For flexibility, Dynamic Language do not need to declare variables of their data type. So it can enhance the language flexibility since the type of variable need not be declare until it is to be used by other function, so their the programmer can be more concentrate on the entire parts of the program. For example, from GameCharacter.py:

```
# return whether comer entering the cell successfully or not
    def interact_with(self, comer):
        if comer.name == "Player":
            print(
                "\033[1;31;46mA goblin at cell (%d, %d) meets Player. The goblin died. Player's HP - 1.\033[0;0m"
                % (self._row, self._col)
            )
            # update properties of the player and the Goblin
            #       return whether the Player successfully enter the cell
            comer.set_hp(comer.get_hp-self._damage)
            self.set_active(false)
            return true
        return false
        # END
```

In the function interact_with(), we no need to define the data type of the parameter “comer” first when we declare it is the parameter of the function, and we can just design the function and also access comer’s attributes directly (eg comer.hp), it can also let the language can be easily perform duck typing (no matter what class of instant is passed to the function, if it have such attribute HP, this function works for it), which beneficial to the flexibility of the language.

On the other hand, the type of the variable can be changed after some operation performed.

The second advantage of dynamic typing is Writability. Dynamic Typing can enhance writability by providing ability to define and use of complex structures that allow details to be ignore. Since it is no need to declare the data type of the variable that you are using until it is used, it is easier to define the whole data structure of the class in big components.

2. Provide example code and necessary elaborations for demonstrating advantages of Duck Typing as specified in Task 4.

Duck Typing is a way of programming in which an object passed into a function or method supports all method signatures and attributes expected of that object at run time.
One of the advantages of duck typing is flexibility. In duck typing, we don't declare the argument types in function prototypes or methods, so that if the particular object have particular methods, it can also perform the function and it let the operation of the program be flexible: One function can serve more than one data type/class of object.
For example: In Engine.py,

```
def run(self):
        # main rountine of the game
        self.print_info()
        while not self.state():
            for obj in self._actors:
                if obj.active:
                    obj.act(self._map)
            self.print_info()
            self.clean_up()
        self.print_result()
```

run(self) perform act on the object of actors. Since player, goblins have the function act(self), but they are in different content of act(self), so this can flexibly perform the act(self) function of their own, but no need to declare two separate function and combine these two in run(self) function’s for loop, the obj can easily manage the object that taking care in that iteration. It can let the code be simpler and the programmer can concentrate on the entire code but no need to worry how to manage 2 similar meaning functions in same for loop function. It reduce the complexity of function define and enhance the writability as a result.