Report 09

1. **Reflections**

- **What was the easiest and hardest part of this assignment?**

  Easiest part:

  The creation of character class and weapon class, just follow the general routine to create a class, write constructor, set default values and give different methods for the class.

  Hardest part:

  To be very clear with the OOD concepts and their usage such as __init__, __str__, self.

  Dealing with the complicated relationships in mulitiple objects.

  The random magic , heal and critical effect, need to go though the logical order before write the code.

- **What did you learn?**

  Some OOD concepts and their usage such as class, object, __init__, __str__, self.

  How to create a class, create an object, and set parameter for the objects.

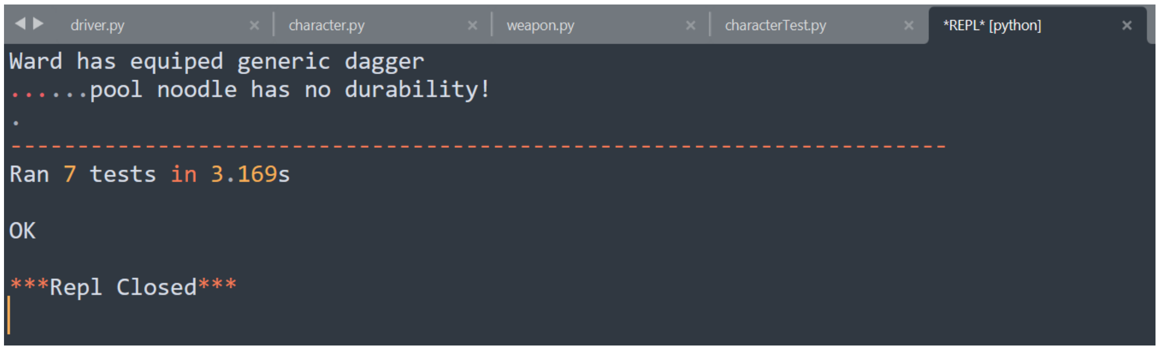
  Practiced the skills to error handling in the user input part.

  Use time and String library to create a type writer printing effect.

- **What grade would you give yourself?**

All 7 test cases has passed, and all basic requirements in rubrics are met. I also added 5 extension ideas such as error handling, fire magic, heal, critical attack, and type writing effect.

For these reasons, I will give myself a 30 to this lab.

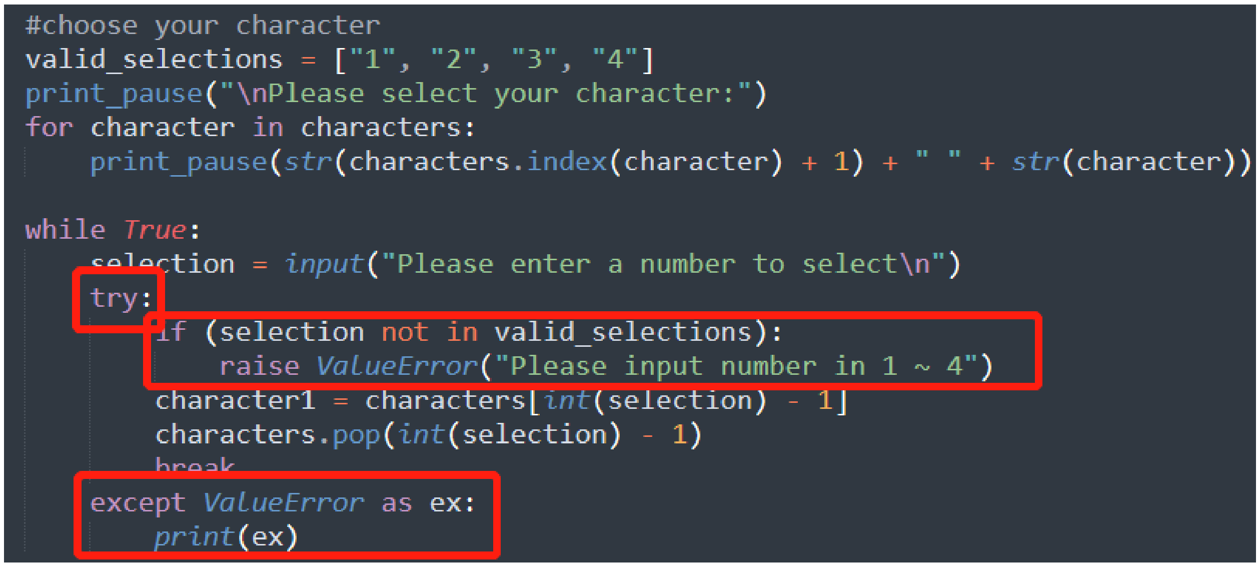


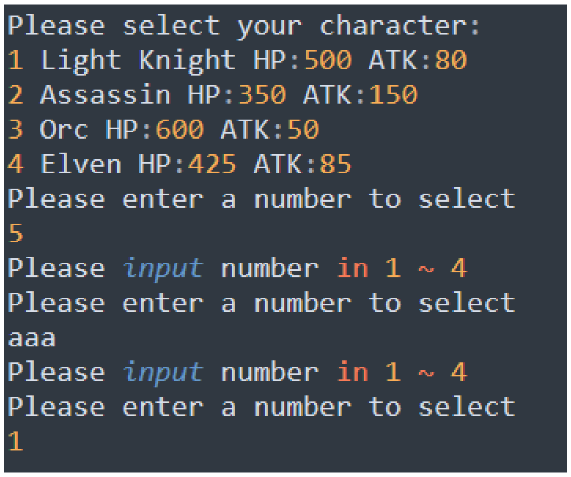
```
Ward has equiped generic dagger
......pool noodle has no durability!
.
----------------------------------------------------------------------
Ran 7 tests in 3.169s

OK

***Repl Closed***
```

2. **Extensions**

- **Add logical exception handling in user input.**

  An exception is added to limit the user input in the character and weapon selection to avoid crushing.

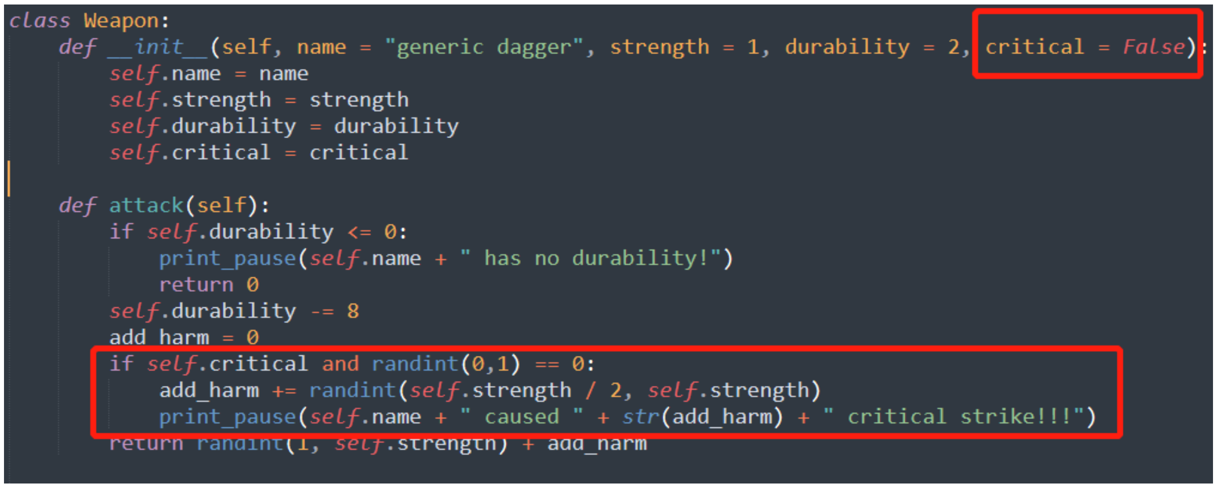

```
#choose your character
valid_selections = ["1", "2", "3", "4"]
print_pause("\nPlease select your character:")
for character in characters:
    print_pause(str(characters.index(character) + 1) + " " + str(character))

while True:
    selection = input("Please enter a number to select\n")
    try:
        if (selection not in valid_selections):
            raise ValueError("Please input number in 1 ~ 4")
        character1 = characters[int(selection) - 1]
        characters.pop(int(selection) - 1)
        break
    except ValueError as ex:
        print(ex)
```

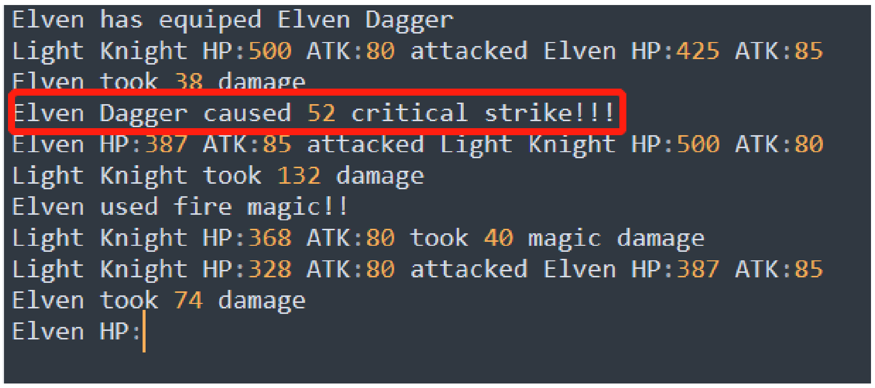

```
Please select your character:
1 Light Knight HP:500 ATK:80
2 Assassin HP:350 ATK:150
3 Orc HP:600 ATK:50
4 Elven HP:425 ATK:85
Please enter a number to select
5
Please input number in 1 ~ 4
Please enter a number to select
aaa
Please input number in 1 ~ 4
Please enter a number to select
1
```

- **Add random effects with a weapon**

In Weapon class, I added a boolean variable “critial”, it is default to be False, but this variable is True for the weapon “Elven Dagger”.

If critical is True, when the character use this weapon attack opponent, there is a 50% chance to cause an extra damage, and user can get the information during battle.


```
class Weapon:
    def __init__(self, name = "generic dagger", strength = 1, durability = 2, critical = False):
        self.name = name
        self.strength = strength
        self.durability = durability
        self.critical = critical

    def attack(self):
        if self.durability <= 0:
            print_pause(self.name + " has no durability!")
            return 0
        self.durability -= 8
        add_harm = 0
        if self.critical and randint(0,1) == 0:
            add_harm += randint(self.strength / 2, self.strength)
            print_pause(self.name + " caused " + str(add_harm) + " critical strike!!!")
        return randint(1, self.strength) + add_harm
```

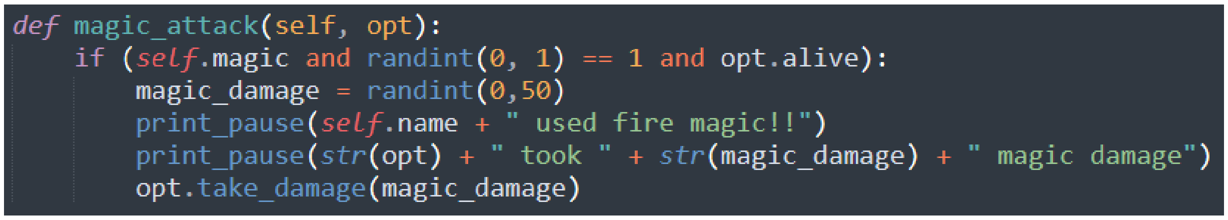


```
Elven has equiped Elven Dagger
Light Knight HP:500 ATK:80 attacked Elven HP:425 ATK:85
Elven took 38 damage
Elven Dagger caused 52 critical strike!!!
Elven HP:387 ATK:85 attacked Light Knight HP:500 ATK:80
Light Knight took 132 damage
Elven used fire magic!!
Light Knight HP:368 ATK:80 took 40 magic damage
Light Knight HP:328 ATK:80 attacked Elven HP:387 ATK:85
Elven took 74 damage
Elven HP:
```

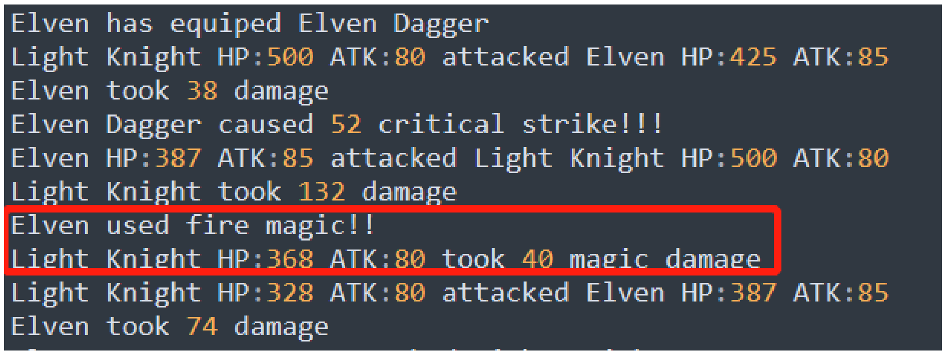
- **Add fire magic to a character**

  In Character class, I added a boolean variable “magic”, it is default to be False, but this variable is True for the Character “Elvenr”.

  If magic I is True, when the character attack opponent, there is a 50% chance to cause an extra fire magic attack, and user can get the information during battle.



```
def magic_attack(self, opt):
    if (self.magic and randint(0, 1) == 1 and opt.alive):
        magic_damage = randint(0,50)
        print_pause(self.name + " used fire magic!!")
        print_pause(str(opt) + " took " + str(magic_damage) + " magic damage")
        opt.take_damage(magic_damage)
```

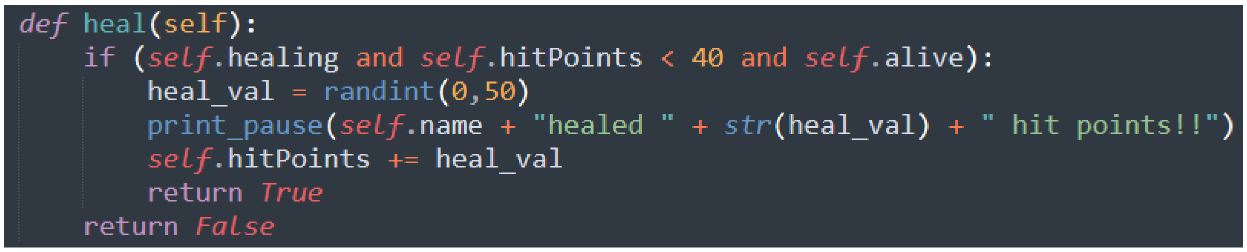


```
Elven has equiped Elven Dagger
Light Knight HP:500 ATK:80 attacked Elven HP:425 ATK:85
Elven took 38 damage
Elven Dagger caused 52 critical strike!!!
Elven HP:387 ATK:85 attacked Light Knight HP:500 ATK:80
Light Knight took 132 damage
Elven used fire magic!!
Light Knight HP:368 ATK:80 took 40 magic damage
Light Knight HP:328 ATK:80 attacked Elven HP:387 ATK:85
Elven took 74 damage
```

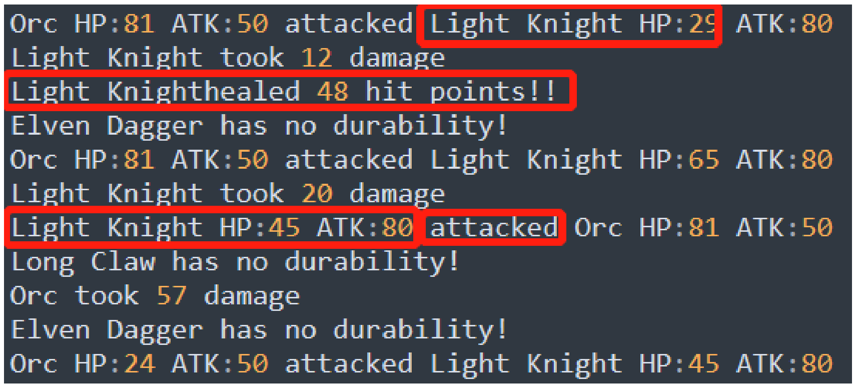
- **Add the ability for a character choose to heal instead of attack**

  In Character class, I added a boolean variable “healing”, it is default to be False, but this variable is True for the Character “Light Knight”.

  If healing is True, when the character hit point is lower than 40, the character will heal instead of attack, and user can get the information during battle.



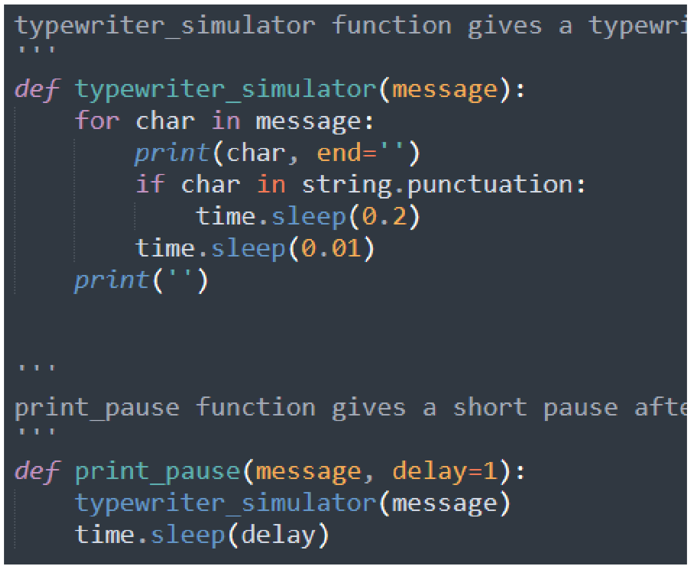
```
def heal(self):
    if (self.healing and self.hitPoints < 40 and self.alive):
        heal_val = randint(0,50)
        print_pause(self.name + "healed " + str(heal_val) + " hit points!!")
        self.hitPoints += heal_val
        return True
    return False
```



```
Orc HP:81 ATK:50 attacked Light Knight HP:29 ATK:80
Light Knight took 12 damage
Light Knighthealed 48 hit points!!
Elven Dagger has no durability!
Orc HP:81 ATK:50 attacked Light Knight HP:65 ATK:80
Light Knight took 20 damage
Light Knight HP:45 ATK:80 attacked Orc HP:81 ATK:50
Long Claw has no durability!
Orc took 57 damage
Elven Dagger has no durability!
Orc HP:24 ATK:50 attacked Light Knight HP:45 ATK:80
```

- **Add type writer effect in message**

  A type writer effect is added in text printing to improve game experience, the text is printed character by character.



```
typewriter_simulator function gives a typewri
'''
def typewriter_simulator(message):
    for char in message:
        print(char, end='')
        if char in string.punctuation:
            time.sleep(0.2)
        time.sleep(0.01)
    print('')


'''
print_pause function gives a short pause afte
'''
def print_pause(message, delay=1):
    typewriter_simulator(message)
    time.sleep(delay)
```