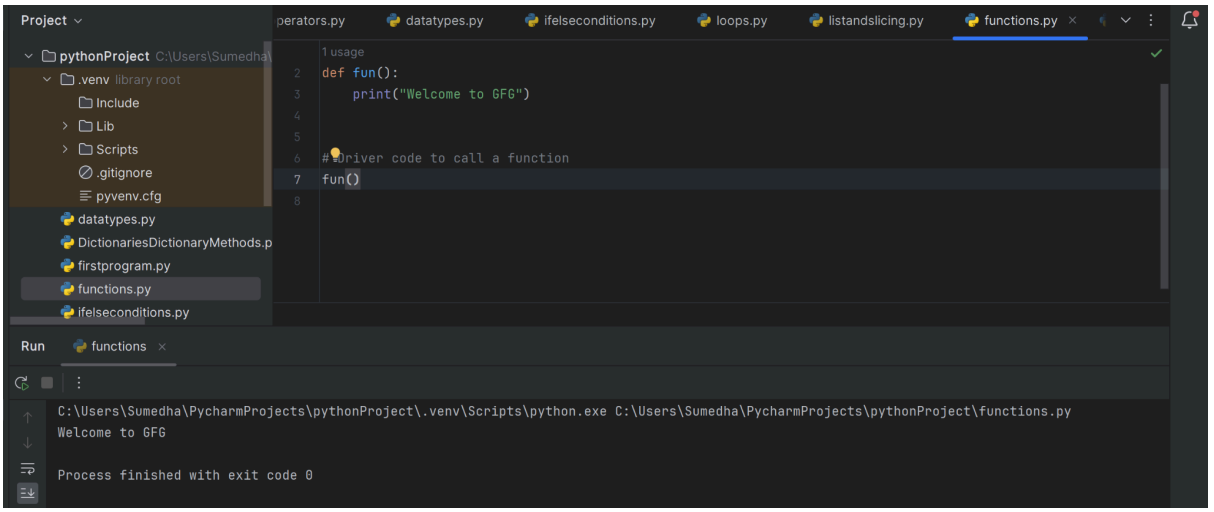
NAME: AKULA SHARATH CHANDRA
BATCH: DATA ENGINEERING
TOPICS: FUNCTIONS ,OOPS CONCEPTS

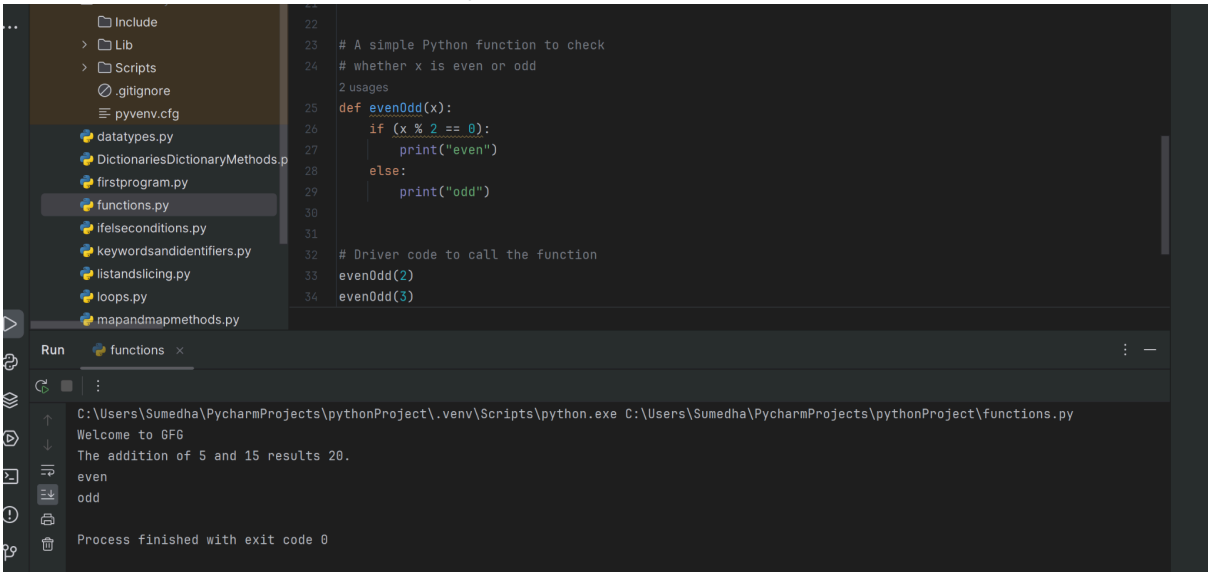
1)FUNCTIONS : Python Functions is a block of statements that return the specific task.

→ Defining and calling a function:



→Defining and calling a function with parameters:

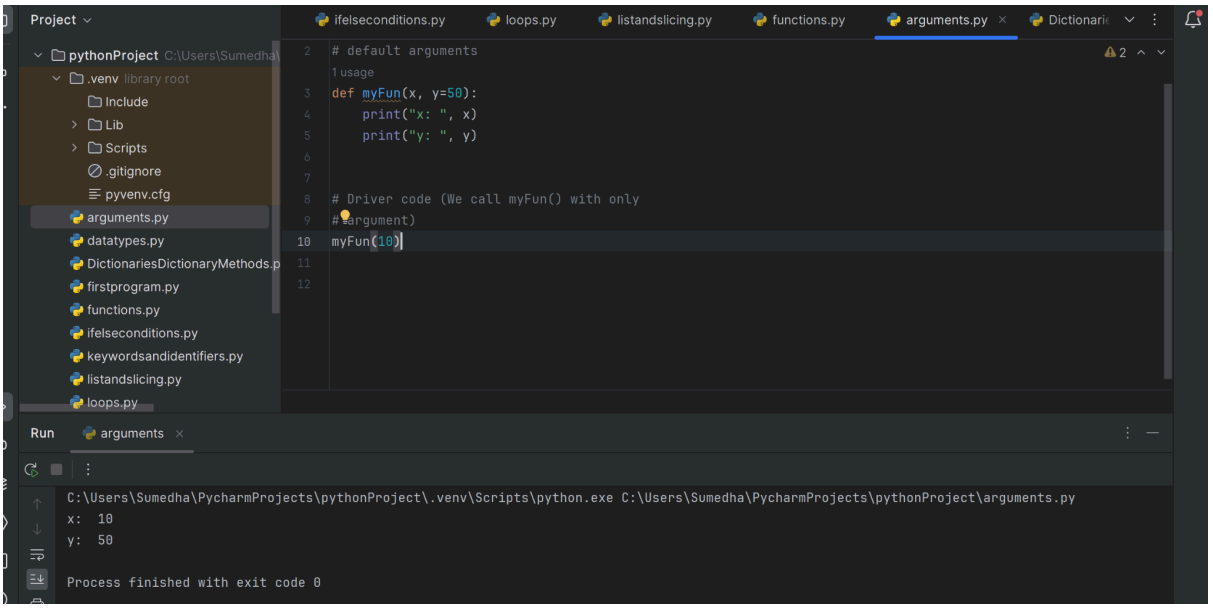
→**Python Function Arguments:**Arguments are the values passed inside the parentheses of the function. A function can have any number of arguments separated by a comma.



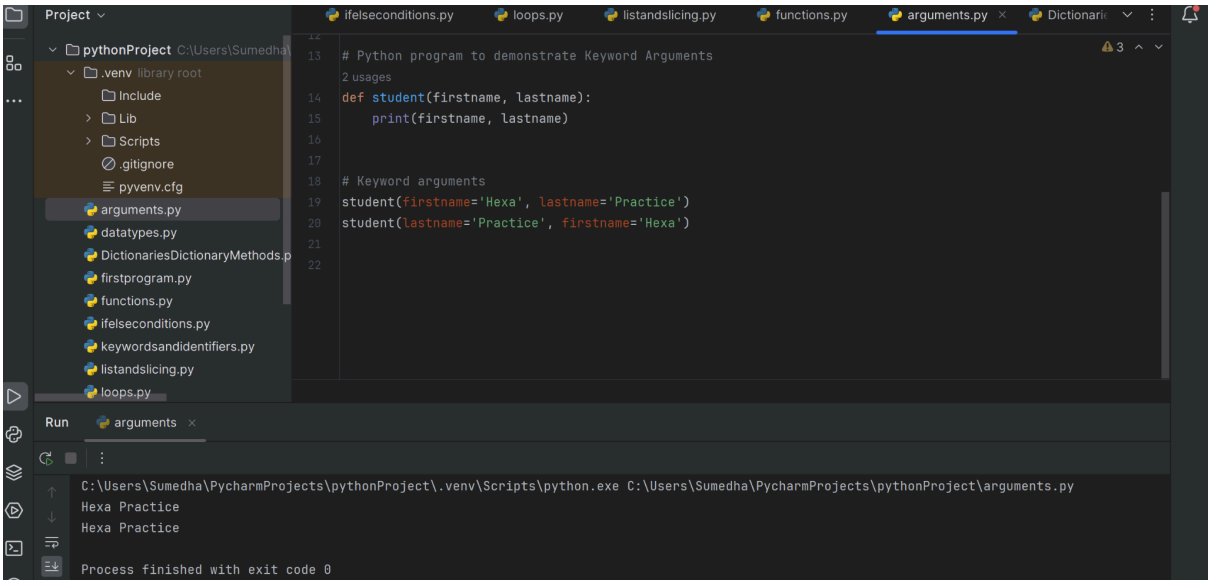
→**Types of Python Function Arguments**

- **Default argument**
- **Keyword arguments (named arguments)**
- **Positional arguments**
- **Arbitrary arguments** (variable-length arguments *args and **kwargs)

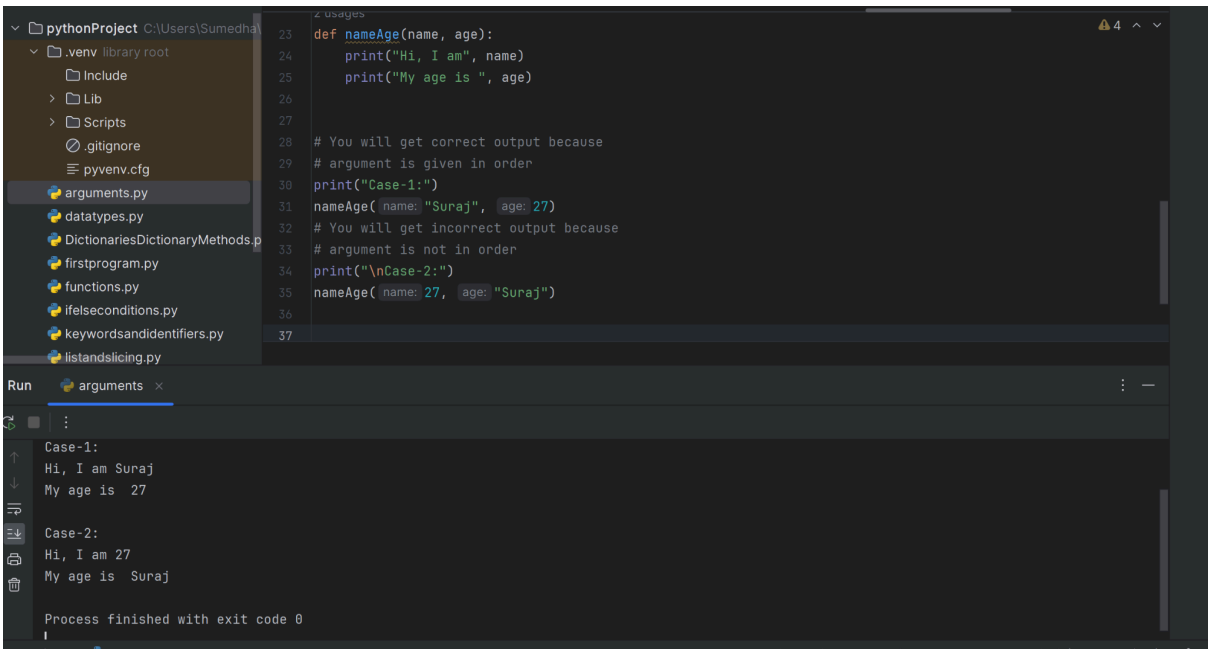
i)**DEFAULT ARGUMENT:** A Default argument is a parameter that assumes a default value if a value is not provided in the function call for that argument.



ii)Keyword arguments (named arguments): In Python, keyword arguments allow passing values to a function by explicitly associating them with parameter names. This provides clarity, flexibility, and allows for non-sequential argument passing.

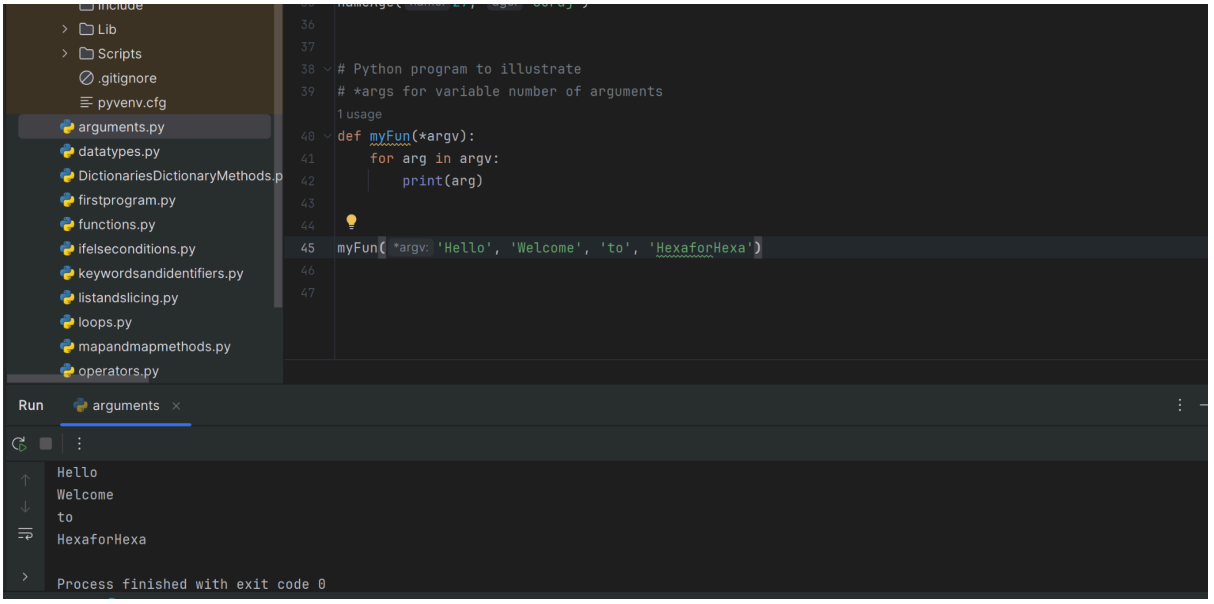


iii)Positional arguments: Positional arguments in Python are passed based on their order in the function signature. Values are assigned to parameters in the order they appear, making the argument order crucial. This is the traditional method of passing arguments.

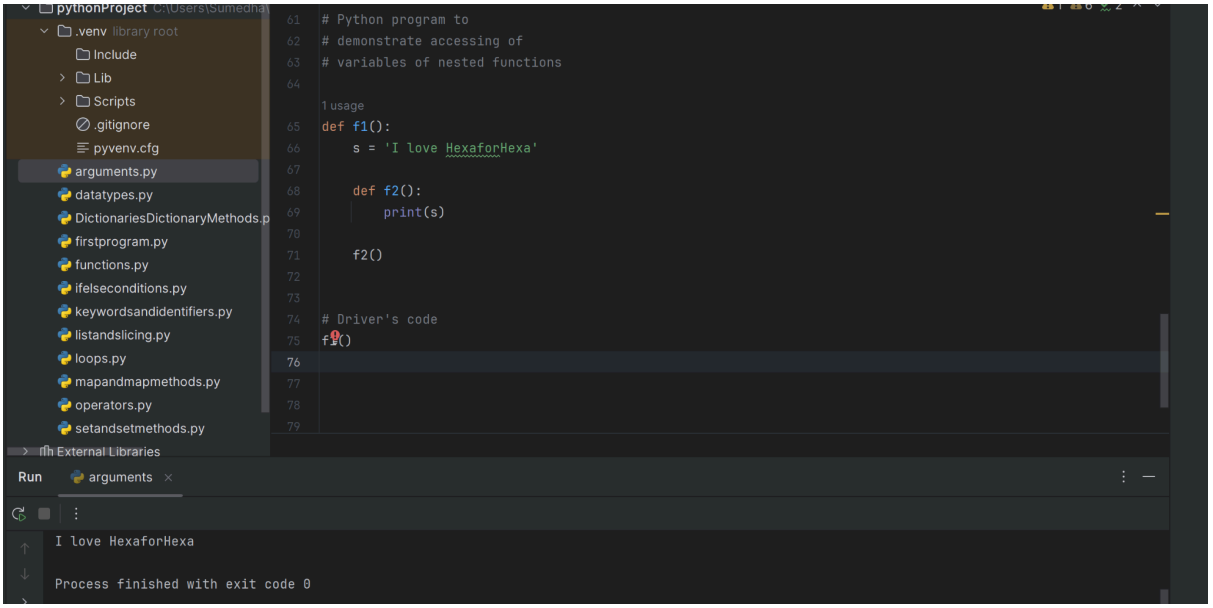


iv)Arbitrary Keyword Arguments: In Python Arbitrary Keyword Arguments , *args and **kwargs can pass a variable number of arguments to a function using special symbols. There are two special symbols:

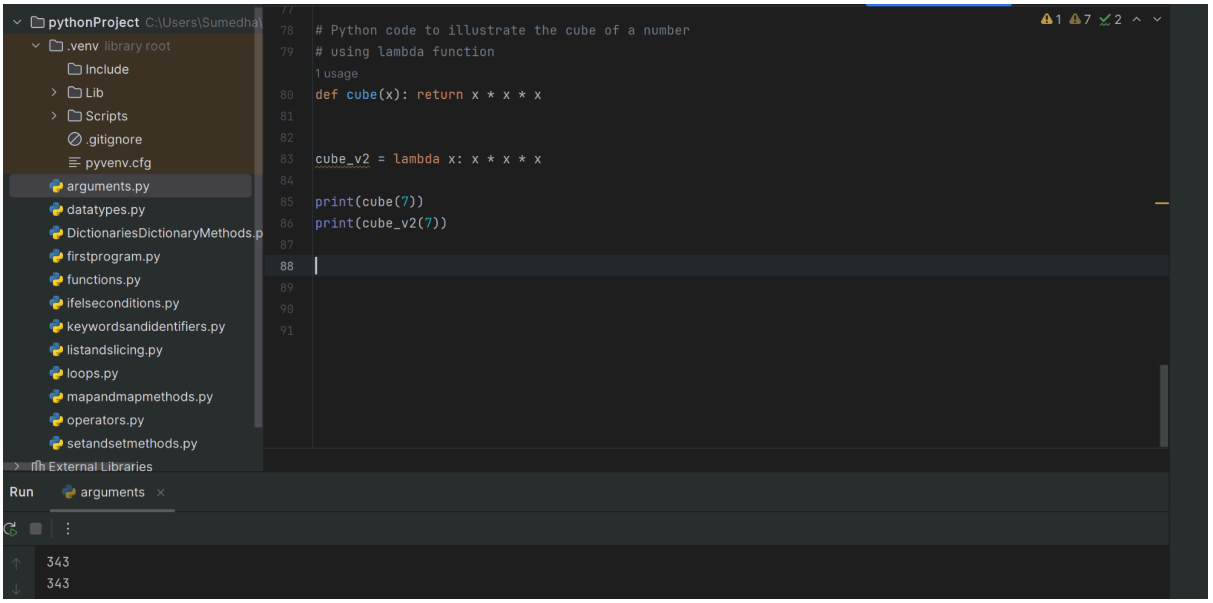
- *args in Python (Non-Keyword Arguments)
- **kwargs in Python (Keyword Arguments)



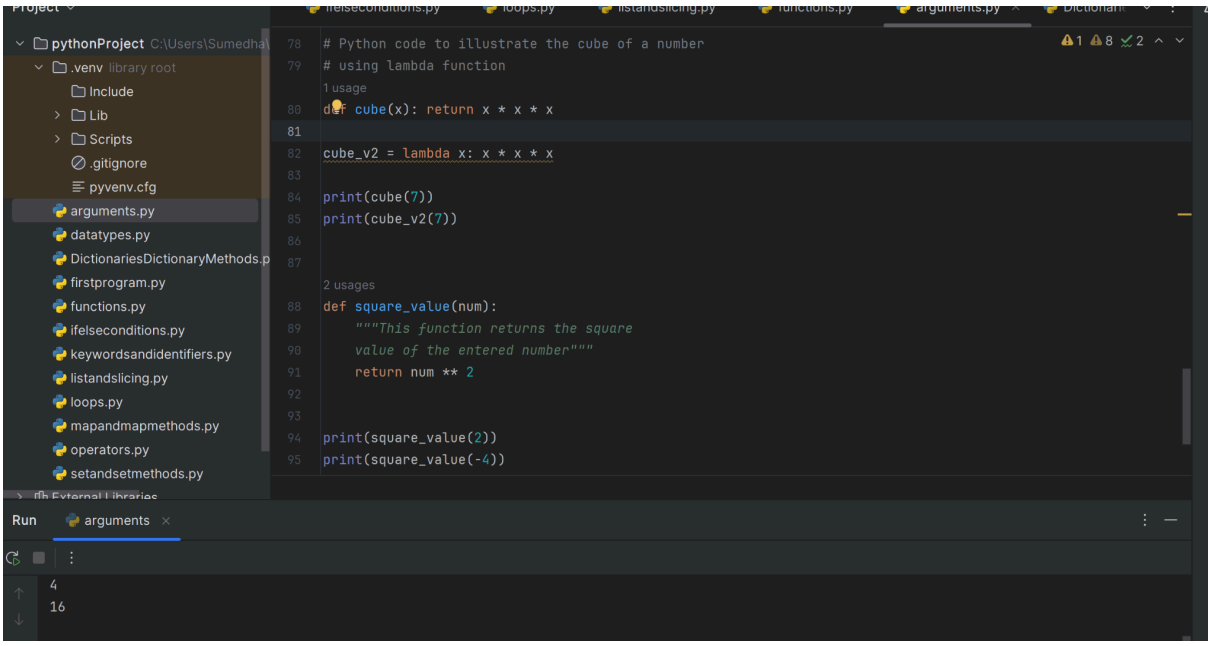
—> **Function within Functions:** A function that is defined inside another function is known as the inner function or nested function.



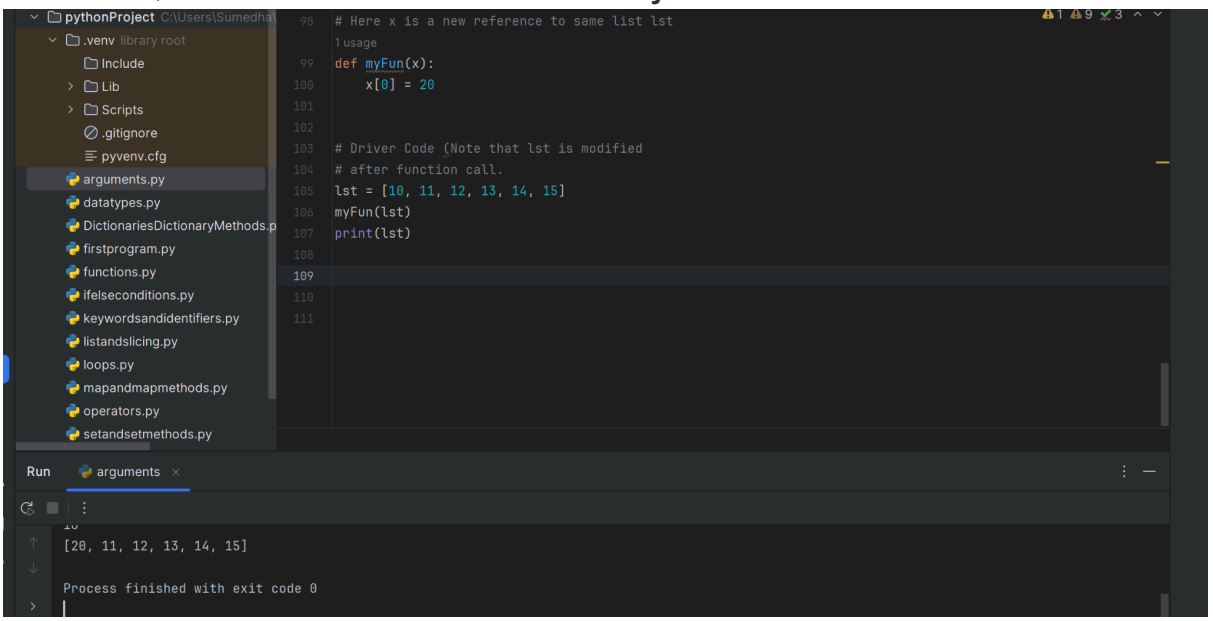
→**Anonymous Functions in Python:** A function without a name is called an anonymous function.



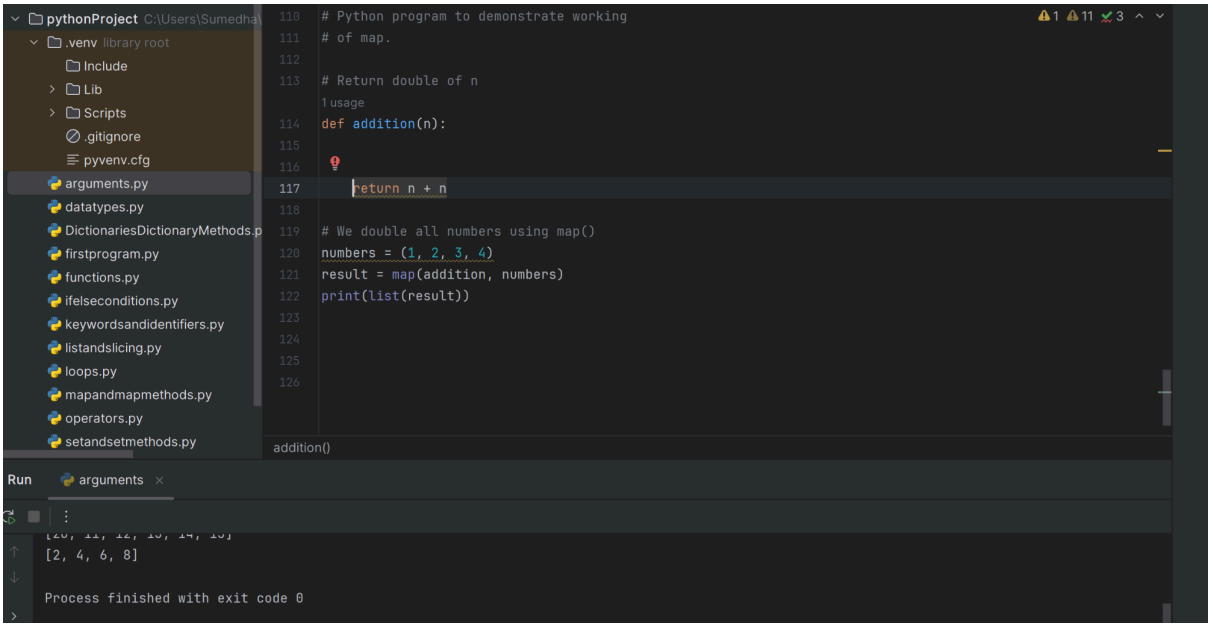
→ **Return Statement in Python Function:**The function return statement is used to exit from a function and go back to the function caller and return the specified value or data item to the caller.



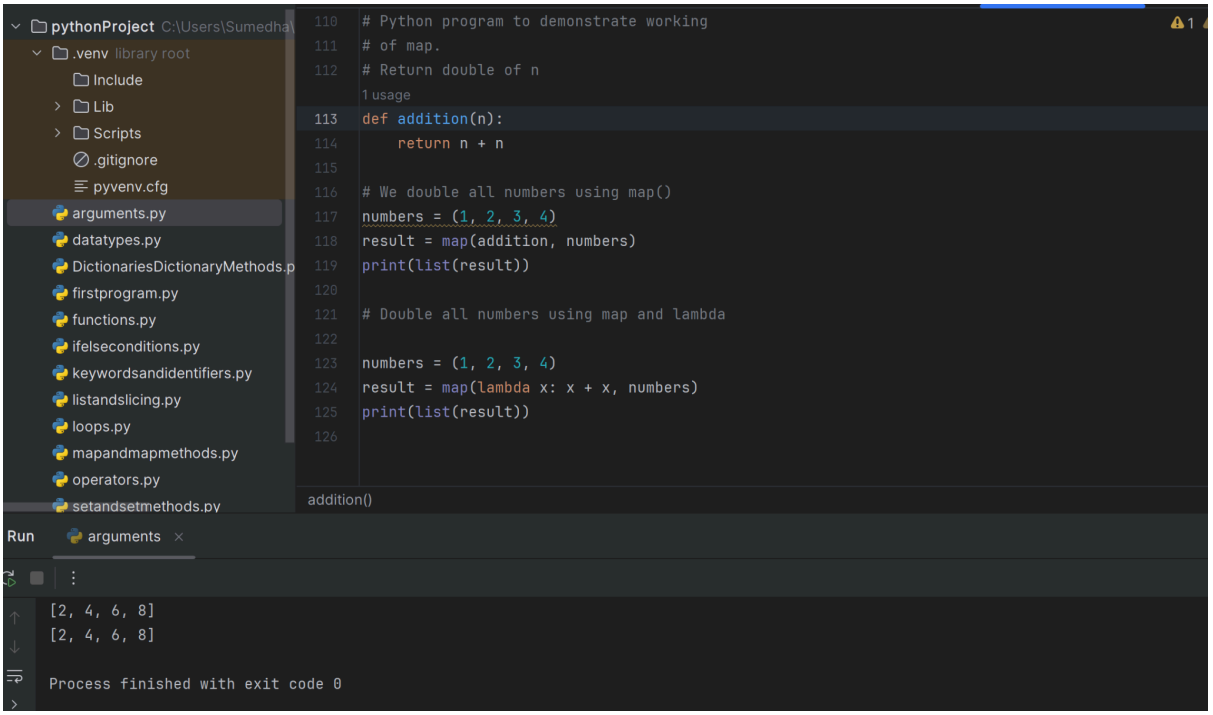
—>**Pass by Reference and Pass by Value:** In Python every variable name is a reference. When we pass a variable to a function, a new reference to the object is created.



→**Python map() Function:**map() function returns a map object(which is an iterator) of the results after applying the given function to each item of a given iterable (list, tuple etc.)
Syntax: map(fun, iter)

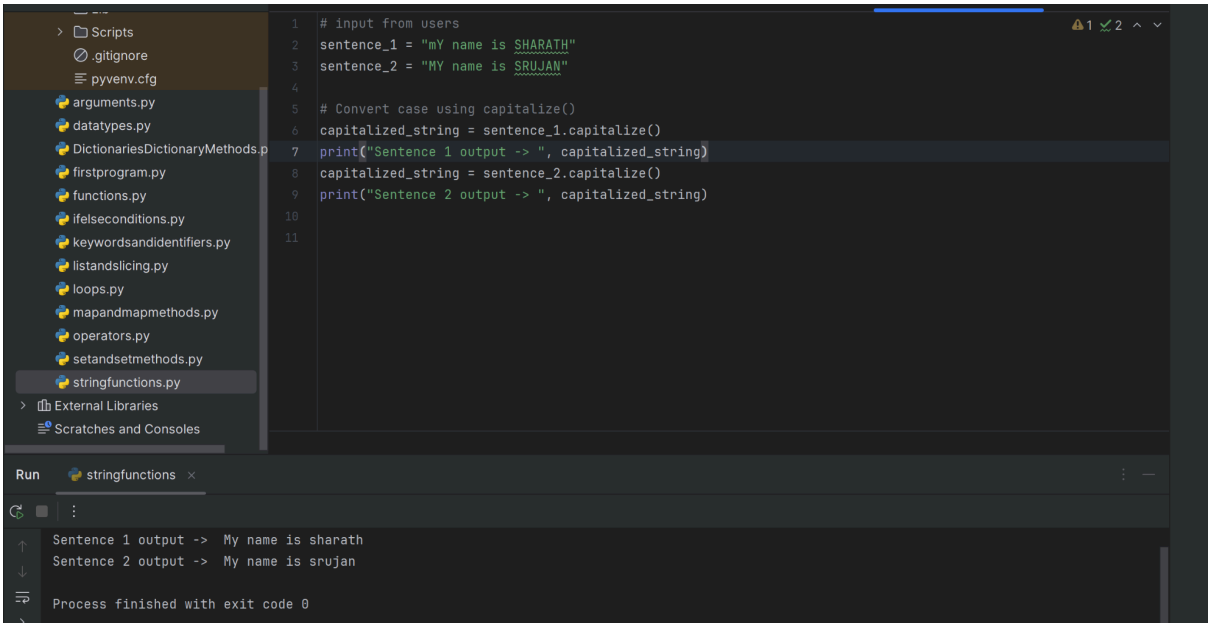


→map() with Lambda Expressions:

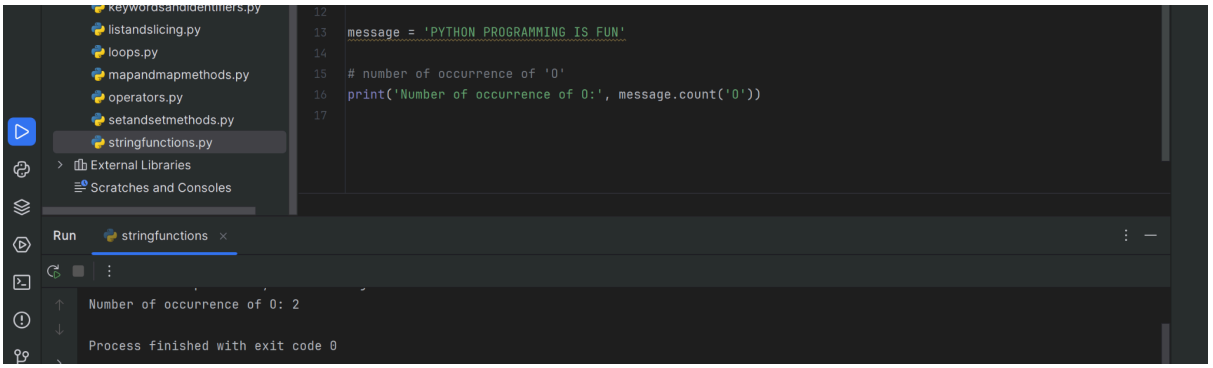


—>String Functions in Python:

i) Capitalize: The capitalize() is used in Python where the first letter of the string is converted into UPPERCASE and the rest of the characters remain the same. On the other hand, if all the characters are in UPPERCASE then the string will return the same value (except the first character).

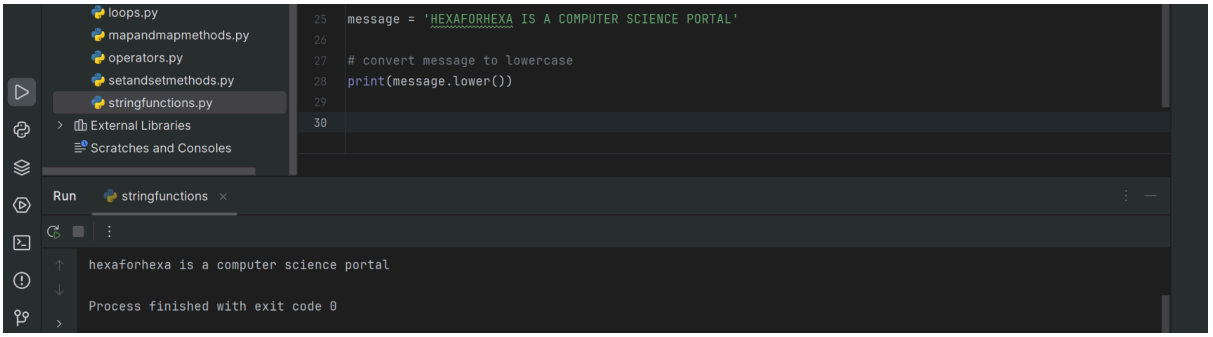


ii)Count: The count() throws the numeric value that provides the detail of an actual count of a given string.

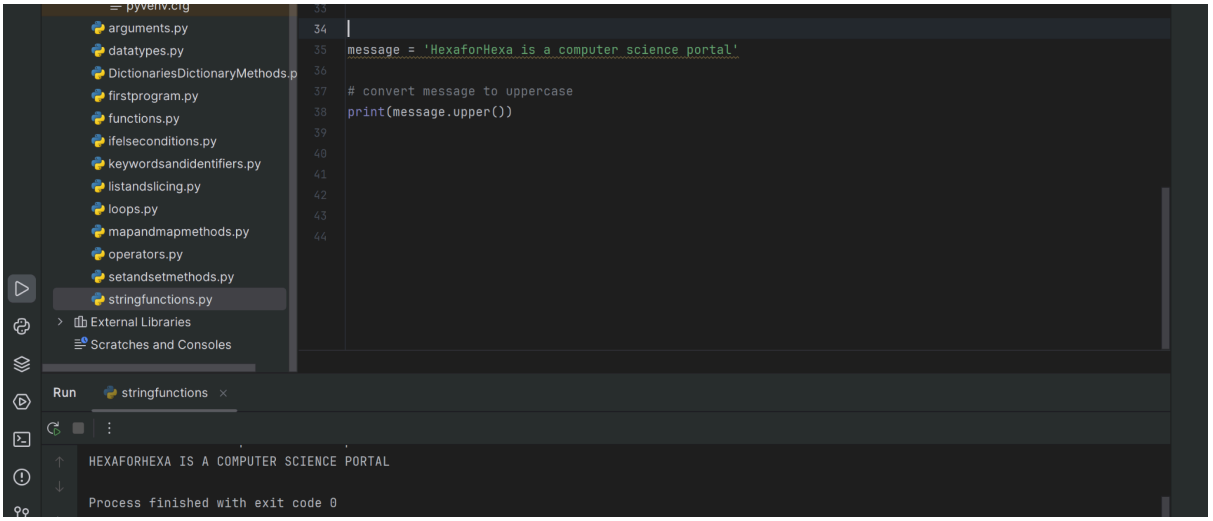


iii)**Find:**The find()is used in Python to return the lowest index value from the first occurrence of a string (only in case it's found): else the value would be -1.

iv) **Lower:** The lower()is used in Python programming to ensure that all the UPPERCASE characters in the string are converted into lowercase and fetched with a new lowercase string and the original copy of the string remains intact.



v)**Upper:**The upper() is used in Python programming to ensure that all the lowercase characters in the string are converted into UPPERCASE and fetched with a new string whereas the original copy of the string remains intact.



vi)**Replace:**The replace() is used in Python to replace any unwanted character or text and replace it with the new desired output within the string.

vii)**JOIN():** join() returns a concatenated string and it will throw a TypeError exception if the iterable contains any non-string element within it.

2)OOPs Concepts in Python:

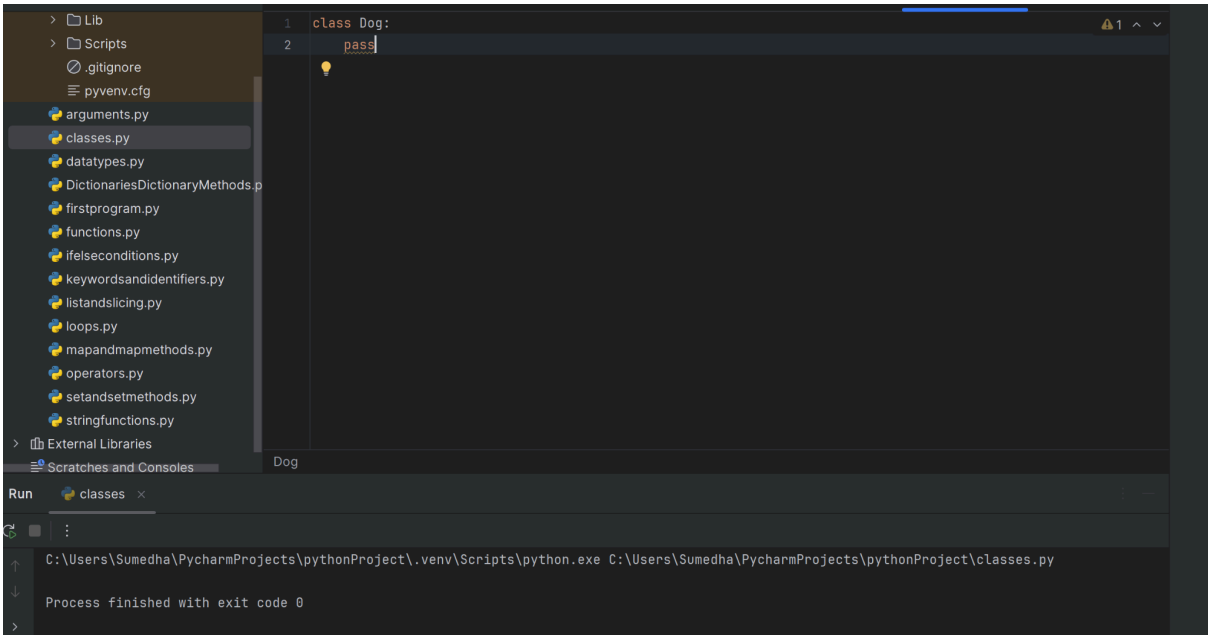
- Class
- Objects
- Polymorphism
- Encapsulation
- Inheritance
- Data Abstraction

i) Class: A class is a collection of objects. A class contains the blueprints or the prototype from which the objects are being created. It is a logical entity that contains some attributes and methods.

Class Definition Syntax:

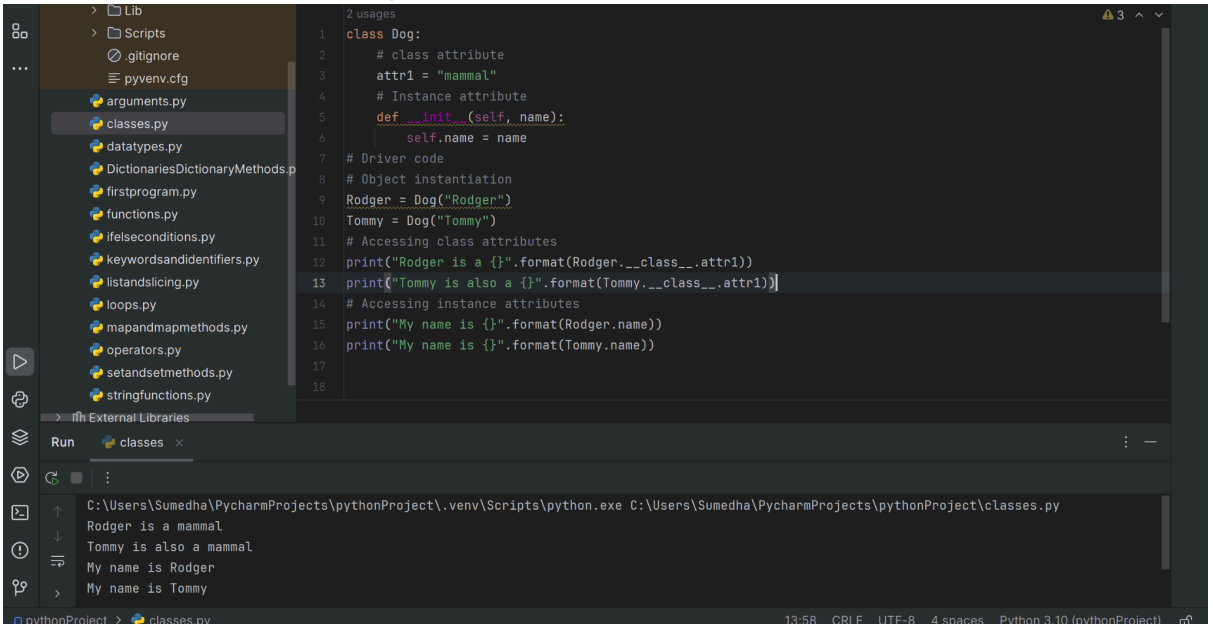
```
class ClassName:
   # Statement-1
   .
   .
   .
   # Statement-N
```

Creating an Empty Class in Python:



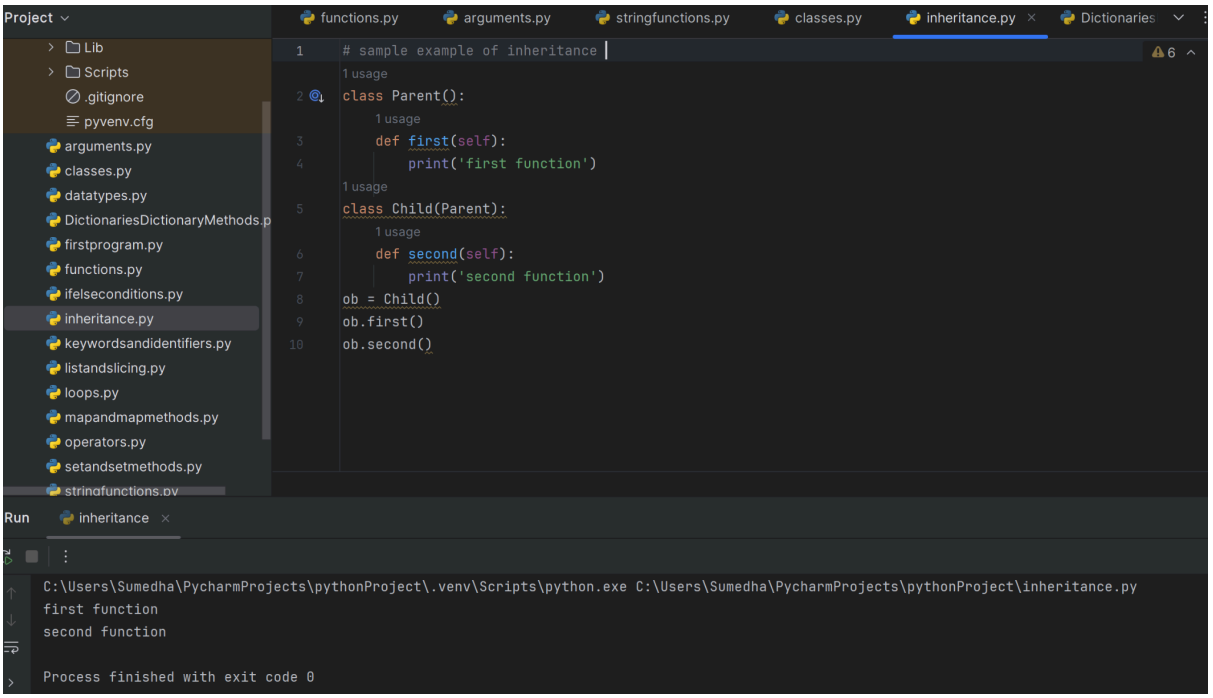
ii)Objects: The object is an entity that has a state and behaviour associated with it.

Creating a class and object with class and instance attributes:



Creating Classes and objects with methods:

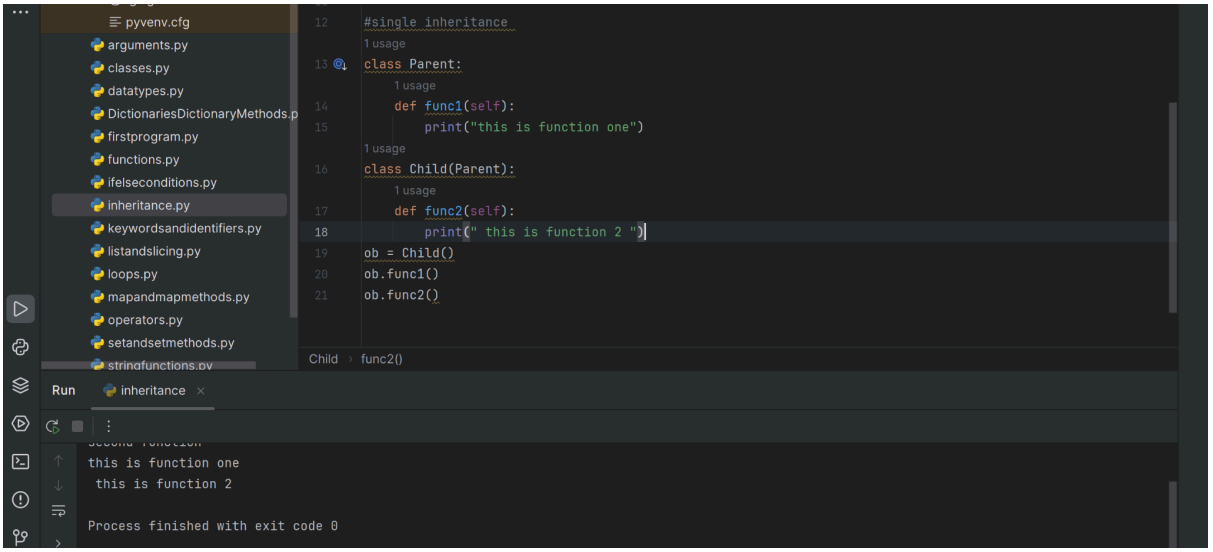
iii) Inheritance: Inheritance allows a class (subclass/derived class) to inherit attributes and methods from another class (superclass/base class).It provides the reusability of a code. We don't have to write the same code again and again. Also, it allows us to add more features to a class without modifying it.



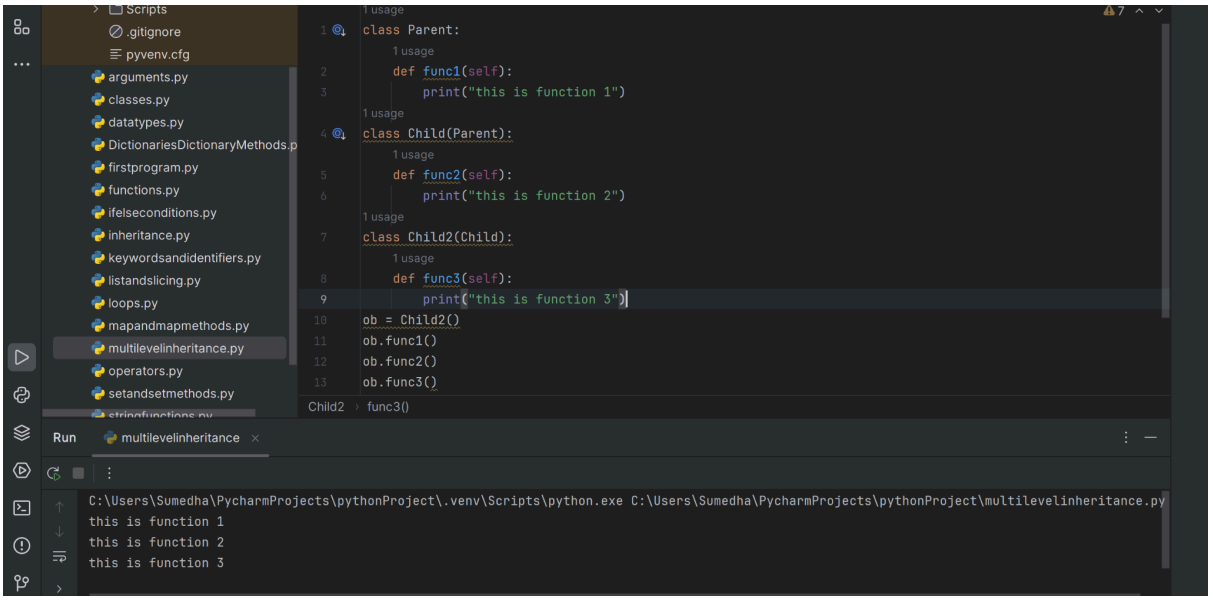
Types of Inheritance:

- **Single Inheritance**: Single-level inheritance enables a derived class to inherit characteristics from a single-parent class.
- **Multilevel Inheritance:** Multi-level inheritance enables a derived class to inherit properties from an immediate parent class which in turn inherits properties from his parent class.
- **Hierarchical Inheritance:** Hierarchical-level inheritance enables more than one derived class to inherit properties from a parent class.
- **Multiple Inheritance:** Multiple-level inheritance enables one derived class to inherit properties from more than one base class.

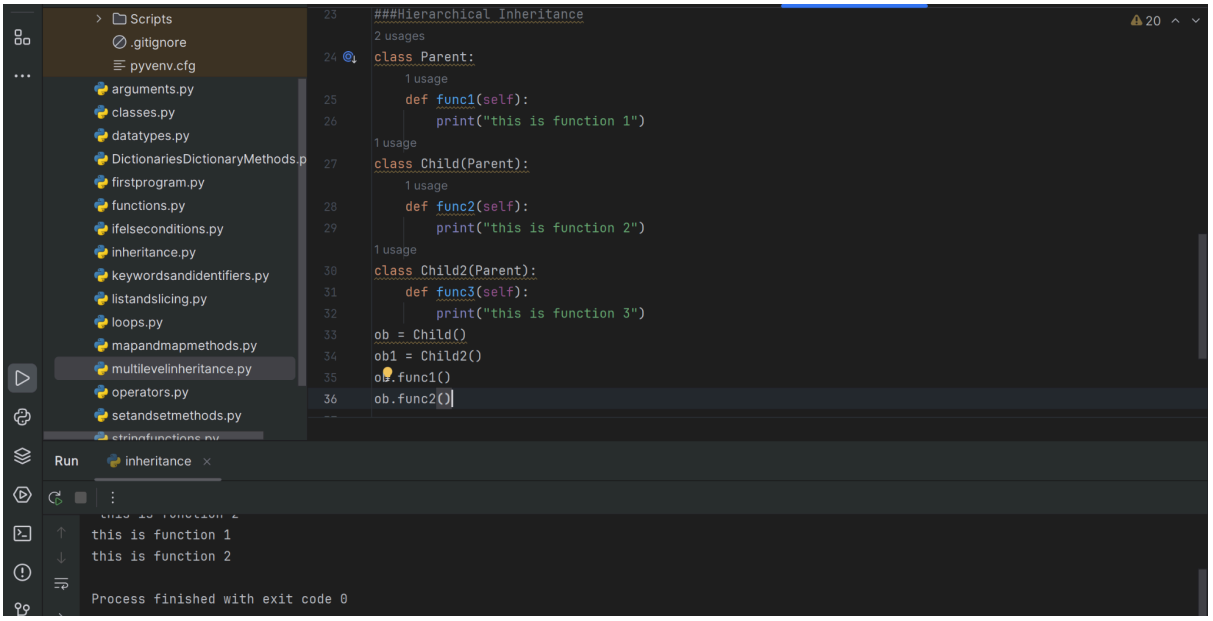
i) single inheritance : When a child class inherits only a single parent class



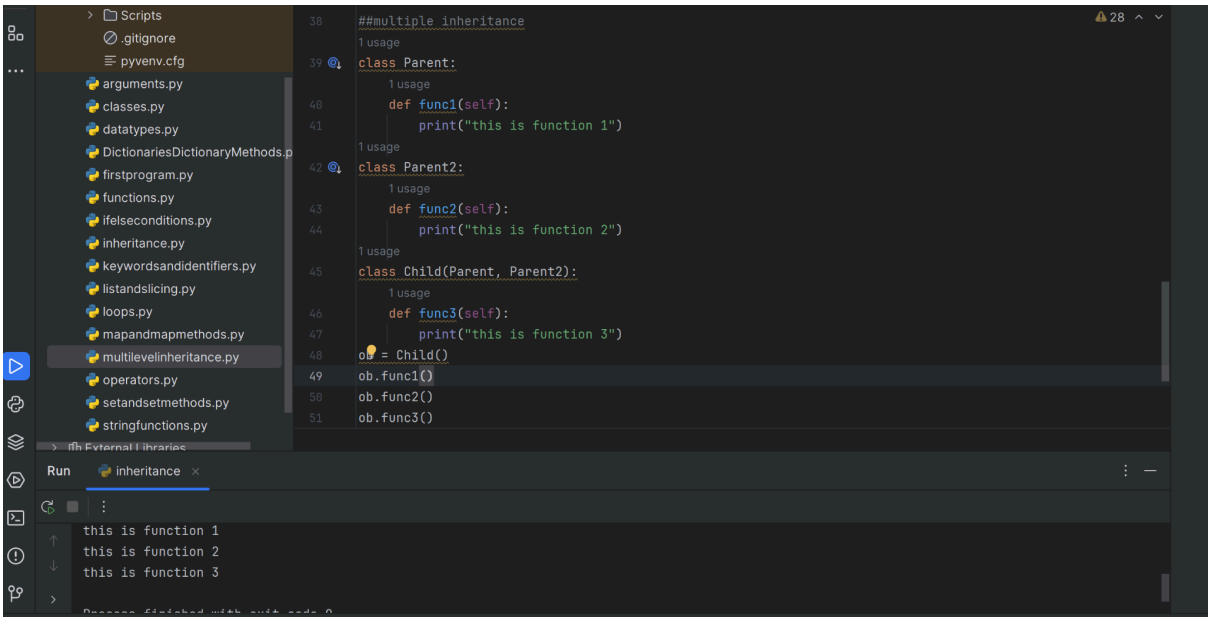
ii)Multilevel Inheritance: When a child class becomes a parent class for another child class.



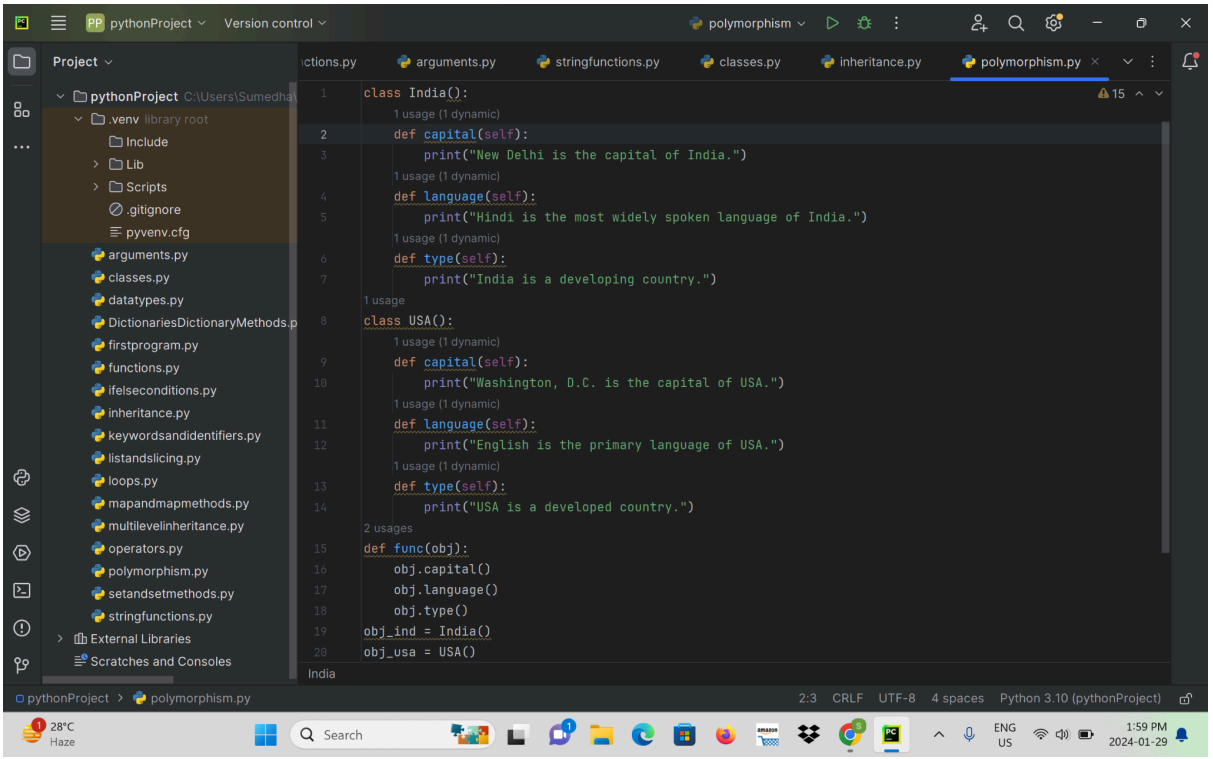
iii)Hierarchical Inheritance: Hierarchical inheritance involves multiple inheritance from the same base or parent class.

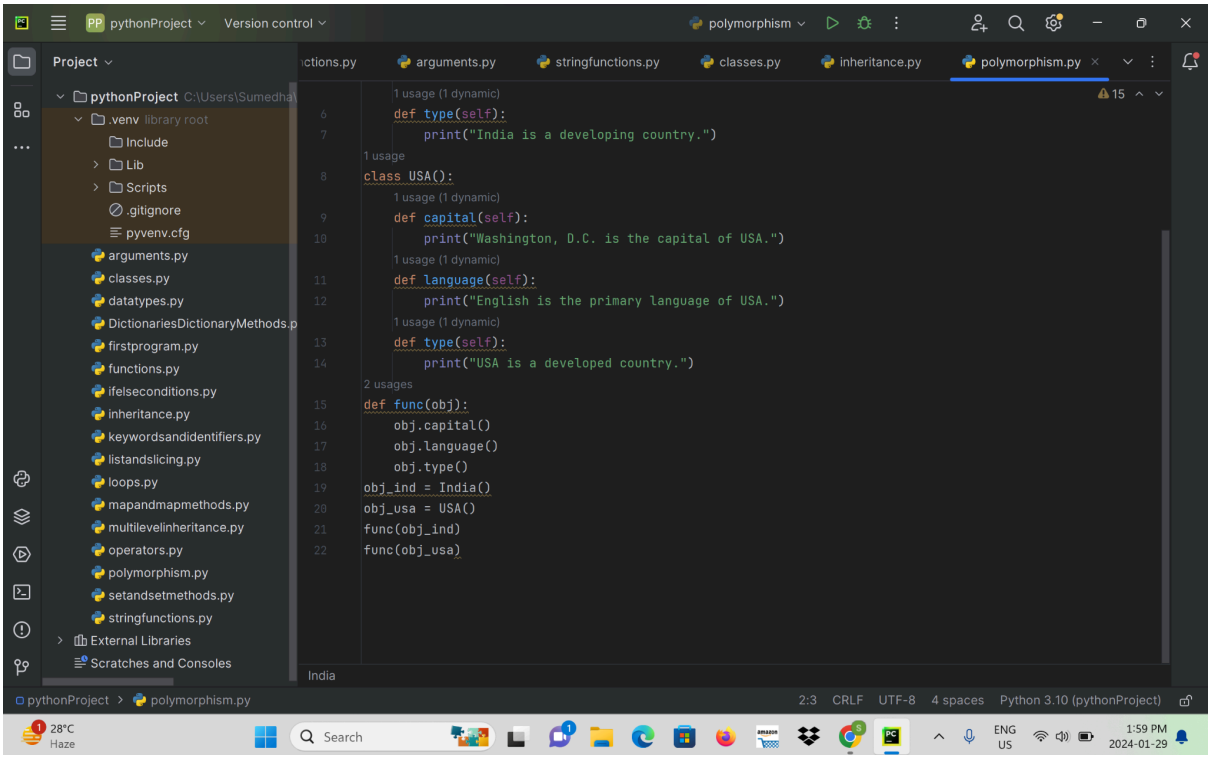


iv)**Multiple Inheritance:**When a child class inherits from more than one parent class.

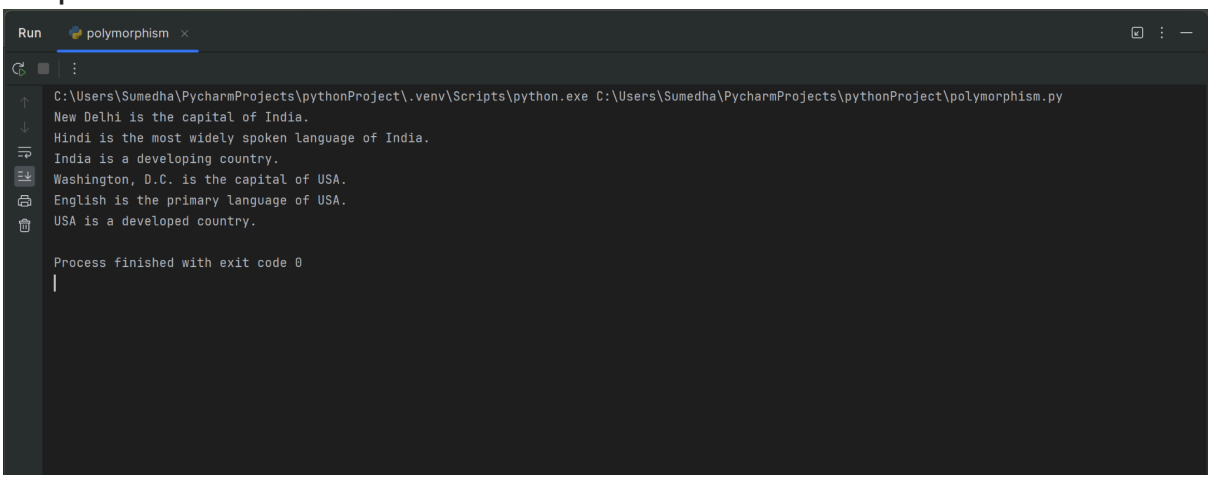


iv) Polymorphism: Polymorphism means taking different forms. In programming, it enables operators, functions, and methods to act differently when subjected to different conditions.



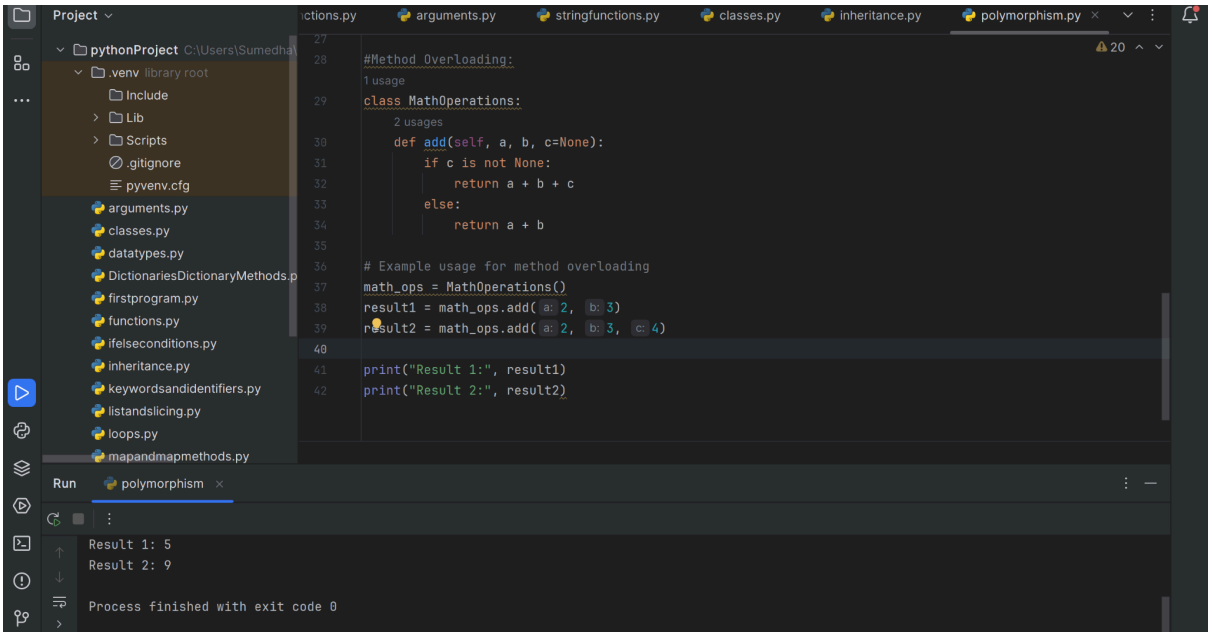


Output:

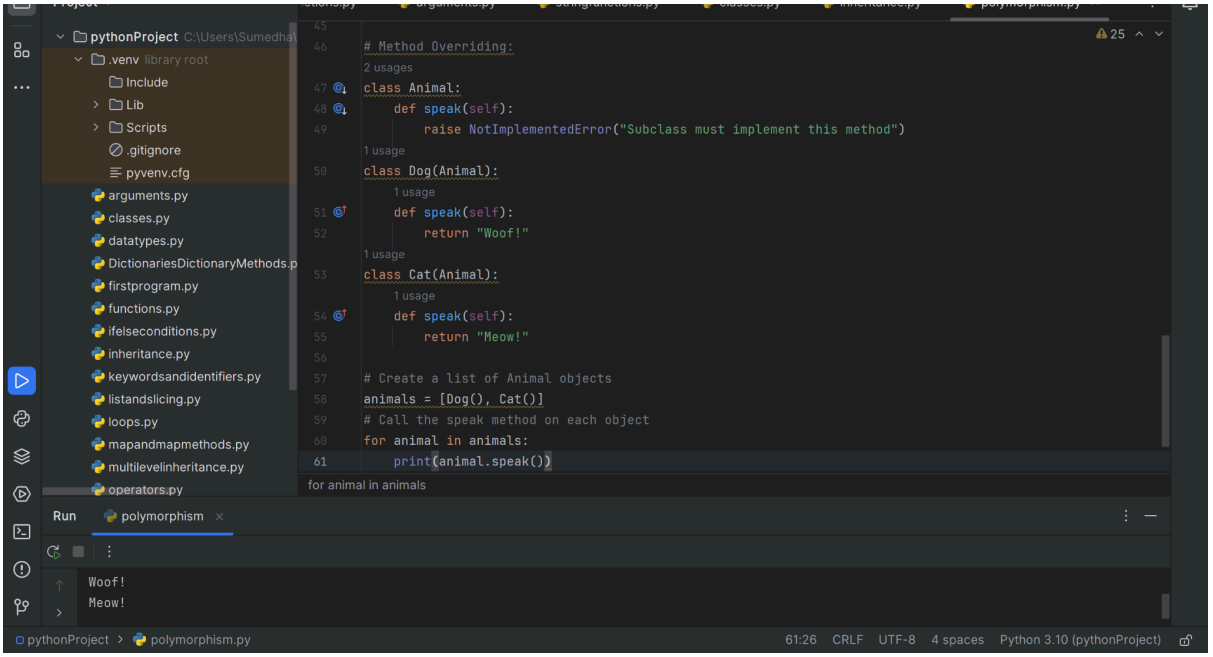


→METHOD OVERLOADING AND METHOD OVERRIDING:

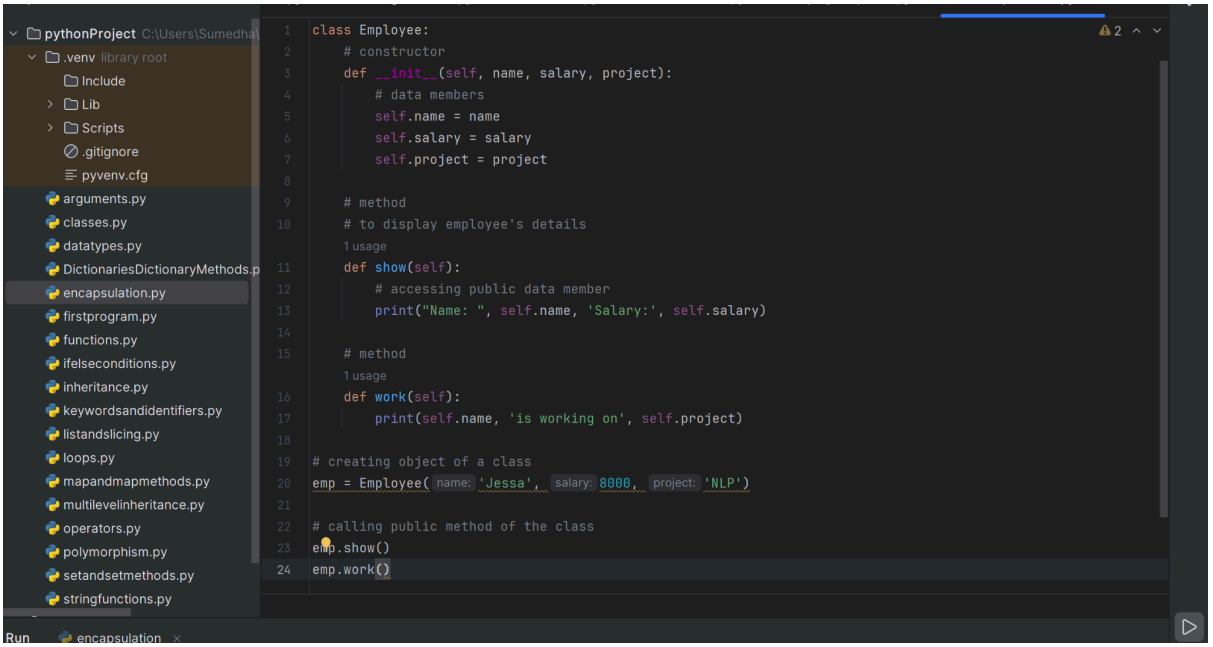
i)METHOD OVERLOADING: Method overloading in Python refers to the ability to define multiple methods in the same class with the same name but different parameters.



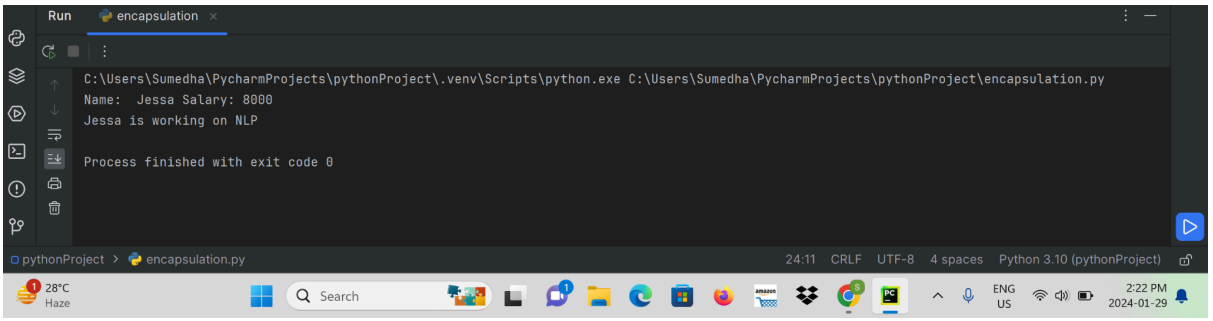
ii) METHOD OVERRIDING: Method overriding occurs when a subclass provides a specific implementation for a method that is already defined in its superclass. This allows a subclass to customise or extend the behaviour of the inherited method.


pythonProject C:\Users\Sumedha\
.venv library root
Include
Lib
Scripts
.gitignore
pyvenv.cfg
arguments.py
classes.py
datatypes.py
DictionariesDictionaryMethods.p
firstprogram.py
functions.py
ifelseconditions.py
inheritance.py
keywordsandidentifiers.py
listandslicing.py
loops.py
mapandmapmethods.py
multilevelinheritance.py
operators.py
Method Overriding:
2 usages
class Animal:
def speak(self):
raise NotImplementedError("Subclass must implement this method")
1 usage
class Dog(Animal):
1 usage
def speak(self):
return "Woof!"
1 usage
class Cat(Animal):
1 usage
def speak(self):
return "Meow!"
# Create a list of Animal objects
animals = [Dog(), Cat()]
# Call the speak method on each object
for animal in animals:
print(animal.speak())
for animal in animals
Run
polymorphism
Woof!
Meow!
pythonProject
polymorphism.py
61:26
CRLF
UTF-8
4 spaces
Python 3.10 (pythonProject)

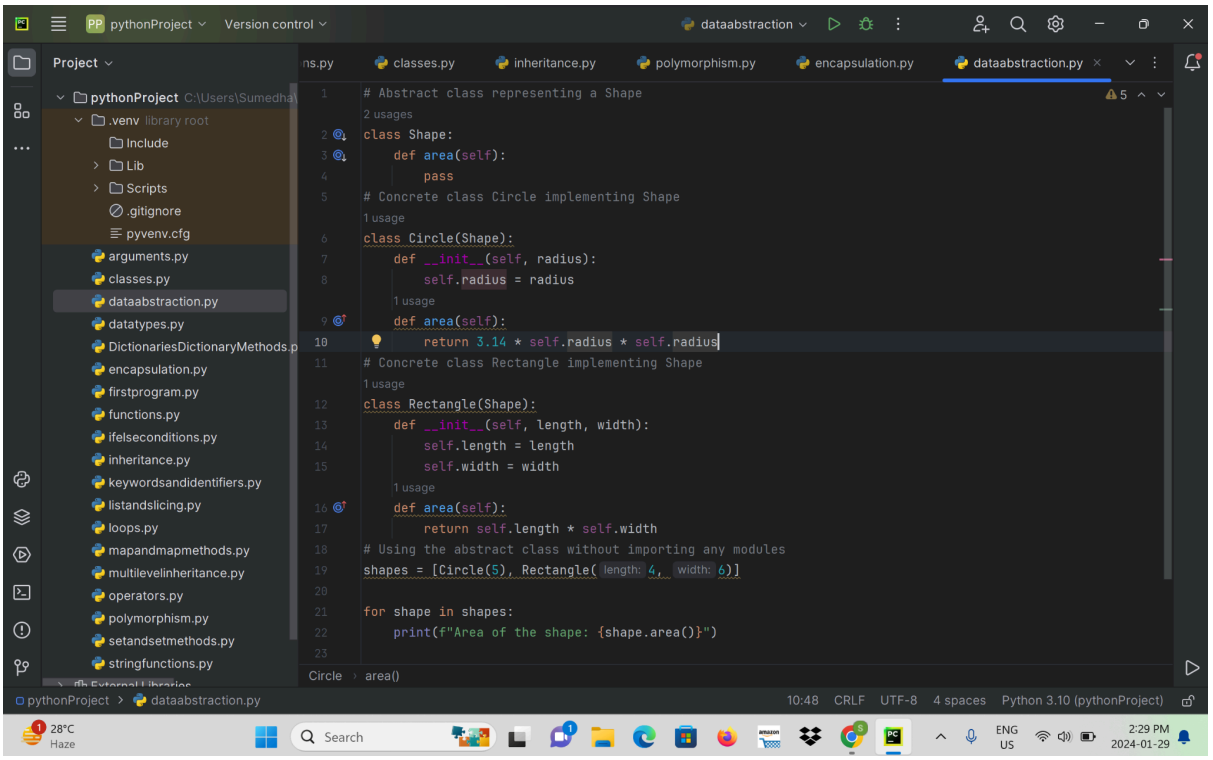
V) ENCAPSULATION: It describes the idea of wrapping data and the methods that work on data within one unit.



OUTPUT:



vi) DATA ABSTRACTION: It focuses on hiding the complex implementation details and showing only the necessary features of an object. In Python, data abstraction is often achieved through the use of abstract classes and abstract methods.



OUTPUT: