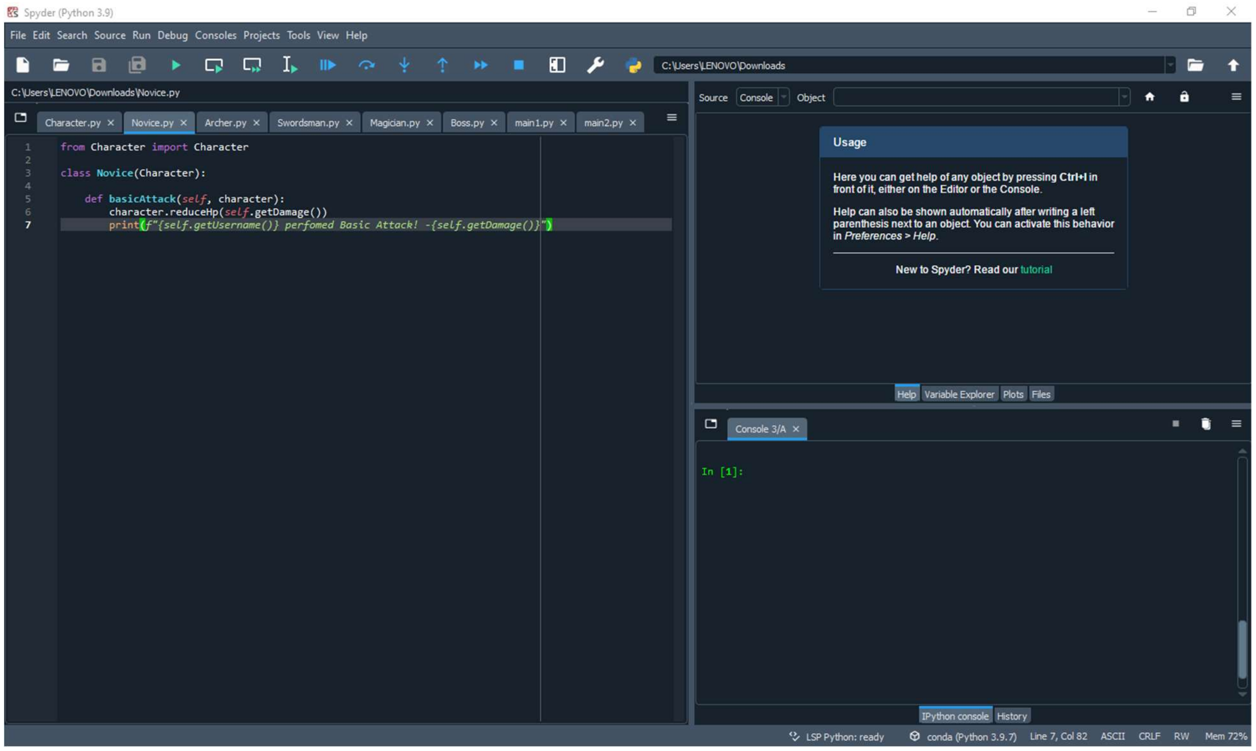
Hands-on Activity 5.1 Inheritance, Encapsulation, and Abstraction	
Buenafe, Dhafny Efa, Christian Francisco, Lauper Xavier V.	4/13/2022
Course/Section- BSCPE12S1	Engr. Roman M. Richard



```python
from Character import Character

class Novice(Character):

    def basicAttack(self, character):
        character.reduceHp(self.getDamage())
        print(f"{self.getUsername()} perfomed Basic Attack! -{self.getDamage()}")
```

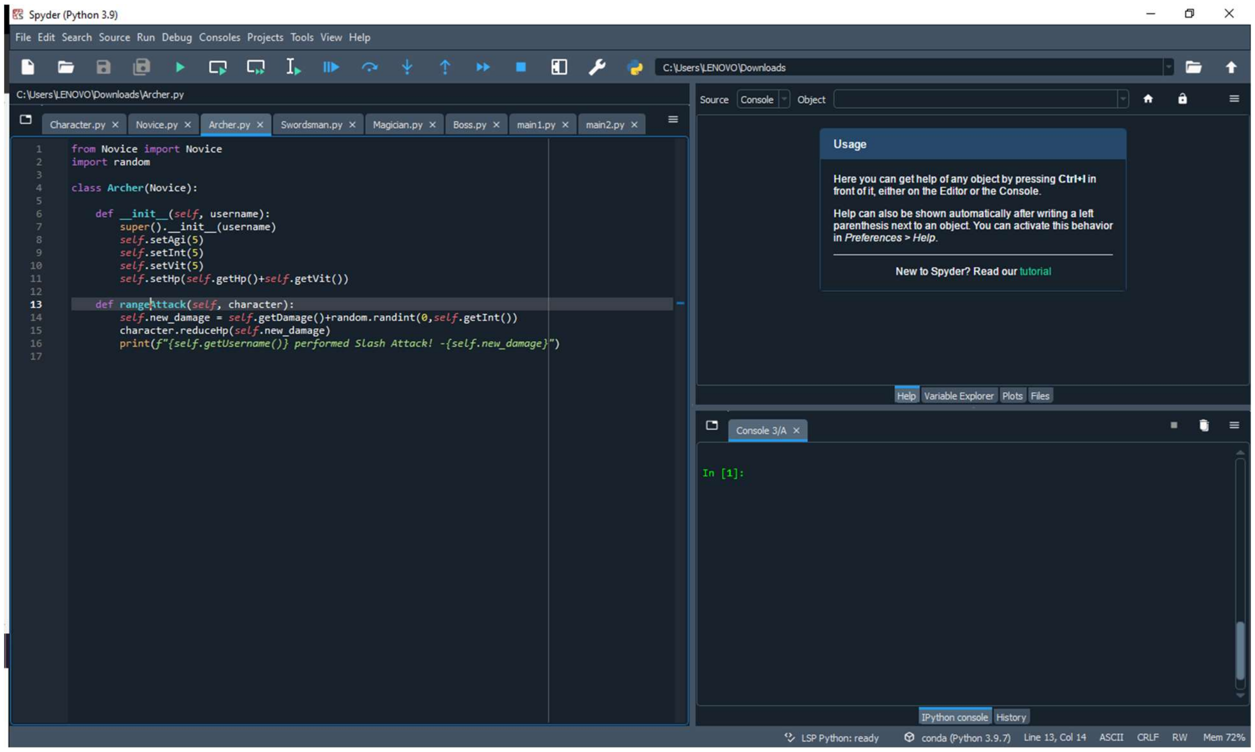
Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
In [1]:
```



C:\Users\LENOVO\Downloads\Archer.py

```python
from Novice import Novice
import random

class Archer(Novice):

    def __init__(self, username):
        super().__init__(username)
        self.setAgi(5)
        self.setInt(5)
        self.setVit(5)
        self.setHp(self.getHp()+self.getVit())

    def rangeAttack(self, character):
        self.new_damage = self.getDamage()+random.randint(0,self.getInt())
        character.reduceHp(self.new_damage)
        print(f"{self.getUsername()} performed Slash Attack! -{self.new_damage}")

```

Usage

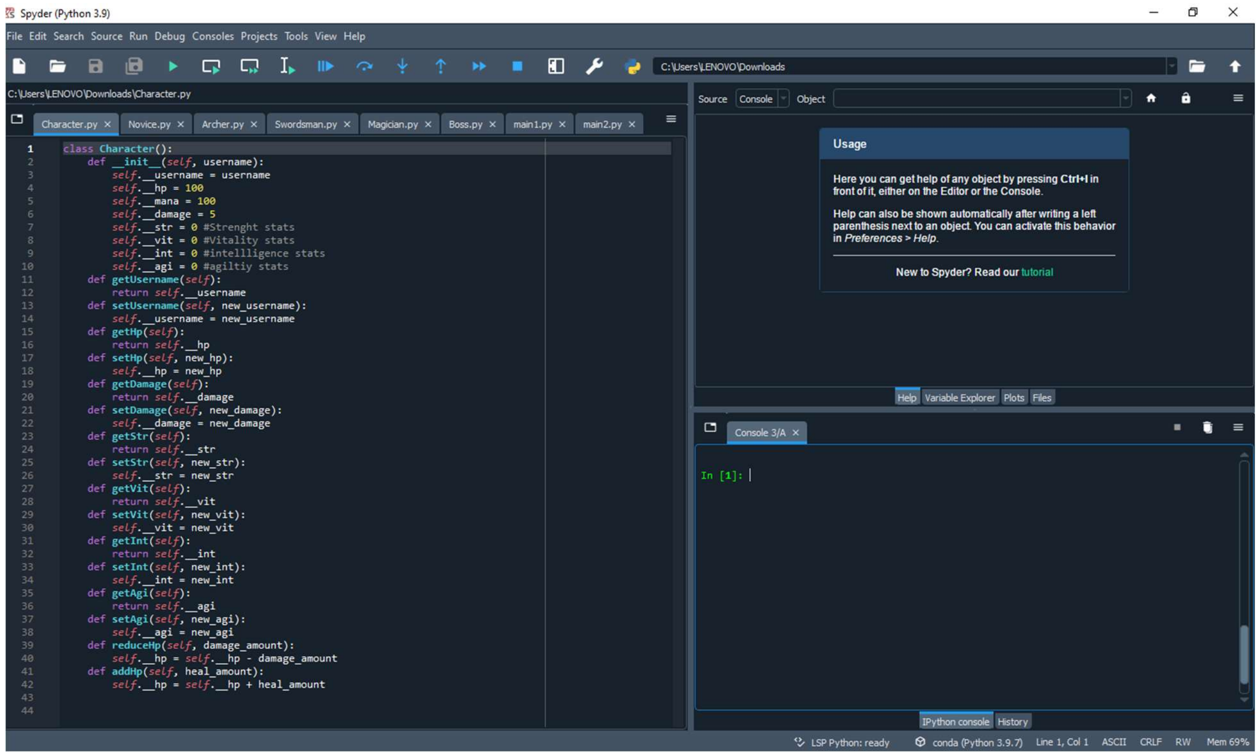
Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
In [1]:
```

LSP Python: ready   conda (Python 3.9.7)   Line 13, Col 14   ASCII   CRLF   RW   Mem 72%



C:\Users\LENOVO\Downloads\Character.py

```python
class Character():
    def __init__(self, username):
        self.__username = username
        self.__hp = 100
        self.__mana = 100
        self.__damage = 5
        self.__str = 0 #Strenght stats
        self.__vit = 0 #Vitality stats
        self.__int = 0 #intellligence stats
        self.__agi = 0 #agiltiy stats
    def getUsername(self):
        return self.__username
    def setUsername(self, new_username):
        self.__username = new_username
    def getHp(self):
        return self.__hp
    def setHp(self, new_hp):
        self.__hp = new_hp
    def getDamage(self):
        return self.__damage
    def setDamage(self, new_damage):
        self.__damage = new_damage
    def getStr(self):
        return self.__str
    def setStr(self, new_str):
        self.__str = new_str
    def getVit(self):
        return self.__vit
    def setVit(self, new_vit):
        self.__vit = new_vit
    def getInt(self):
        return self.__int
    def setInt(self, new_int):
        self.__int = new_int
    def getAgi(self):
        return self.__agi
    def setAgi(self, new_agi):
        self.__agi = new_agi
    def reduceHp(self, damage_amount):
        self.__hp = self.__hp - damage_amount
    def addHp(self, heal_amount):
        self.__hp = self.__hp + heal_amount

```

Usage

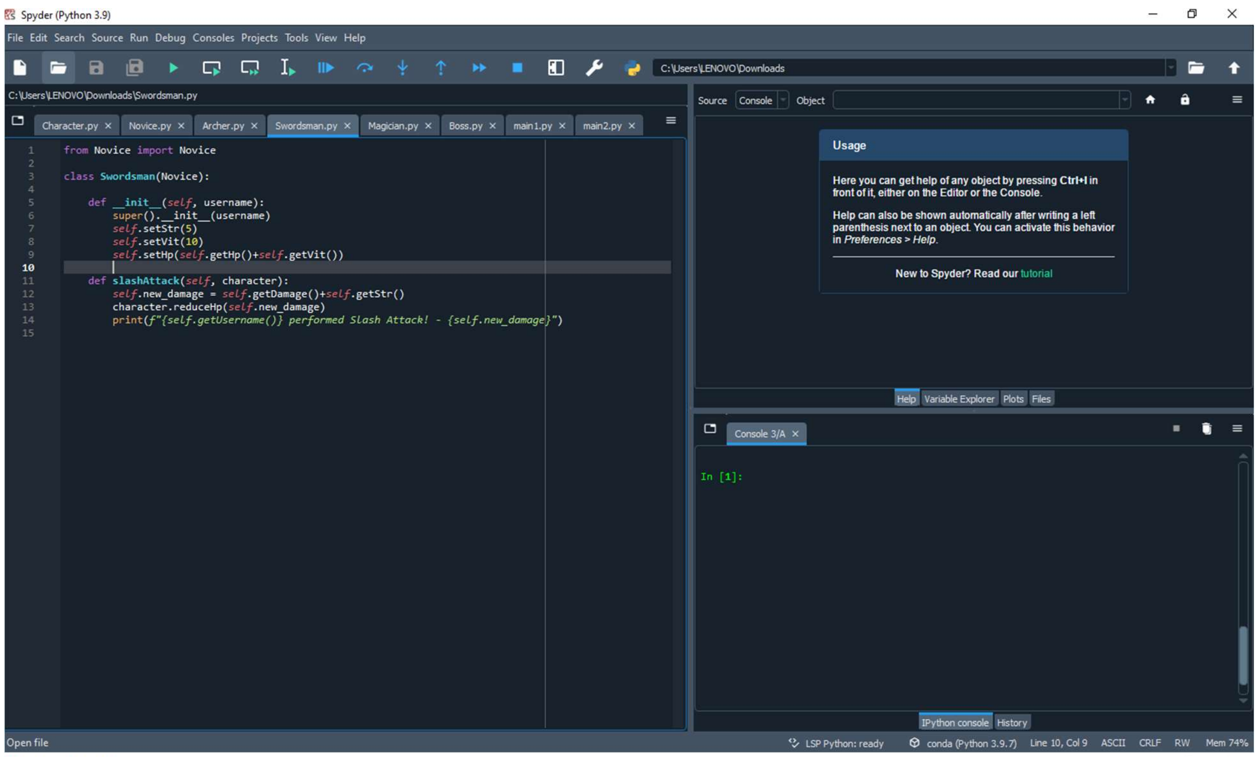
Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
In [1]:
```

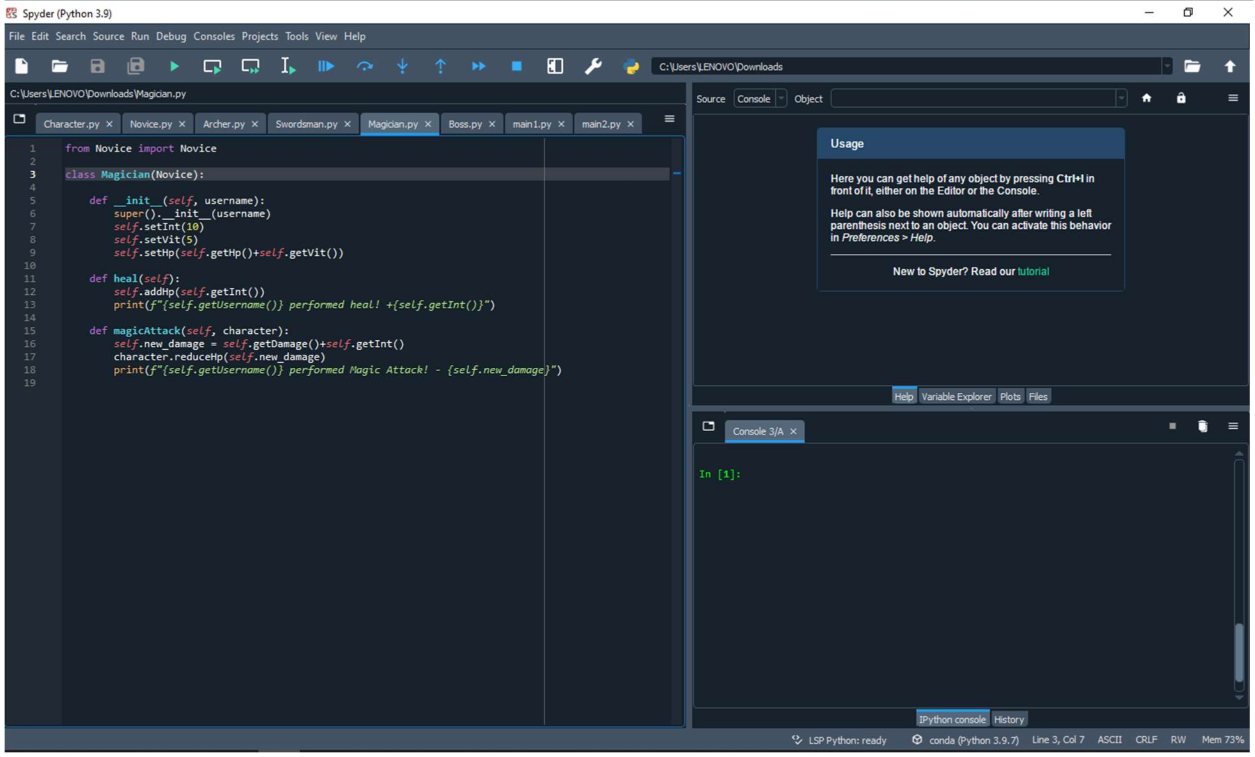
LSP Python: ready   conda (Python 3.9.7)   Line 1, Col 1   ASCII   CRLF   RW   Mem 69%



C:\Users\LENOVO\Downloads\Swordsman.py

```python
from Novice import Novice

class Swordsman(Novice):

    def __init__(self, username):
        super().__init__(username)
        self.setStr(5)
        self.setVit(10)
        self.setHp(self.getHp()+self.getVit())
        
    def slashAttack(self, character):
        self.new_damage = self.getDamage()+self.getStr()
        character.reduceHp(self.new_damage)
        print(f"{self.getUsername()} performed Slash Attack! - {self.new_damage}")

```

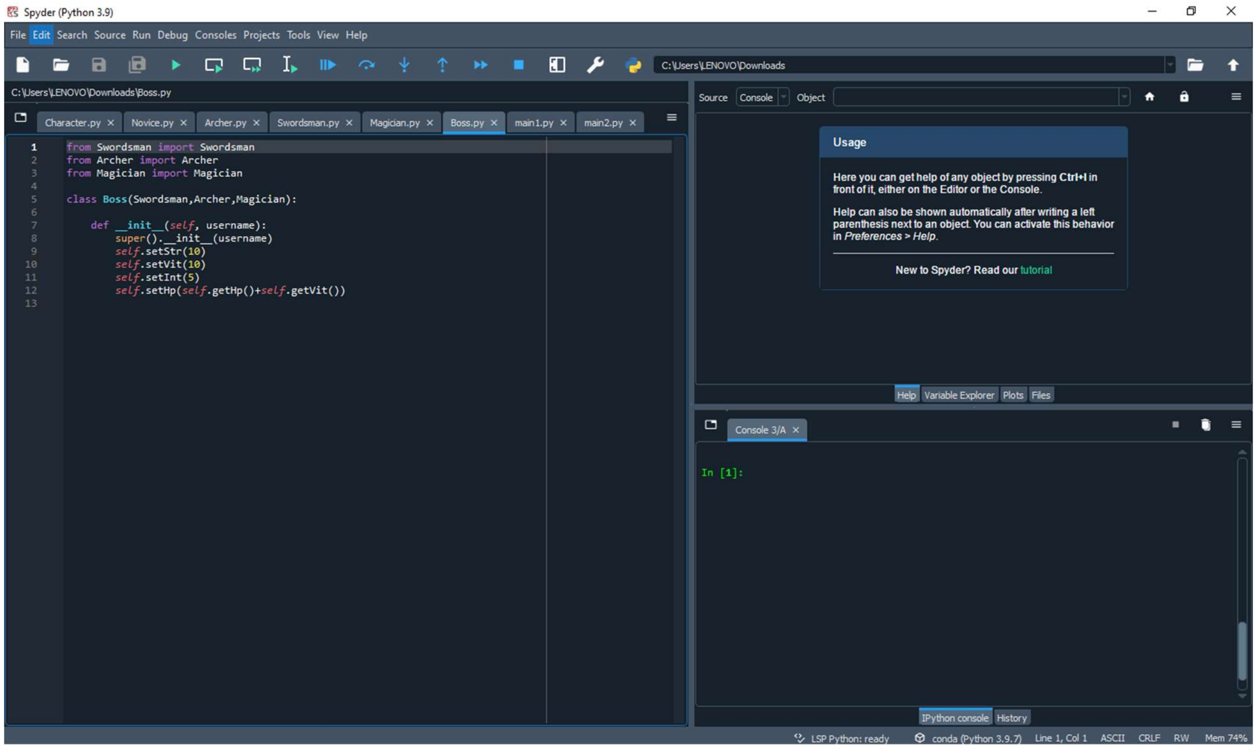
Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\LENOVO\Downloads
Character.py
Novice.py
Archer.py
Swordsman.py
Magician.py
Boss.py
main1.py
main2.py
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Help Variable Explorer Plots Files
Console 3/A
In [1]:
IPython console History
Open file
LSP Python: ready
conda (Python 3.9.7)
Line 10, Col 9
ASCII
CRLF
RW
Mem 74%



C:\Users\LENOVO\Downloads\Magician.py

```python
from Novice import Novice

class Magician(Novice):

    def __init__(self, username):
        super().__init__(username)
        self.setInt(10)
        self.setVit(5)
        self.setHp(self.getHp()+self.getVit())

    def heal(self):
        self.addHp(self.getInt())
        print(f"{self.getUsername()} performed heal! +{self.getInt()}")

    def magicAttack(self, character):
        self.new_damage = self.getDamage()+self.getInt()
        character.reduceHp(self.new_damage)
        print(f"{self.getUsername()} performed Magic Attack! - {self.new_damage}")

```

Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\LENOVO\Downloads
Character.py
Novice.py
Archer.py
Swordsman.py
Magician.py
Boss.py
main1.py
main2.py
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Help Variable Explorer Plots Files
Console 3/A
In [1]:
IPython console History
LSP Python: ready
conda (Python 3.9.7)
Line 3, Col 7
ASCII
CRLF
RW
Mem 73%



C:\Users\LENOVO\Downloads\Boss.py

```python
from Swordsman import Swordsman
from Archer import Archer
from Magician import Magician

class Boss(Swordsman,Archer,Magician):

    def __init__(self, username):
        super().__init__(username)
        self.setStr(10)
        self.setVit(10)
        self.setInt(5)
        self.setHp(self.getHp()+self.getVit())

```

Usage

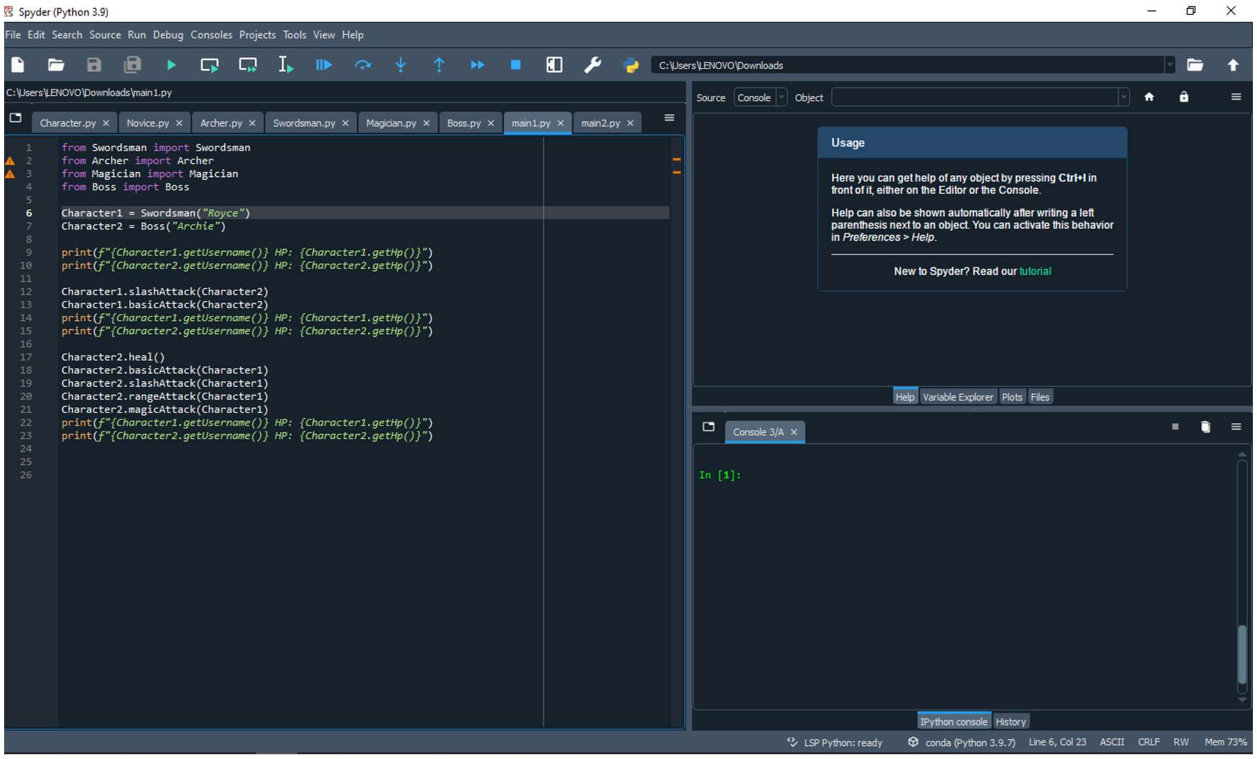
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
In [1]:
```

LSP Python: ready   conda (Python 3.9.7)   Line 1, Col 1   ASCII   CRLF   RW   Mem 74%



C:\Users\LENOVO\Downloads\main1.py

```python
from Swordsman import Swordsman
from Archer import Archer
from Magician import Magician
from Boss import Boss

Character1 = Swordsman("Royce")
Character2 = Boss("Archie")

print(f"{Character1.getUsername()} HP: {Character1.getHp()}")
print(f"{Character2.getUsername()} HP: {Character2.getHp()}")

Character1.slashAttack(Character2)
Character1.basicAttack(Character2)
print(f"{Character1.getUsername()} HP: {Character1.getHp()}")
print(f"{Character2.getUsername()} HP: {Character2.getHp()}")

Character2.heal()
Character2.basicAttack(Character1)
Character2.slashAttack(Character1)
Character2.rangeAttack(Character1)
Character2.magicAttack(Character1)
print(f"{Character1.getUsername()} HP: {Character1.getHp()}")
print(f"{Character2.getUsername()} HP: {Character2.getHp()}")



```

Usage

Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
In [1]:
```

LSP Python: ready   conda (Python 3.9.7)   Line 6, Col 23   ASCII   CRLF   RW   Mem 73%

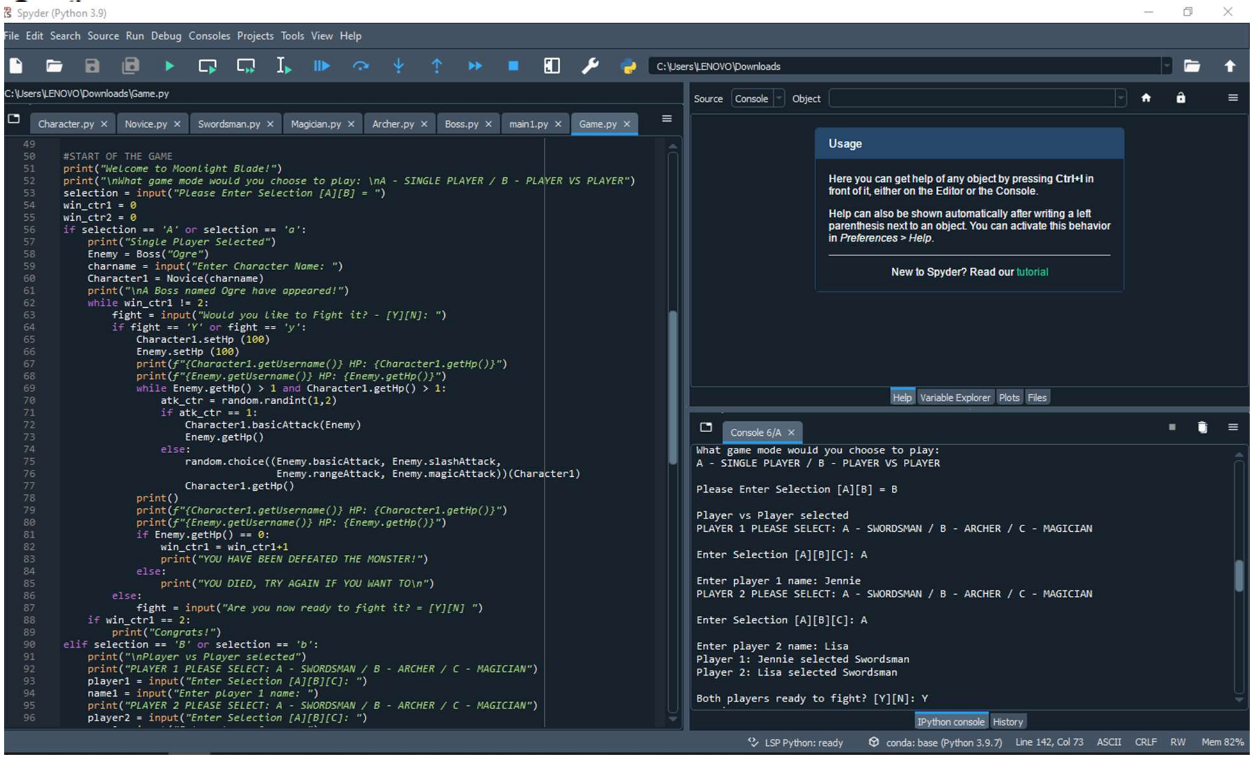
6. Supplementary Activity:

Task

Create a new file Game.py inside the same folder use the pre-made classes to create a simple Game where two players or one player vs a computer will be able to reduce their opponent's hp to 0.

Requirements:

1. The game must be able to select between 2 modes: Single player and Player vs Player. The game can spawn multiple matches where single player or player vs player can take place.
2. In Single player:
   - the player must start as a Novice, then after 2 wins, the player should be able to select a new role between Swordsman, Archer, and Magician.
   - The opponent will always be a boss named Monster.
3. In Player vs Player, both players must be able to select among all the possible roles available except Boss.
4. Turns of each player for both modes should be randomized and the match should end when one of the players hp is zero.
5. Wins of each player in a game for both the modes should be counted.



C:\Users\LENOVO\Downloads\Game.py

```python

#START OF THE GAME
print("Welcome to Moonlight Blade!")
print("\nWhat game mode would you choose to play: \nA - SINGLE PLAYER / B - PLAYER VS PLAYER")
selection = input("Please Enter Selection [A][B] = ")
win_ctr1 = 0
win_ctr2 = 0
if selection == 'A' or selection == 'a':
    print("Single Player Selected")
    Enemy = Boss("Ogre")
    charname = input("Enter Character Name: ")
    Character1 = Novice(charname)
    print("\nA Boss named Ogre have appeared!")
    while win_ctr1 != 2:
        fight = input("Would you like to Fight it? - [Y][N]: ")
        if fight == 'Y' or fight == 'y':
            Character1.setHp (100)
            Enemy.setHp (100)
            print(f"{Character1.getUsername()} HP: {Character1.getHp()}")
            print(f"{Enemy.getUsername()} HP: {Enemy.getHp()}")
            while Enemy.getHp() > 1 and Character1.getHp() > 1:
                atk_ctr = random.randint(1,2)
                if atk_ctr == 1:
                    Character1.basicAttack(Enemy)
                    Enemy.getHp()
                else:
                    random.choice((Enemy.basicAttack, Enemy.slashAttack,
                                   Enemy.rangeAttack, Enemy.magicAttack))(Character1)
                    Character1.getHp()
            print()
            print(f"{Character1.getUsername()} HP: {Character1.getHp()}")
            print(f"{Enemy.getUsername()} HP: {Enemy.getHp()}")
            if Enemy.getHp() == 0:
                win_ctr1 = win_ctr1+1
                print("YOU HAVE BEEN DEFEATED THE MONSTER!")
            else:
                print("YOU DIED, TRY AGAIN IF YOU WANT TO\n")
        else:
            fight = input("Are you now ready to fight it? = [Y][N] ")
    if win_ctr1 == 2:
        print("Congrats!")
elif selection == 'B' or selection == 'b':
    print("\nPlayer vs Player selected")
    print("PLAYER 1 PLEASE SELECT: A - SWORDSMAN / B - ARCHER / C - MAGICIAN")
    player1 = input("Enter Selection [A][B][C]: ")
    name1 = input("Enter player 1 name: ")
    print("PLAYER 2 PLEASE SELECT: A - SWORDSMAN / B - ARCHER / C - MAGICIAN")
    player2 = input("Enter Selection [A][B][C]: ")
```

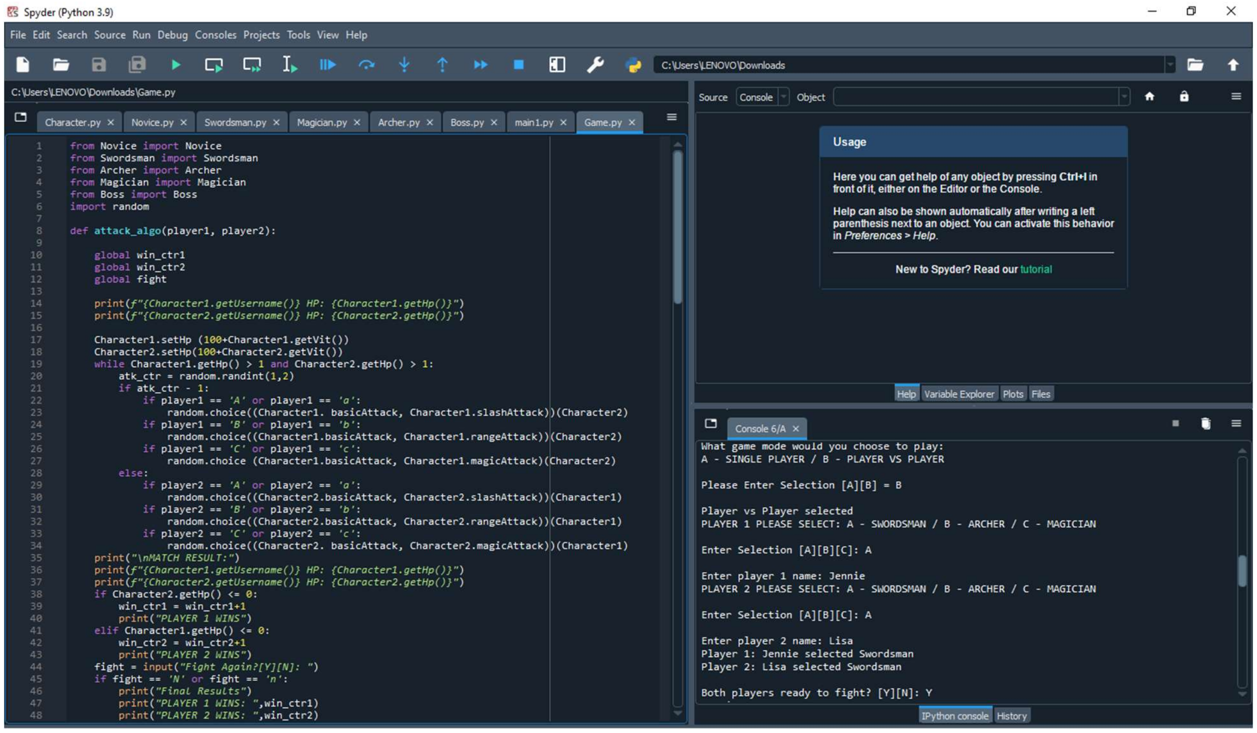
Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
What game mode would you choose to play:
A - SINGLE PLAYER / B - PLAYER VS PLAYER

Please Enter Selection [A][B] = B

Player vs Player selected
PLAYER 1 PLEASE SELECT: A - SWORDSMAN / B - ARCHER / C - MAGICIAN

Enter Selection [A][B][C]: A

Enter player 1 name: Jennie
PLAYER 2 PLEASE SELECT: A - SWORDSMAN / B - ARCHER / C - MAGICIAN

Enter Selection [A][B][C]: A

Enter player 2 name: Lisa
Player 1: Jennie selected Swordsman
Player 2: Lisa selected Swordsman

Both players ready to fight? [Y][N]: Y
```



C:\Users\LENOVO\Downloads\Game.py

```python
from Novice import Novice
from Swordsman import Swordsman
from Archer import Archer
from Magician import Magician
from Boss import Boss
import random

def attack_algo(player1, player2):

    global win_ctr1
    global win_ctr2
    global fight

    print(f"{Character1.getUsername()} HP: {Character1.getHp()}")
    print(f"{Character2.getUsername()} HP: {Character2.getHp()}")

    Character1.setHp (100+Character1.getVit())
    Character2.setHp(100+Character2.getVit())
    while Character1.getHp() > 1 and Character2.getHp() > 1:
        atk_ctr = random.randint(1,2)
        if atk_ctr - 1:
            if player1 == 'A' or player1 == 'a':
                random.choice((Character1. basicAttack, Character1.slashAttack))(Character2)
            if player1 == 'B' or player1 == 'b':
                random.choice((Character1.basicAttack, Character1.rangeAttack))(Character2)
            if player1 == 'C' or player1 == 'c':
                random.choice (Character1.basicAttack, Character1.magicAttack)(Character2)
        else:
            if player2 == 'A' or player2 == 'a':
                random.choice((Character2.basicAttack, Character2.slashAttack))(Character1)
            if player2 == 'B' or player2 == 'b':
                random.choice((Character2.basicAttack, Character2.rangeAttack))(Character1)
            if player2 == 'C' or player2 == 'c':
                random.choice((Character2. basicAttack, Character2.magicAttack))(Character1)
    print("\nMATCH RESULT:")
    print(f"{Character1.getUsername()} HP: {Character1.getHp()}")
    print(f"{Character2.getUsername()} HP: {Character2.getHp()}")
    if Character2.getHp() <= 0:
        win_ctr1 = win_ctr1+1
        print("PLAYER 1 WINS")
    elif Character1.getHp() <= 0:
        win_ctr2 = win_ctr2+1
        print("PLAYER 2 WINS")
    fight = input("Fight Again?[Y][N]: ")
    if fight == 'N' or fight == 'n':
        print("Final Results")
        print("PLAYER 1 WINS: ",win_ctr1)
        print("PLAYER 2 WINS: ",win_ctr2)
```

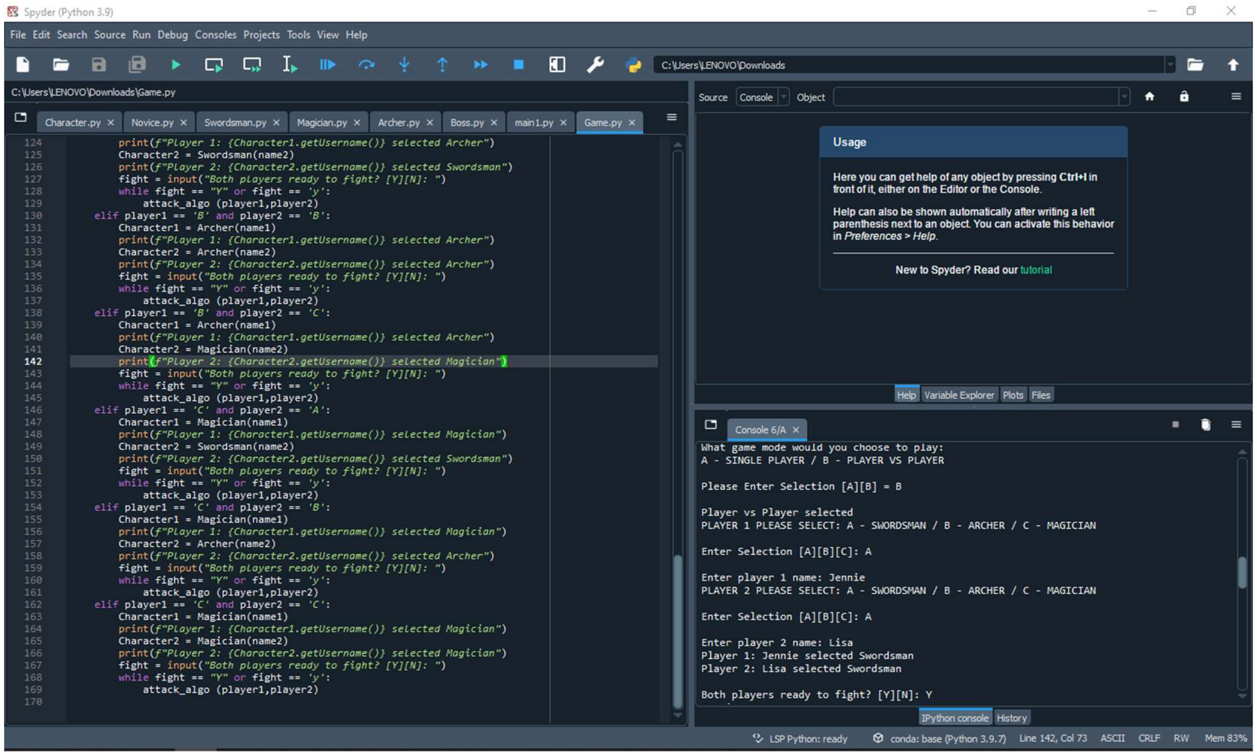
Usage

Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
What game mode would you choose to play:
A - SINGLE PLAYER / B - PLAYER VS PLAYER

Please Enter Selection [A][B] = B

Player vs Player selected
PLAYER 1 PLEASE SELECT: A - SWORDSMAN / B - ARCHER / C - MAGICIAN

Enter Selection [A][B][C]: A

Enter player 1 name: Jennie
PLAYER 2 PLEASE SELECT: A - SWORDSMAN / B - ARCHER / C - MAGICIAN

Enter Selection [A][B][C]: A

Enter player 2 name: Lisa
Player 1: Jennie selected Swordsman
Player 2: Lisa selected Swordsman

Both players ready to fight? [Y][N]: Y
```



C:\Users\LENOVO\Downloads\Game.py

```python
            print(f"Player 1: {Character1.getUsername()} selected Archer")
            Character2 = Swordsman(name2)
            print(f"Player 2: {Character2.getUsername()} selected Swordsman")
            fight = input("Both players ready to fight? [Y][N]: ")
            while fight == "Y" or fight == 'y':
                attack_algo (player1,player2)
        elif player1 == 'B' and player2 == 'B':
            Character1 = Archer(name1)
            print(f"Player 1: {Character1.getUsername()} selected Archer")
            Character2 = Archer(name2)
            print(f"Player 2: {Character2.getUsername()} selected Archer")
            fight = input("Both players ready to fight? [Y][N]: ")
            while fight == "Y" or fight == 'y':
                attack_algo (player1,player2)
        elif player1 == 'B' and player2 == 'C':
            Character1 = Archer(name1)
            print(f"Player 1: {Character1.getUsername()} selected Archer")
            Character2 = Magician(name2)
            print(f"Player 2: {Character2.getUsername()} selected Magician")
            fight = input("Both players ready to fight? [Y][N]: ")
            while fight == "Y" or fight == 'y':
                attack_algo (player1,player2)
        elif player1 == 'C' and player2 == 'A':
            Character1 = Magician(name1)
            print(f"Player 1: {Character1.getUsername()} selected Magician")
            Character2 = Swordsman(name2)
            print(f"Player 2: {Character2.getUsername()} selected Swordsman")
            fight = input("Both players ready to fight? [Y][N]: ")
            while fight == "Y" or fight == 'y':
                attack_algo (player1,player2)
        elif player1 == 'C' and player2 == 'B':
            Character1 = Magician(name1)
            print(f"Player 1: {Character1.getUsername()} selected Magician")
            Character2 = Archer(name2)
            print(f"Player 2: {Character2.getUsername()} selected Archer")
            fight = input("Both players ready to fight? [Y][N]: ")
            while fight == "Y" or fight == 'y':
                attack_algo (player1,player2)
        elif player1 == 'C' and player2 == 'C':
            Character1 = Magician(name1)
            print(f"Player 1: {Character1.getUsername()} selected Magician")
            Character2 = Magician(name2)
            print(f"Player 2: {Character2.getUsername()} selected Magician")
            fight = input("Both players ready to fight? [Y][N]: ")
            while fight == "Y" or fight == 'y':
                attack_algo (player1,player2)
```

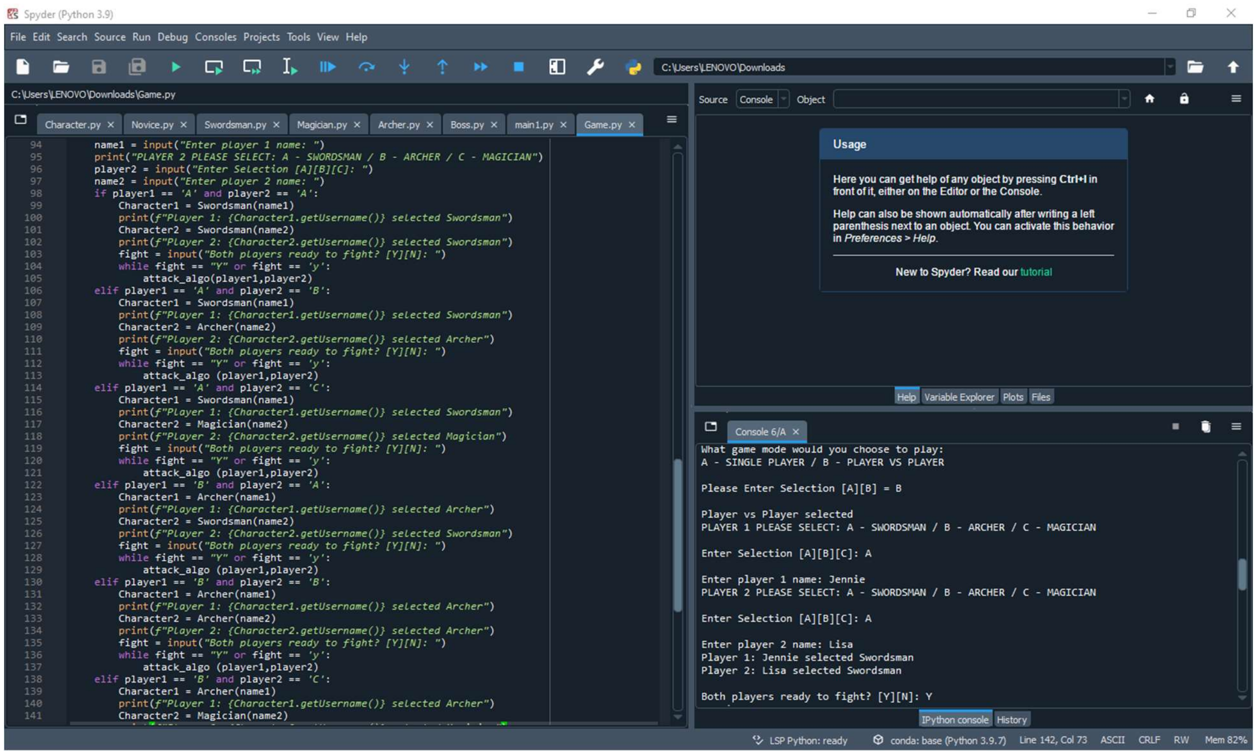
Usage

Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
What game mode would you choose to play:
A - SINGLE PLAYER / B - PLAYER VS PLAYER

Please Enter Selection [A][B] = B

Player vs Player selected
PLAYER 1 PLEASE SELECT: A - SWORDSMAN / B - ARCHER / C - MAGICIAN

Enter Selection [A][B][C]: A

Enter player 1 name: Jennie
PLAYER 2 PLEASE SELECT: A - SWORDSMAN / B - ARCHER / C - MAGICIAN

Enter Selection [A][B][C]: A

Enter player 2 name: Lisa
Player 1: Jennie selected Swordsman
Player 2: Lisa selected Swordsman

Both players ready to fight? [Y][N]: Y
```



```
        name1 = input("Enter player 1 name: ")
        print("PLAYER 2 PLEASE SELECT: A - SWORDSMAN / B - ARCHER / C - MAGICIAN")
        player2 = input("Enter Selection [A][B][C]: ")
        name2 = input("Enter player 2 name: ")
        if player1 == 'A' and player2 == 'A':
            Character1 = Swordsman(name1)
            print(f"Player 1: {Character1.getUsername()} selected Swordsman")
            Character2 = Swordsman(name2)
            print(f"Player 2: {Character2.getUsername()} selected Swordsman")
            fight = input("Both players ready to fight? [Y][N]: ")
            while fight == "Y" or fight == 'y':
                attack_algo(player1,player2)
        elif player1 == 'A' and player2 == 'B':
            Character1 = Swordsman(name1)
            print(f"Player 1: {Character1.getUsername()} selected Swordsman")
            Character2 = Archer(name2)
            print(f"Player 2: {Character2.getUsername()} selected Archer")
            fight = input("Both players ready to fight? [Y][N]: ")
            while fight == "Y" or fight == 'y':
                attack_algo (player1,player2)
        elif player1 == 'A' and player2 == 'C':
            Character1 = Swordsman(name1)
            print(f"Player 1: {Character1.getUsername()} selected Swordsman")
            Character2 = Magician(name2)
            print(f"Player 2: {Character2.getUsername()} selected Magician")
            fight = input("Both players ready to fight? [Y][N]: ")
            while fight == "Y" or fight == 'y':
                attack_algo (player1,player2)
        elif player1 == 'B' and player2 == 'A':
            Character1 = Archer(name1)
            print(f"Player 1: {Character1.getUsername()} selected Archer")
            Character2 = Swordsman(name2)
            print(f"Player 2: {Character2.getUsername()} selected Swordsman")
            fight = input("Both players ready to fight? [Y][N]: ")
            while fight == "Y" or fight == 'y':
                attack_algo (player1,player2)
        elif player1 == 'B' and player2 == 'B':
            Character1 = Archer(name1)
            print(f"Player 1: {Character1.getUsername()} selected Archer")
            Character2 = Archer(name2)
            print(f"Player 2: {Character2.getUsername()} selected Archer")
            fight = input("Both players ready to fight? [Y][N]: ")
            while fight == "Y" or fight == 'y':
                attack_algo (player1,player2)
        elif player1 == 'B' and player2 == 'C':
            Character1 = Archer(name1)
            print(f"Player 1: {Character1.getUsername()} selected Archer")
            Character2 = Magician(name2)
```

Usage

Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
What game mode would you choose to play:
A - SINGLE PLAYER / B - PLAYER VS PLAYER

Please Enter Selection [A][B] = B

Player vs Player selected
PLAYER 1 PLEASE SELECT: A - SWORDSMAN / B - ARCHER / C - MAGICIAN

Enter Selection [A][B][C]: A

Enter player 1 name: Jennie
PLAYER 2 PLEASE SELECT: A - SWORDSMAN / B - ARCHER / C - MAGICIAN

Enter Selection [A][B][C]: A

Enter player 2 name: Lisa
Player 1: Jennie selected Swordsman
Player 2: Lisa selected Swordsman

Both players ready to fight? [Y][N]: Y
```

Questions

1. **Why is Inheritance important?**

Inheritance is important because they provide reusability and they help programmers reuse code. in inheritance children class inherits the traits of the parent class.

2. **Explain the advantages and disadvantages of using applying inheritance in an Object-Oriented Program.**

Inheritance makes programmers life easier because can easily identify which is which and inheritance provides reusability of codes to the program because when we create a new class we can let it inherit the traits of the parent class instead of creating it again from scratch. While the disadvantage of inheritance for me is that sometimes it is confusing and the children class or the subclass becomes dependent on the parent class since the sub class only inherited the traits of the parent class.

3**. Differentiate single inheritance, multiple inheritance, and multi-level inheritance.**

In single inheritance it means that it does not include different levels of inheritance and only inherits single derived class from a single base class. In multiple inheritance a class can inherit features from more than one base class or parent class. Multilevel inheritance is like multiple inheritance but the newly created subclass becomes the base or parent class for another new class.

4**. Why is super().__init__(username) added in the codes of Swordsman, Archer, Magician, and Boss?**

It gives its child class access to all of its methods and returns an object that represents the parent class. It's also used to make single and multiple inheritances possible. To name its access and methods, we don't need to specify its parent class.

5. **How do you think Encapsulation and Abstraction helps in making good Object-Oriented Programs?**

Data is hidden or protected through encapsulation, which involves combining portions. OOP encapsulation is a design concept. Meaning that only members of that class may access the object's contents. Encapsulated objects are not accessible by clients. A degree of encapsulation conceals complex characteristics. Less human errors and simple app maintenance. Simplifies the app's use. Languages provide control abstraction as one of their basic features. Computers can move bits around in memory and add two sequences of bits. Higher-level programming languages help.

Conclusion:

Creating a program with the use of object-oriented programming and with the knowledge of using the different kinds of inheritance is helpful in doing such advance program since using inheritance makes it easier than creating every class from scratch but by using inheritance you will be able to program faster since the newly created classes can inherit the traits that the previous class have.

Honor Pledge for Grouped Projects " We accept responsibility for our role in ensuring the integrity of the work submitted by the group in which we participated"