Lab 11: RPG

Joshua Kruse and Joshua Pollock

Lab Section 1

November 22, 2016

**Problem Statement:**

The problem given to us is to create a program that simulates a Role-Playing Game. Some code is given to us, but there are places where it is filled in. The code will be using classes and inheritance to solve the problem. The lab will be using random to randomized the outcome of the code. Some constraints are:

* Damage is calculated by taking defense (magic and physical) in to account and if the damage inflicted is less than the defense, the inflicted damage is 0.
* Damage inflicted will never be negative. If it is negative, then that just means defense is greater than the damage.
* The Fighter and Wizard class will take inheritance from the Adventurer class.
* Fighter HP is 40. Fighter Defense is 10. Fighter Magic Defense is 4.
* Wizard HP is 20. Wizard defense is 4. Wizard Magic Defense is 10.
* (1d8) means 1 dice is rolled 8 times.
* Before each round, initiative will be rolled.

Some assumptions that will be made are:

* All functions inside of the classes will be made private.
* Health will never be negative.
* The user will not input any incorrect data types into the program

**Planning:**

To begin we wanted to decide on what type of code to use. We came to the conclusion to use while loops, for loops, if loops and basic math functions in order for our code to function. We were also asked to create UML diagrams for our code. These UML diagrams will guide us on setting up our classes.

|  |
| --- |
| Dice Roller |
| Roll(self,times,sides) |
| None |

|  |
| --- |
| Attack |
| \_\_init\_\_  Get\_attack\_type  Get\_damage  Get\_name  Get\_sides  Get\_num\_die |
| Self.\_name  Self.\_sides  Self.\_number  Self.\_type |

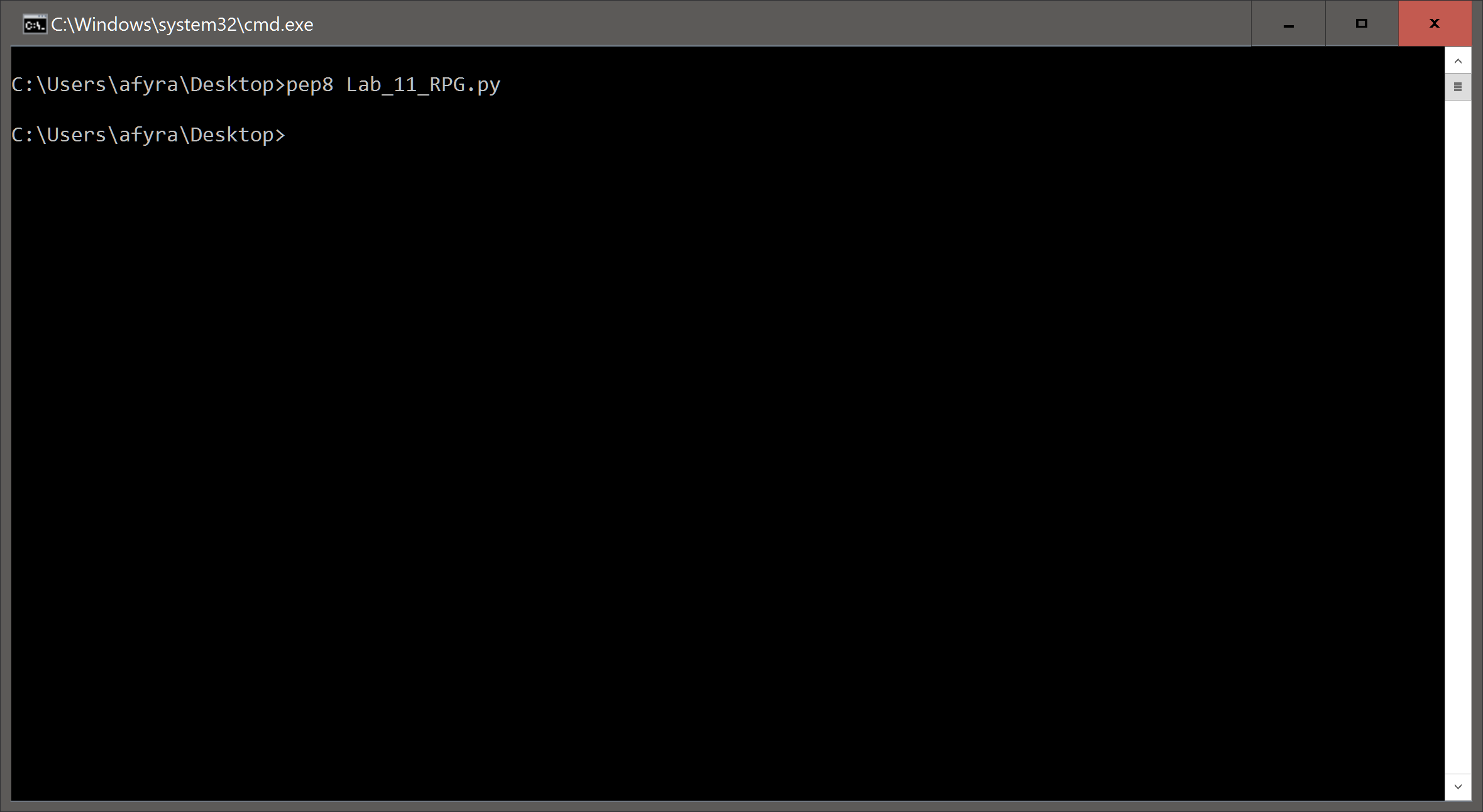
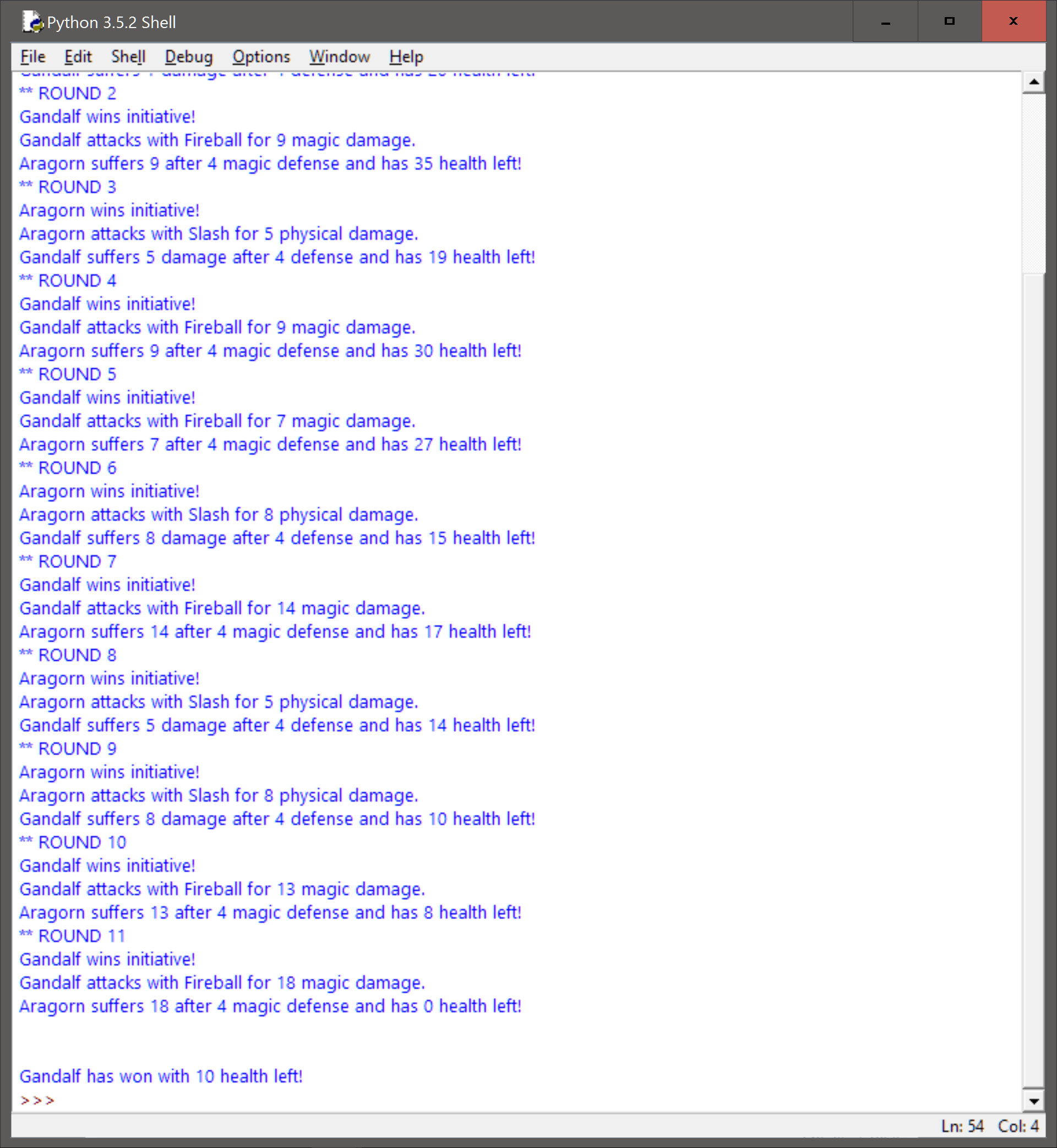
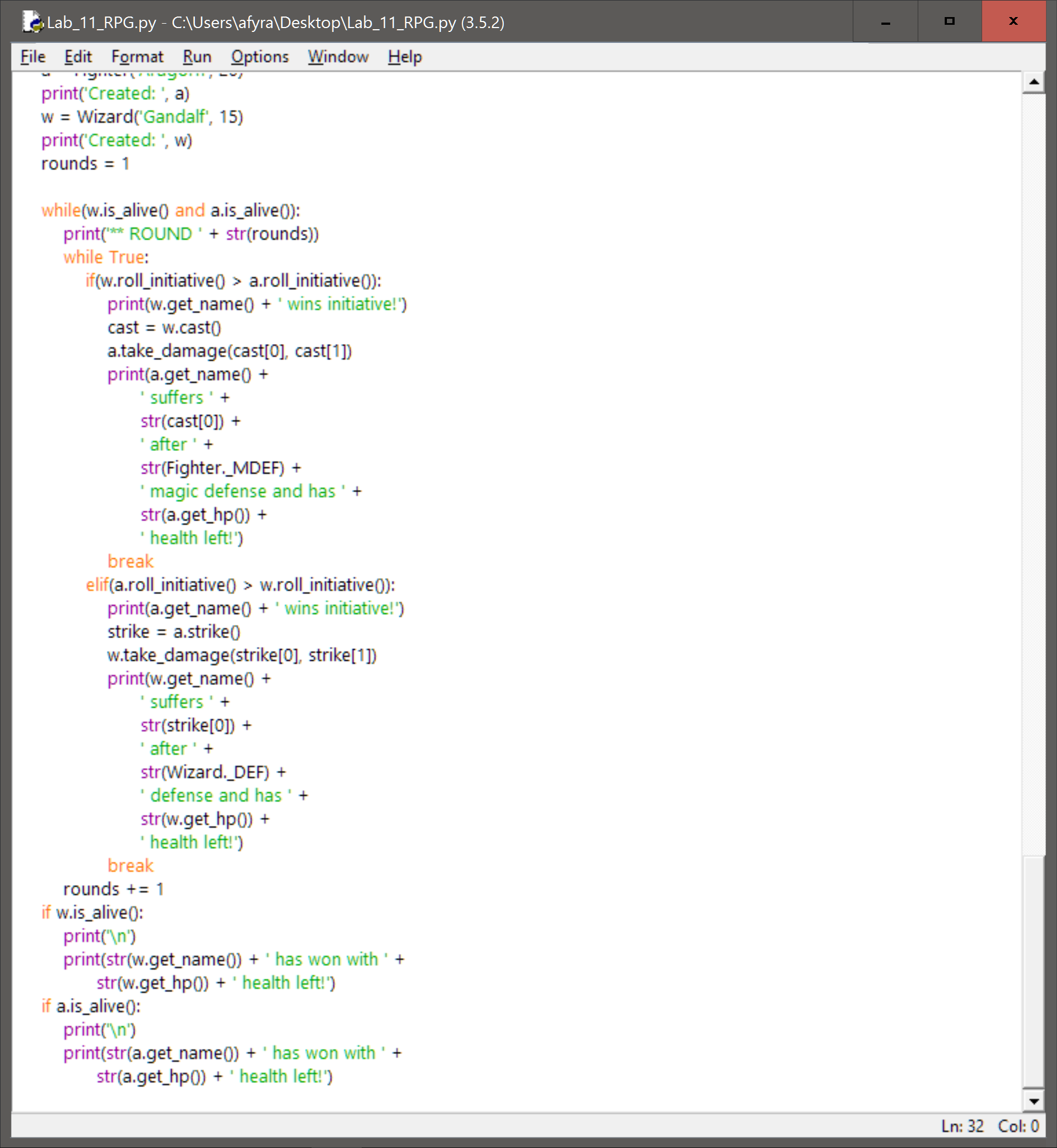
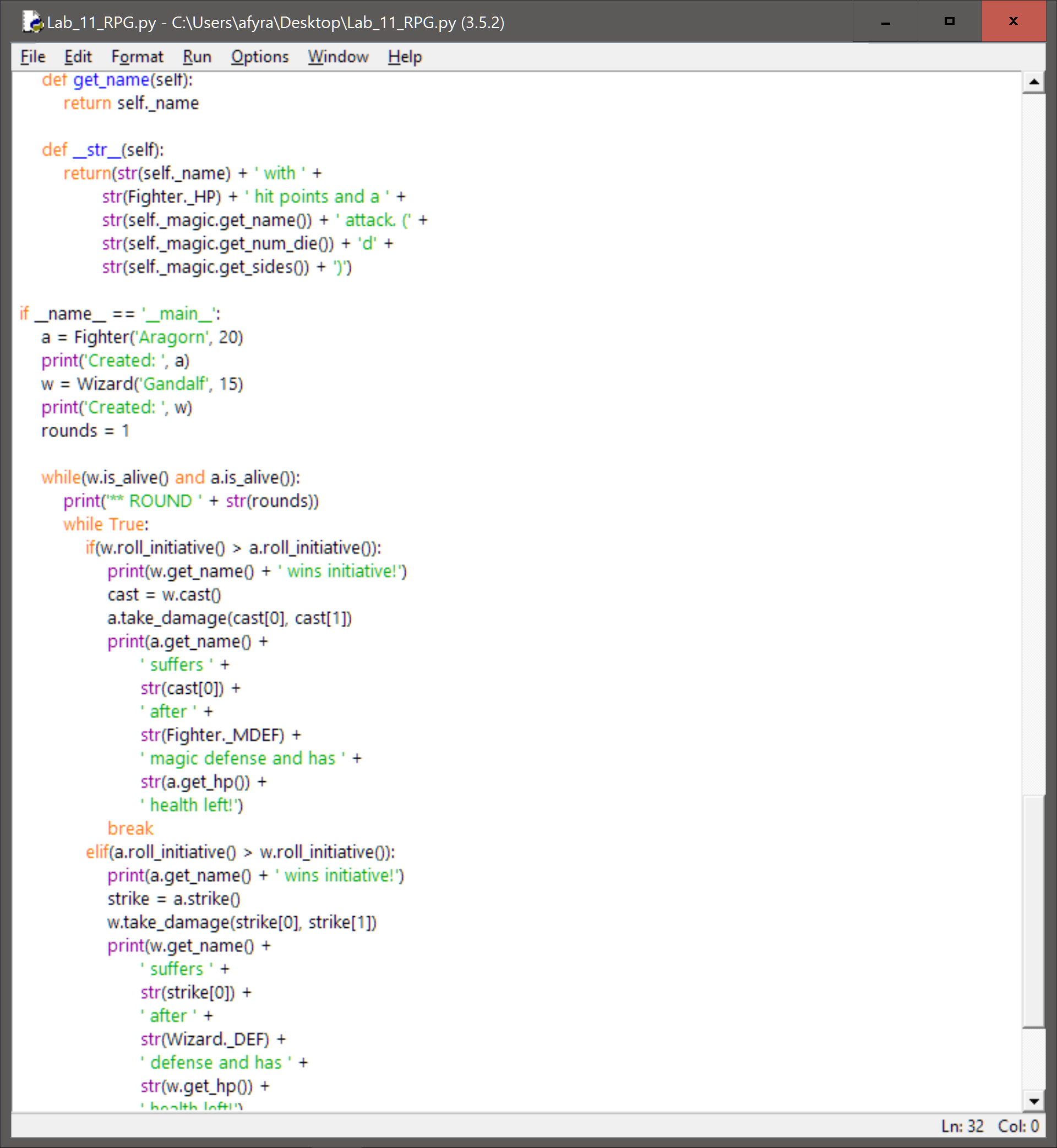
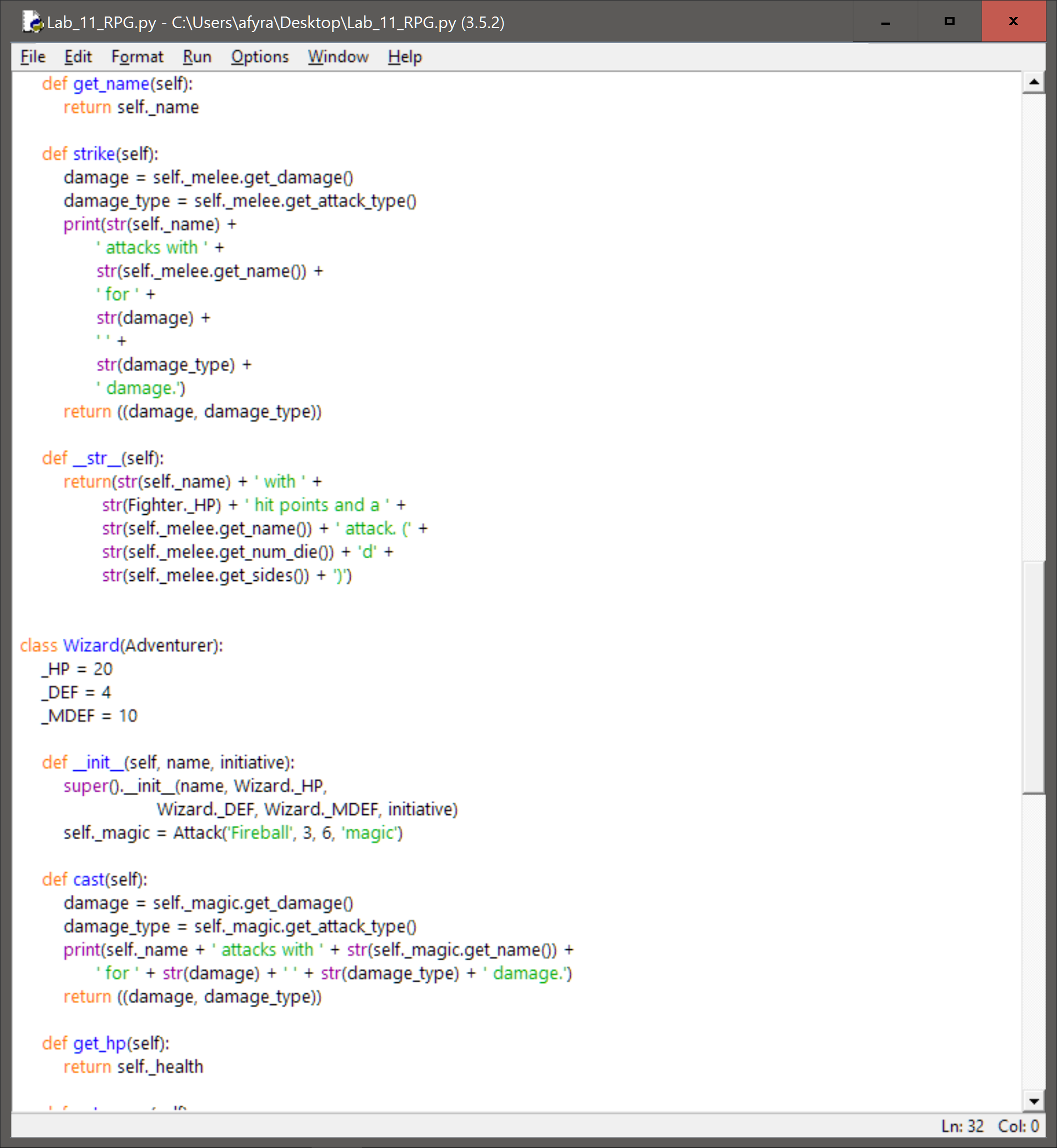
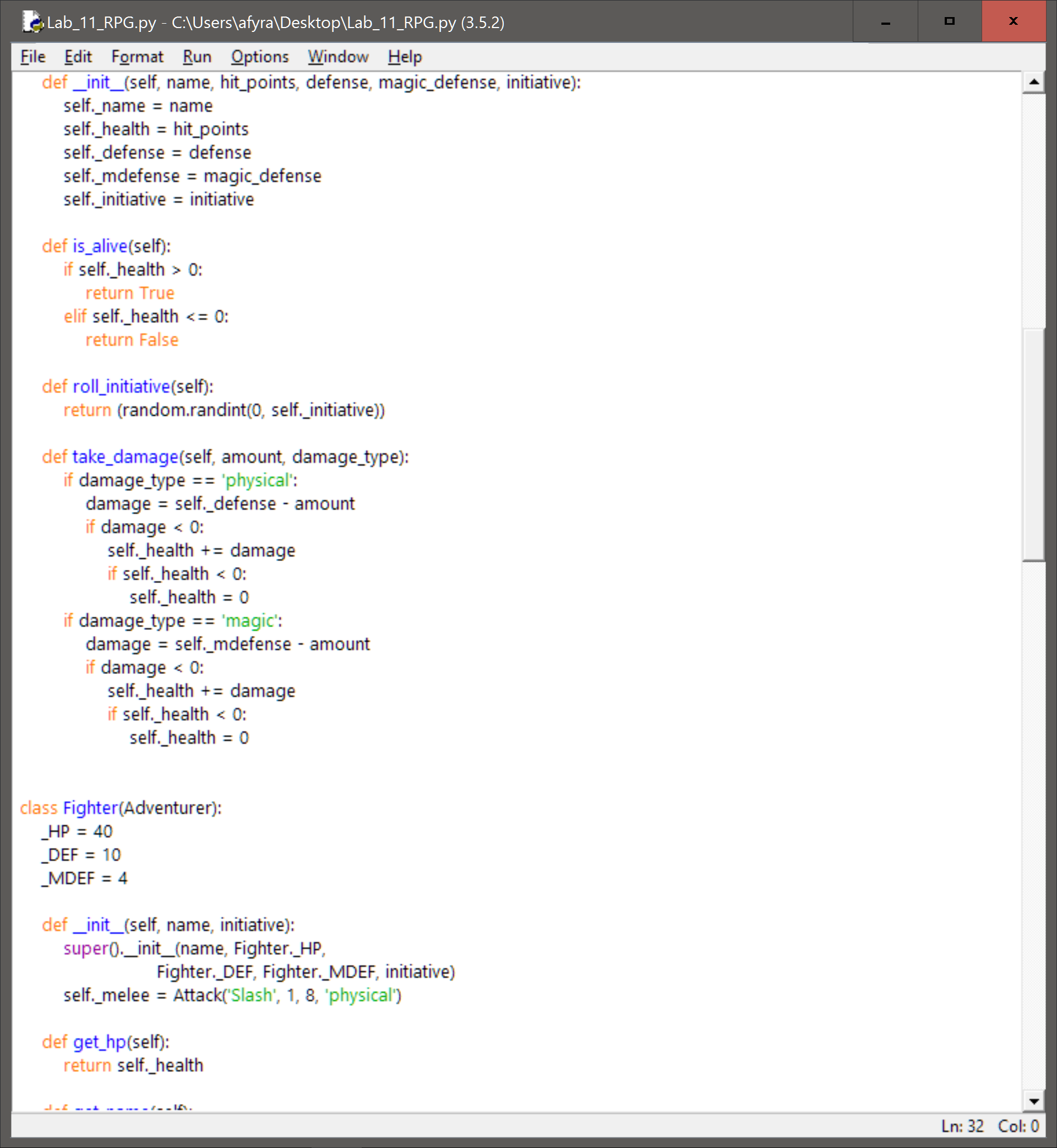
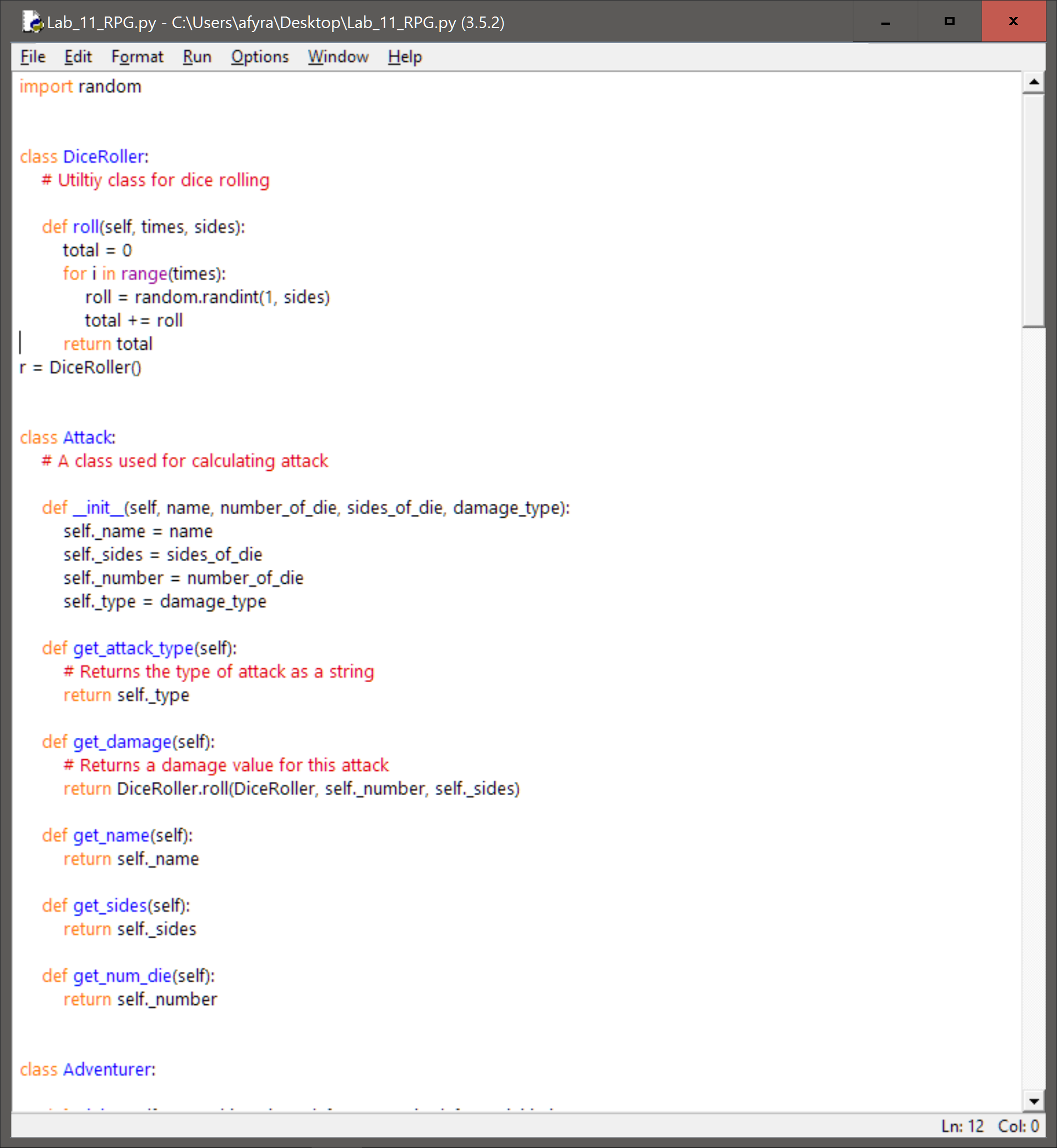
|  |
| --- |
| Adventurer |
| \_\_init\_\_  Is\_alive  Roll\_initiative  Take\_damage |
| Self.\_name  Self.\_health  Self.\_defense  Self.\_mdefense  Self.\_initiative |

|  |
| --- |
| Fighter(Adventurer) |
| \_HP  \_DEF  \_MDEF  \_\_init\_\_  Get\_hp  Get\_name  \_\_str\_\_ |
| Super().\_\_init\_\_(name,Fighter.\_HP,Fighter.\_DEF,Fighter.\_MDEF,initiative)  Self.\_melee |

|  |
| --- |
| Wizard(Adventurer) |
| \_HP  \_DEF  \_MDEF  \_\_init\_\_  Get\_hp  Get\_name  \_\_str\_\_ |
| Super().\_\_init\_\_(name,Wizard.\_HP,Wizard.\_DEF,Wizard.\_MDEF,initiative)  Self.\_magic |

After this we started to type out the example code given to us for use. This given code has gaps within it that we were required to fill.

**Implementation and Testing:**

**Reflection/Refactoring:**

Overall, we are quite happy with our code. The code does what is intended and provides a randomized result for each execution of the code. The current solution keeps within the constraints of the lab. Some refactoring that could be done is reduction in the amount of class member functions. Currently there are many of them, which was first intended for troubleshooting purposes. Since they worked quite well, we decided to leave some of the unnecessary ones in the class. If we wanted to simplify our code further these would be the first things to go. Currently all the print statements are outside of the classes. If we wanted to, we may be able to create a \_\_str\_\_ function within the Adventurer class. With these tiny improvements, the code would run even better, however, we are already quite happy with the solution we have.