Python OOPS

* Class
* Object
* Method
* Inheritance
* Polymorphism
* Data Abstraction
* Encapsulation

Class –

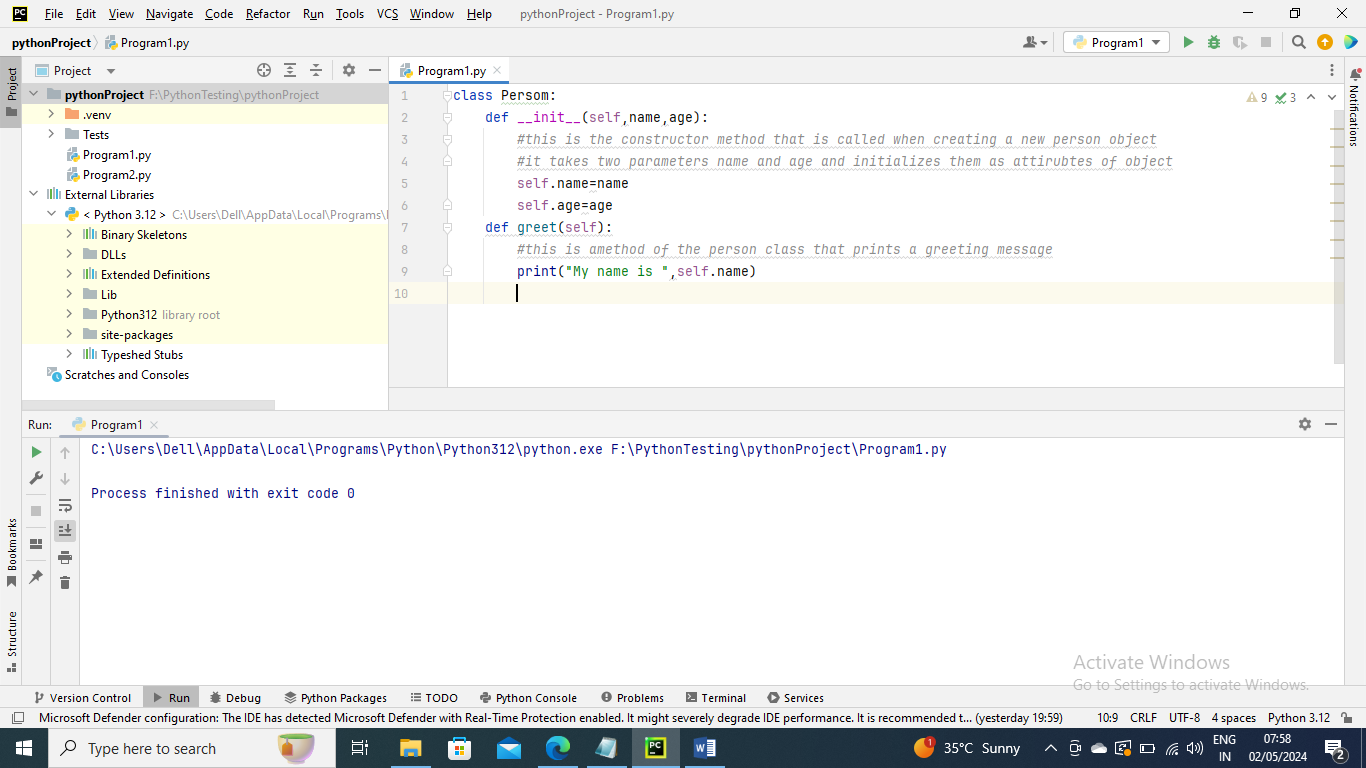
Class is a user-defined data type that contain both the data itself and the methods that may be used to manipulate it

Syntax –

Class classname:

Statements

All Methods should have self inside the class

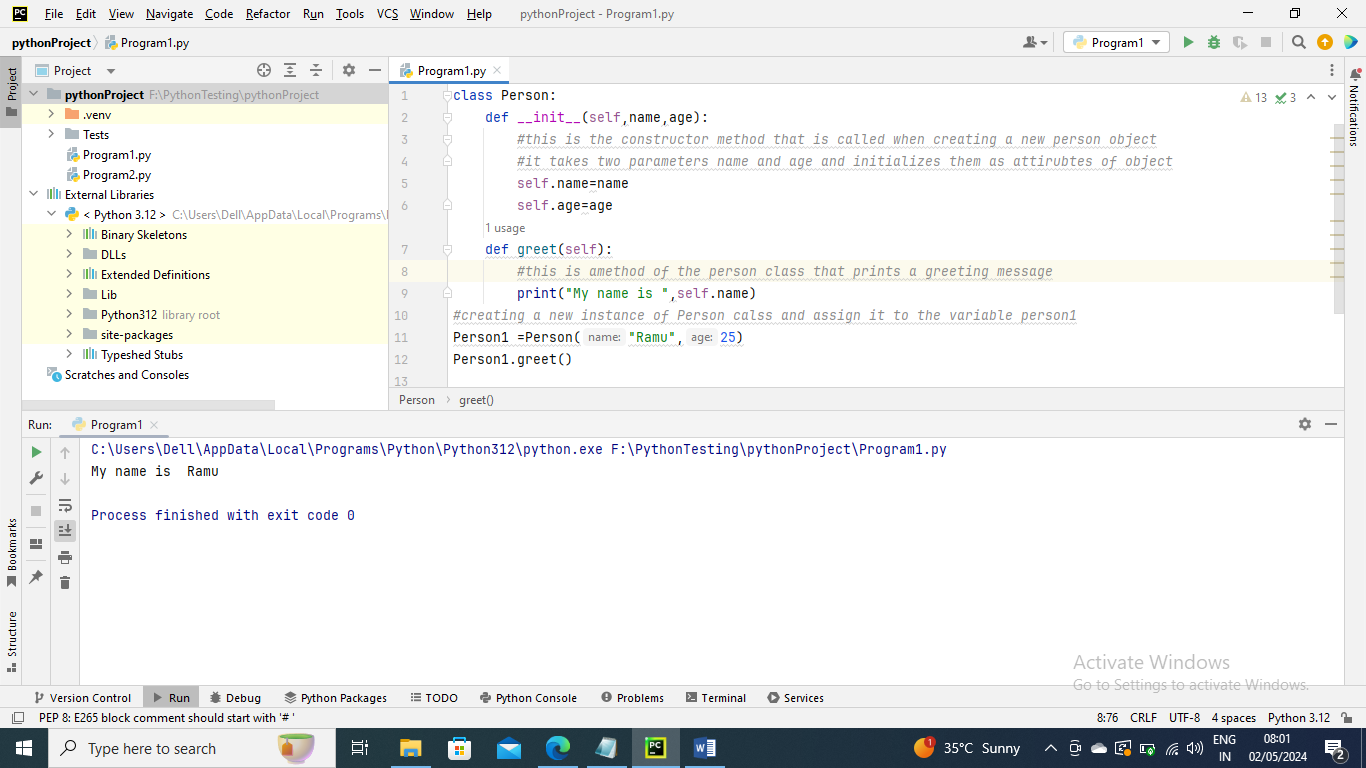


Objects –

An object is a particular instance of a class with unique characteristic and functions

Syntax –

Object\_name =Class\_name (arguments)



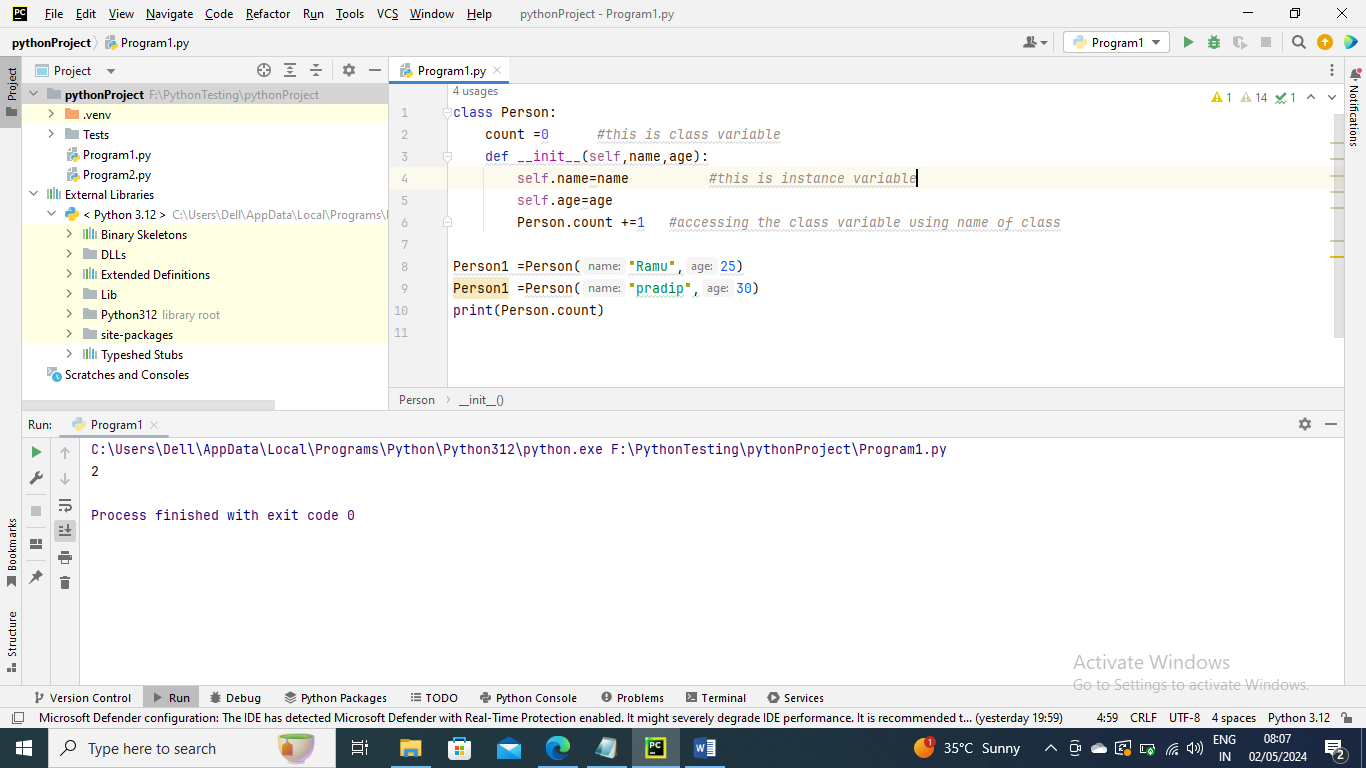
The Self parameter –

The Self parameter refers to the current instance of the class and access the class variables

\_ \_init\_ \_ method

In order to make an instance of a class in Python, a specific function called \_ \_init\_ \_ is called. Although it is used to set the objects attributes, it is often reffered to as constructor

Class and Instance Variables



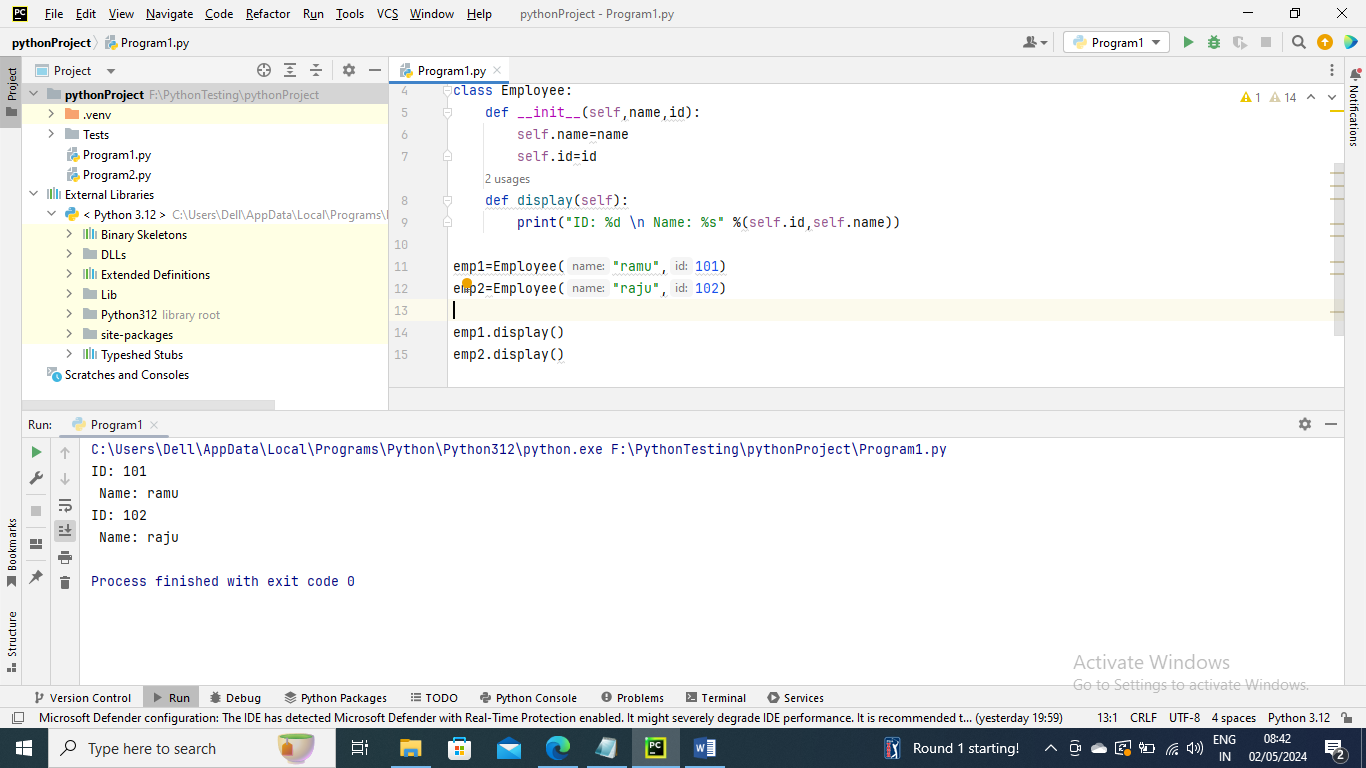
Python Constructors –

A constructor is a special type of method which is used to initialize the instance members of the class

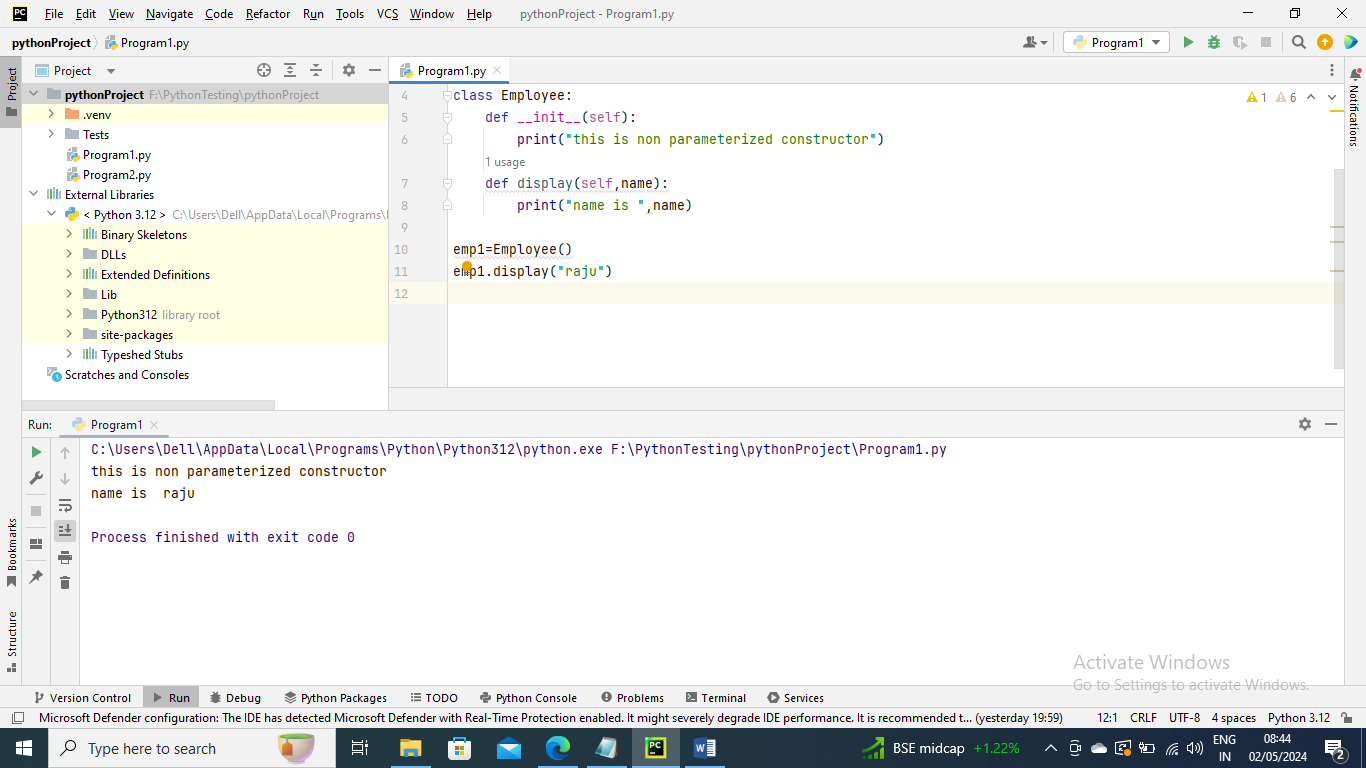
Constructor can be of two types

1. Parameterized Constructor
2. Non-parameterized Constructor

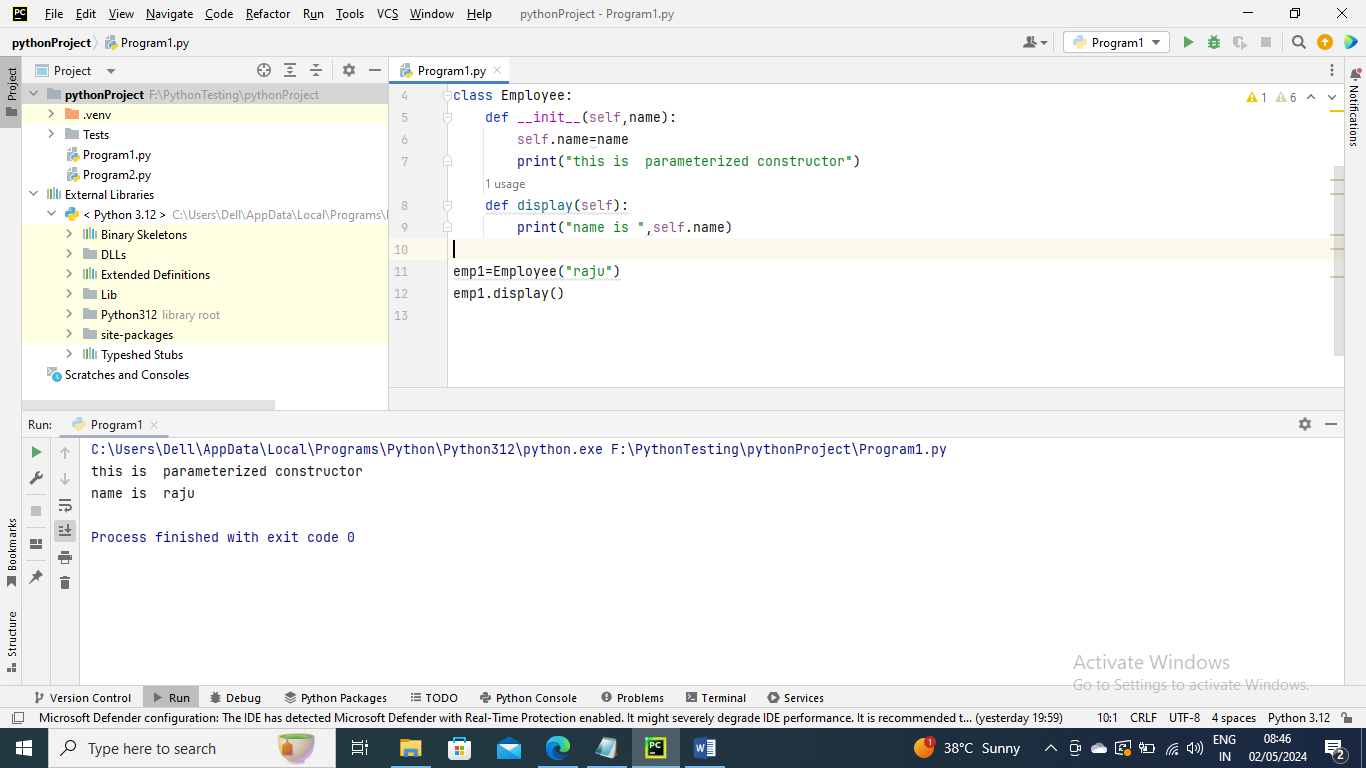
Creating constructor in Python



Non-Parameterized Constructor

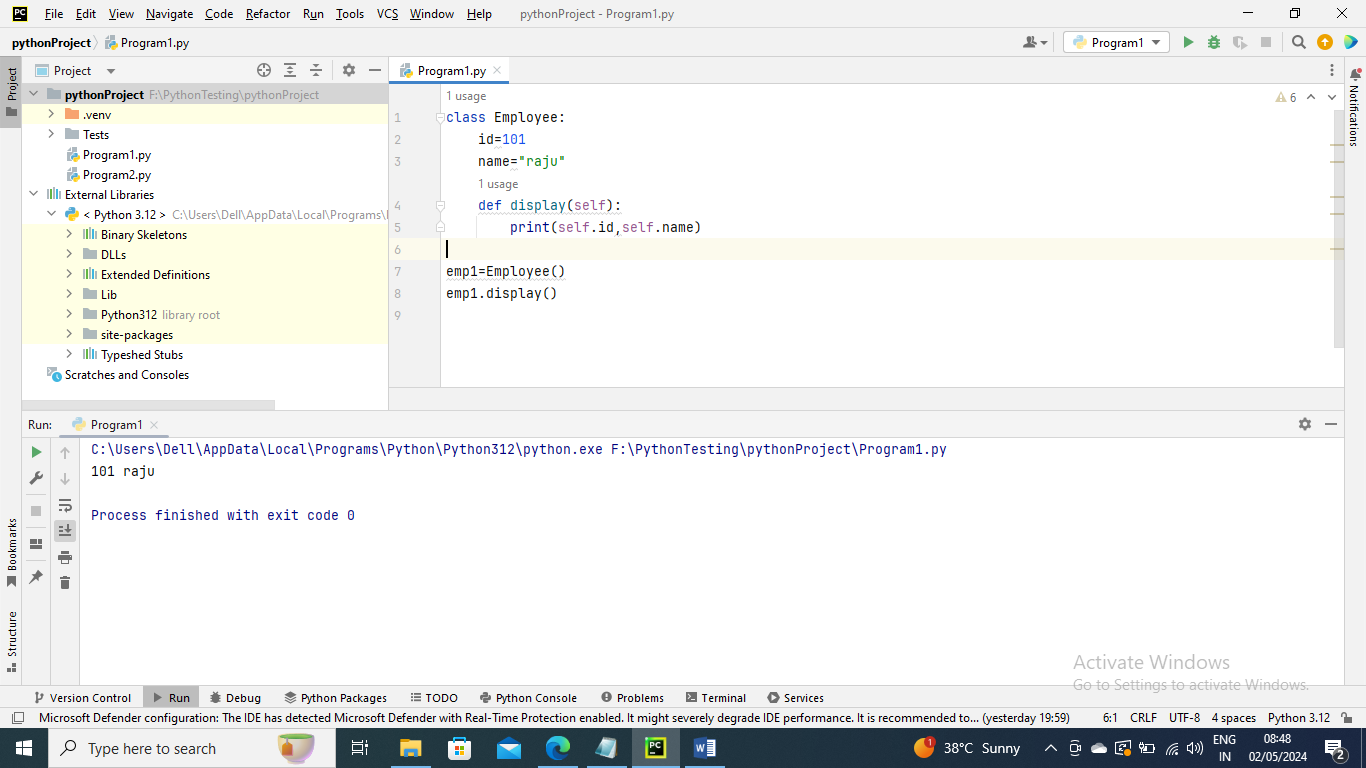


Parameterized Constructor

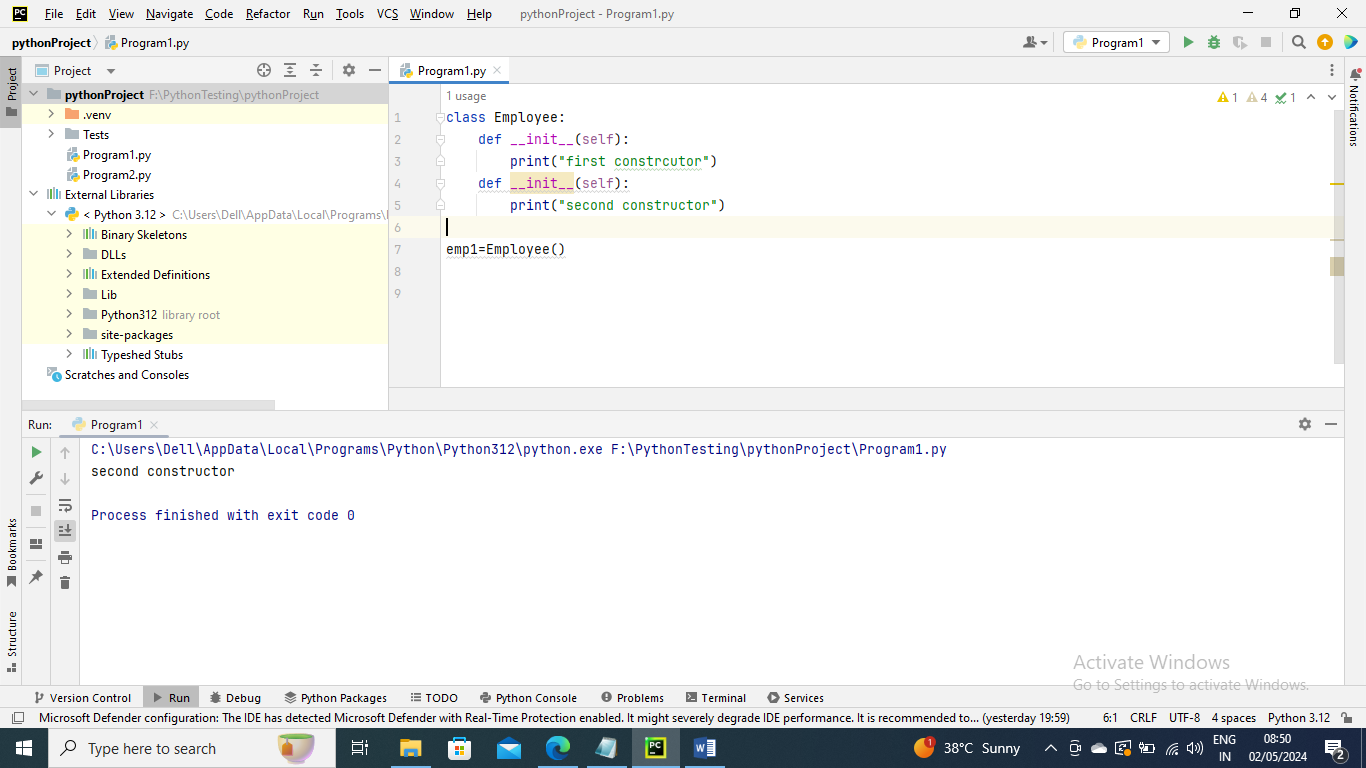


Default Coonstructor –

When we do not include the constructor in the class or forget to declare it, then that becomes the default constructor



More than One constructor in Single class



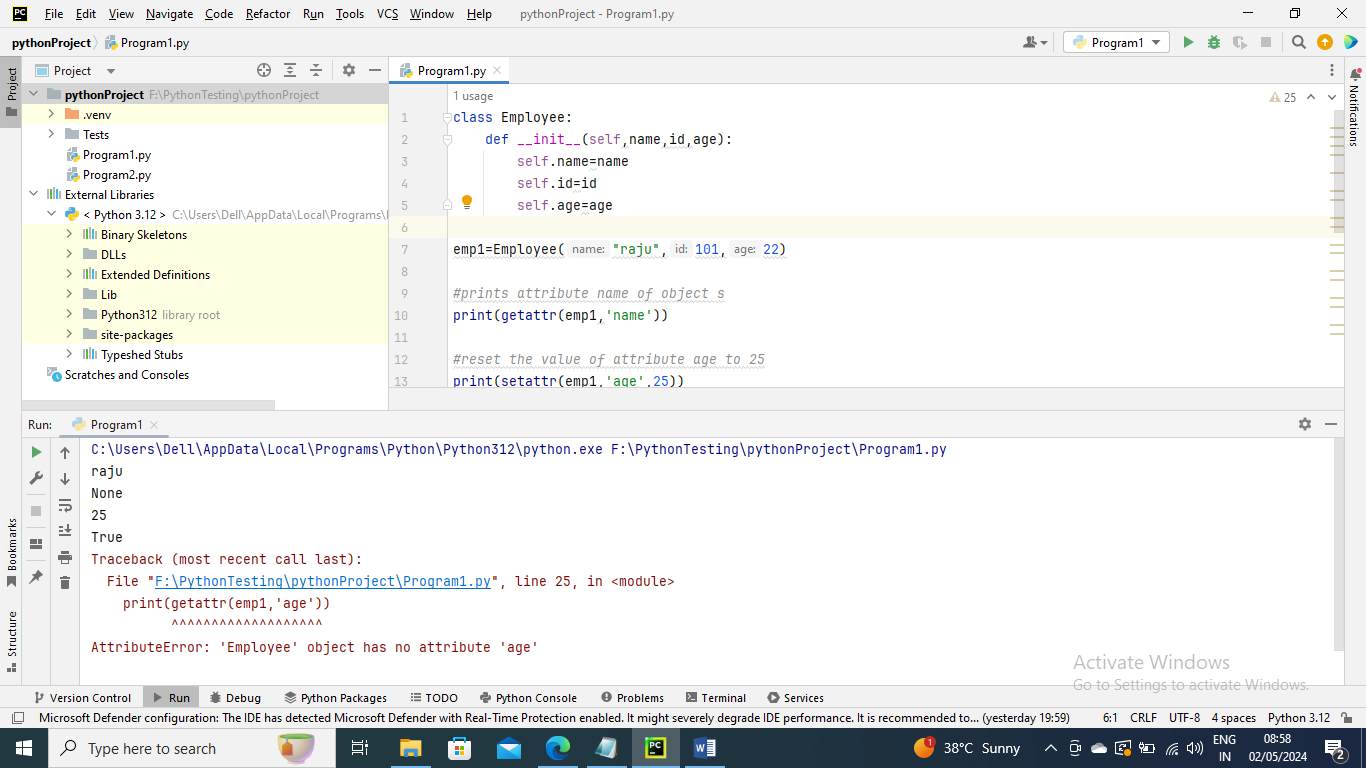
Python Built-in class functions

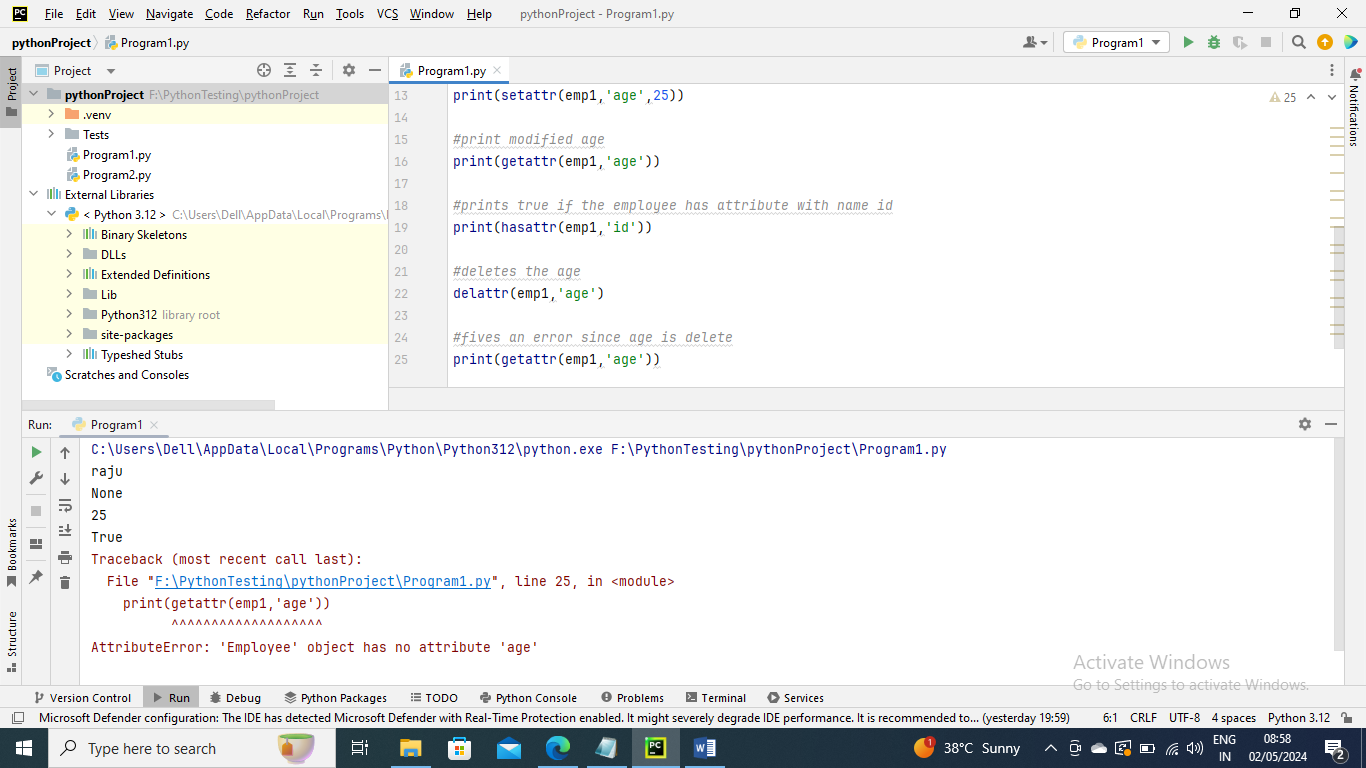
getattr(ob,name,default) – is used to access the attribute of object

setattr(obj,name,value) – is used to set a particular value to specific attribute of an object

delattr(obj,name) – used to delete a specific attribute

hasattr(obj,name) – return true if the object contains specific attribute





Python Inheritance

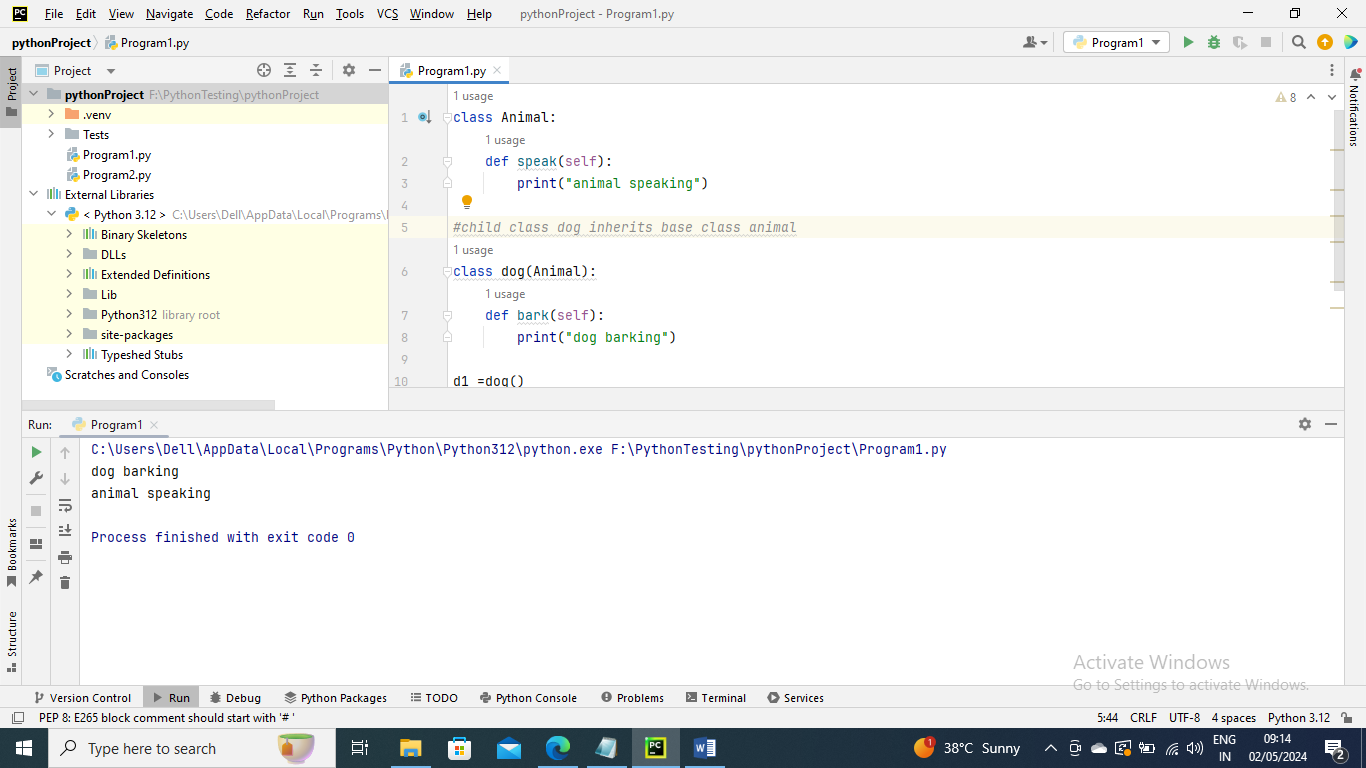
Syntax –

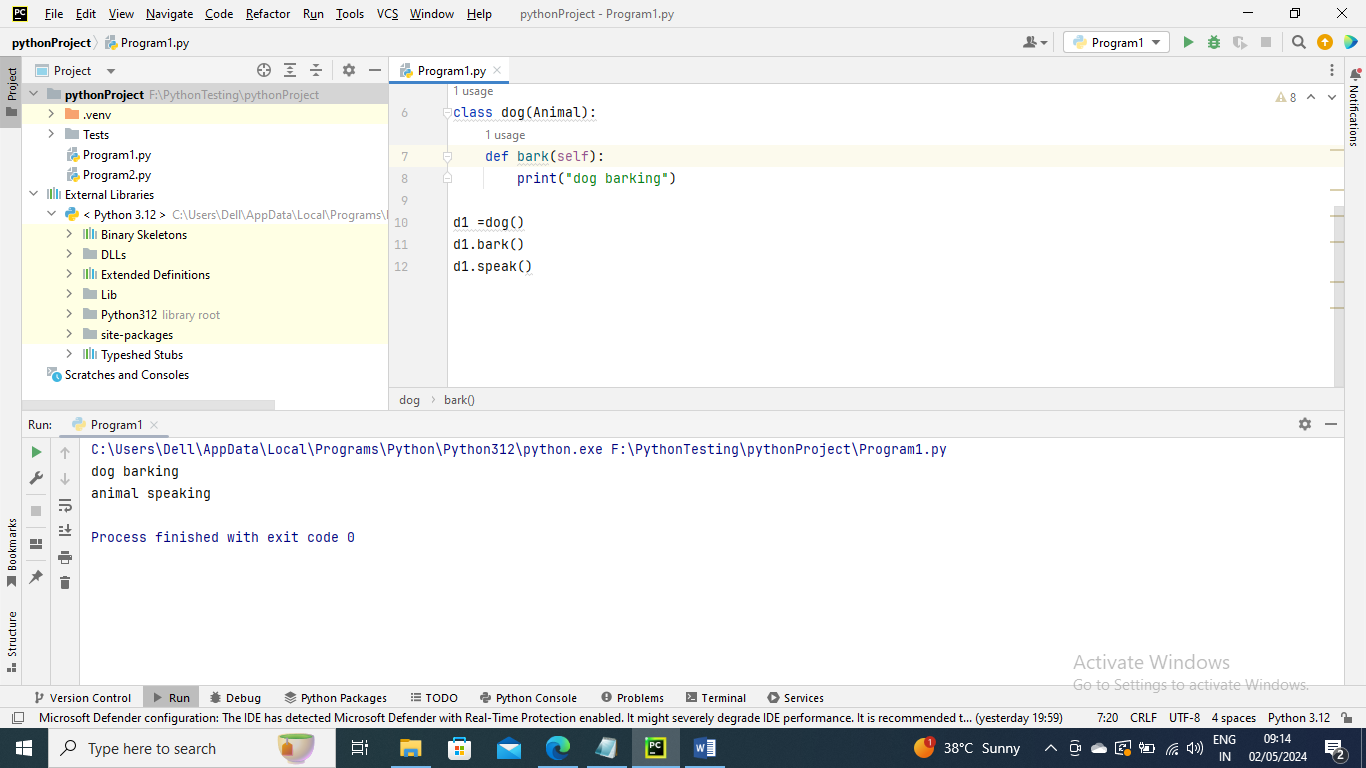
Class derived-class (base class):

Statements

Class derived-class (base class1, base class2 ,…):

Statements





Multi-level Inheritance

Synatx –

class class1:

Statements

class class 2(class1):

Statements

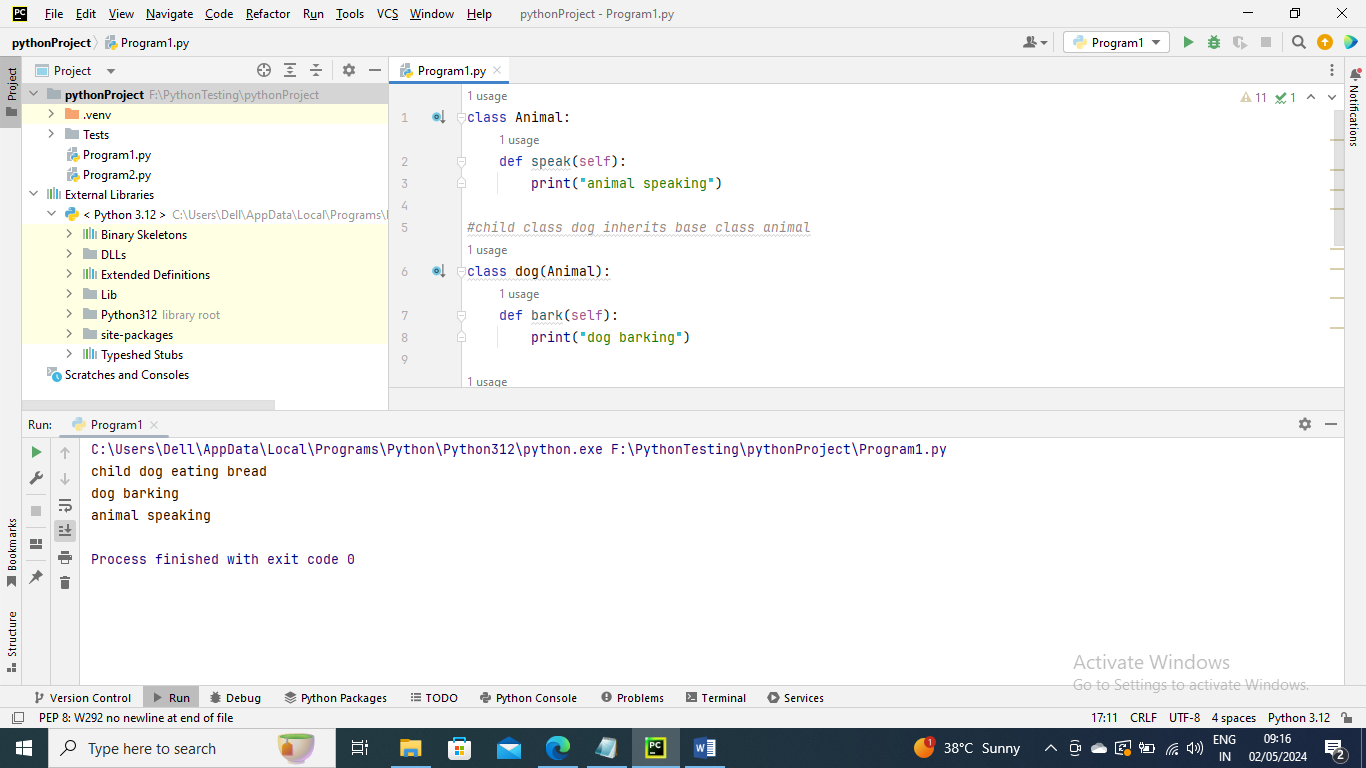
Class class3(class2):

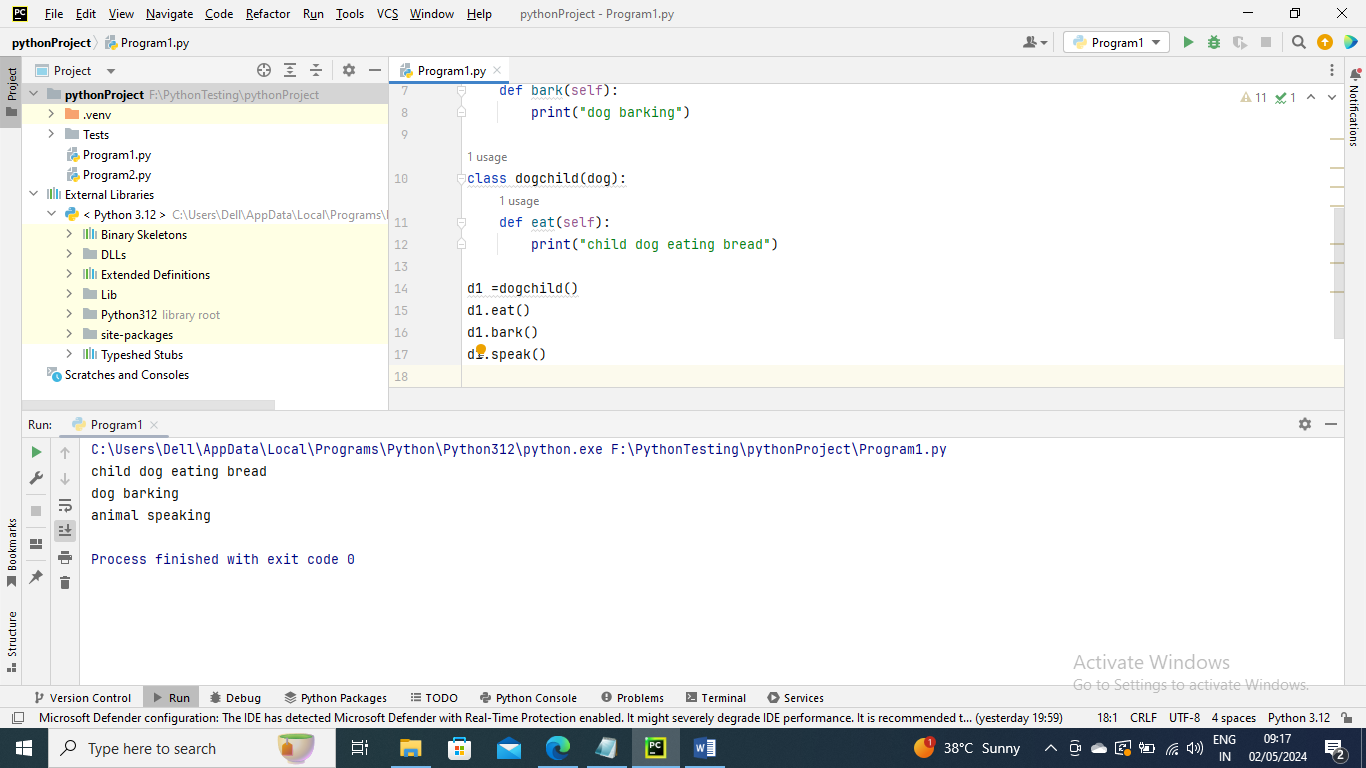
Statements

.

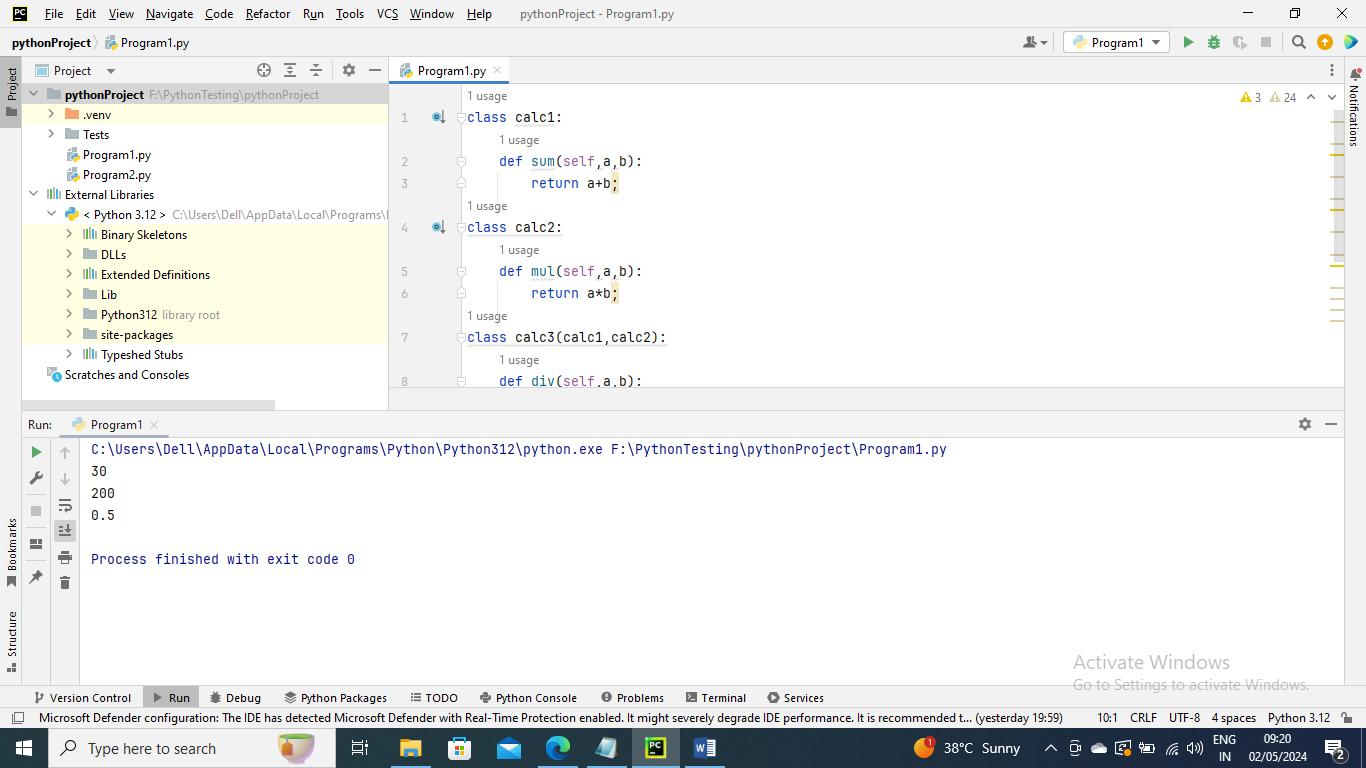
.

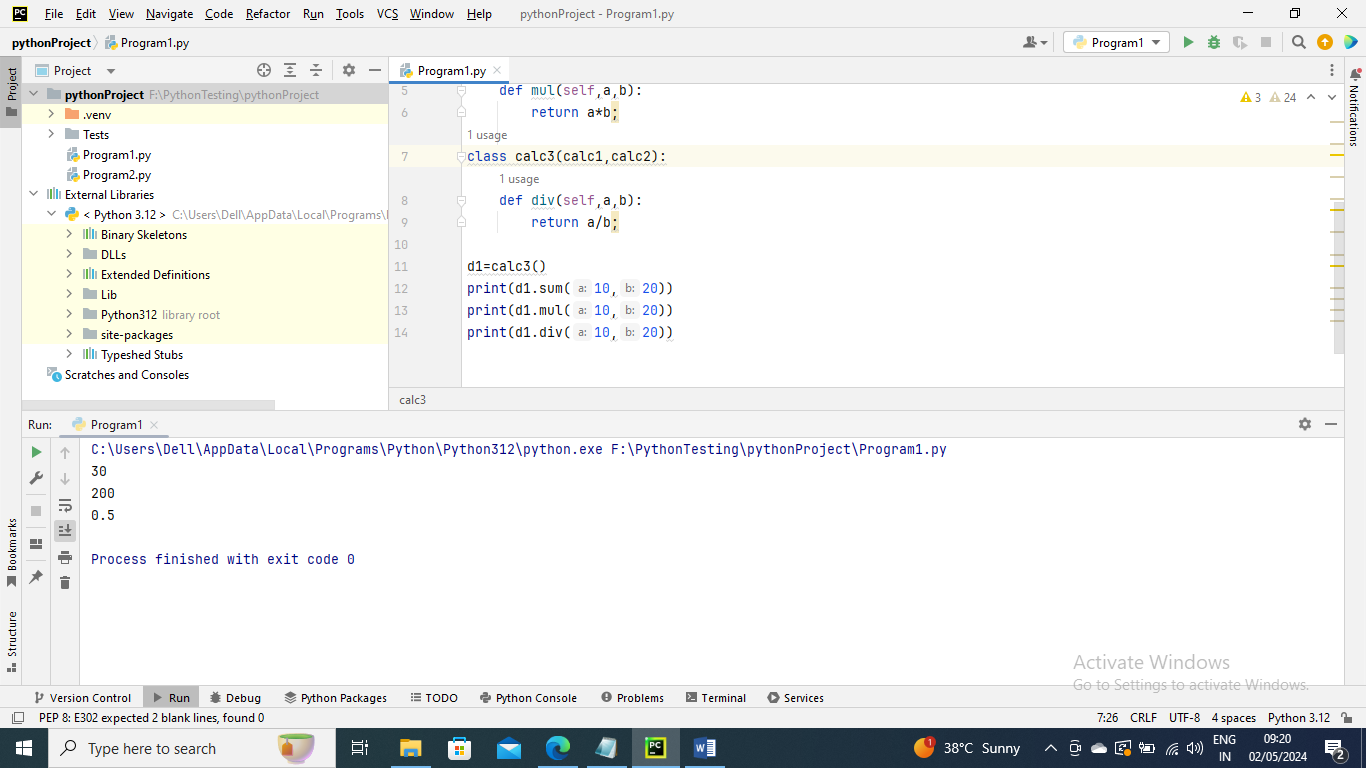
.





Mutiple Inheritance



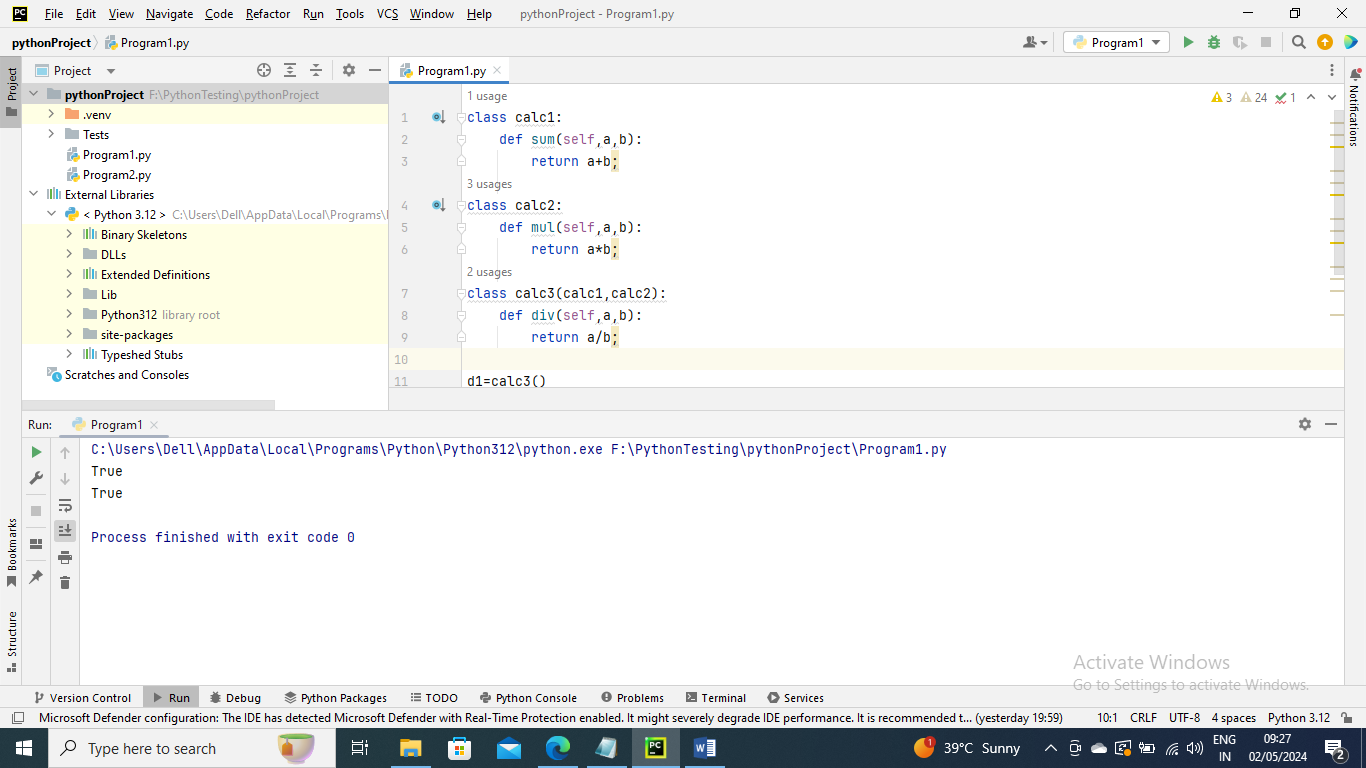


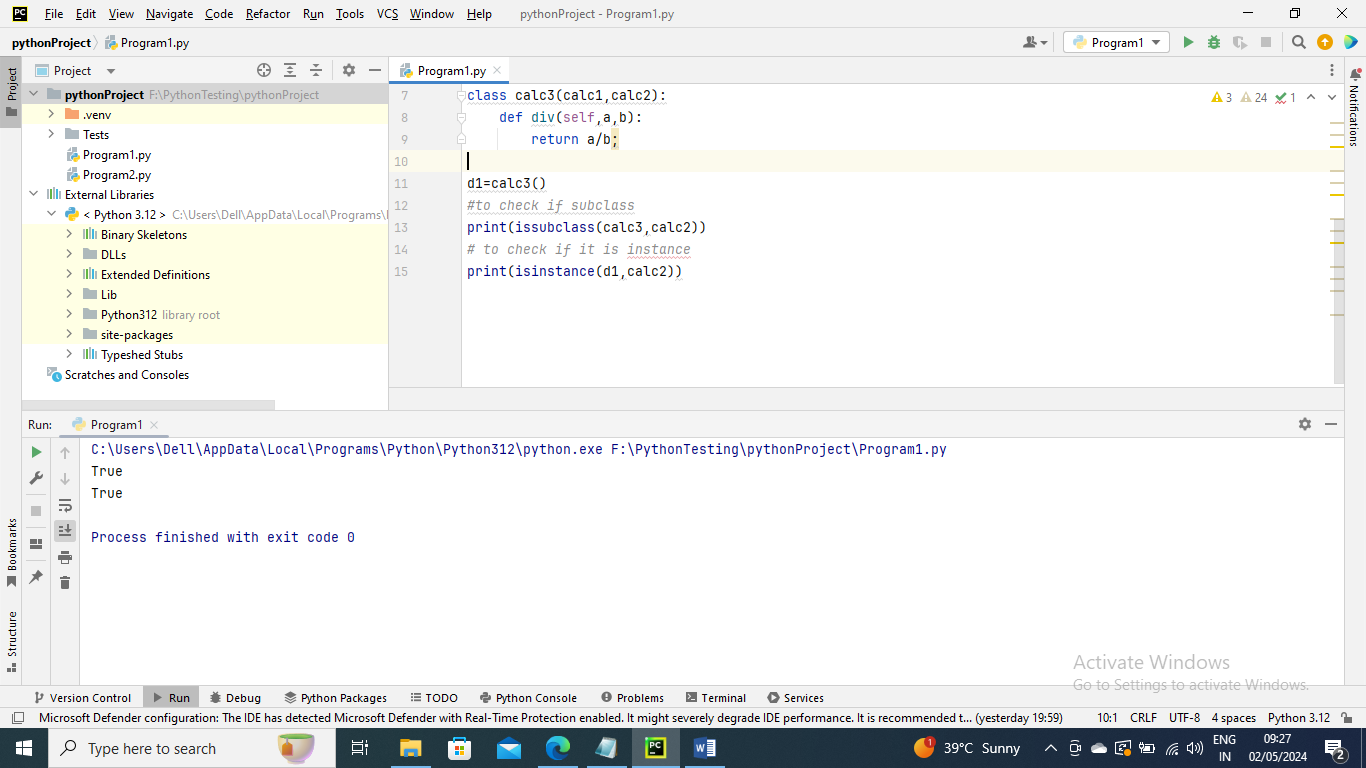
The issubclass(sub,sup) method

issubclass(sub,sup) method is used to check the relationships between the specified classes. It returns true if the firys class is subclass of the second class and false otherwise

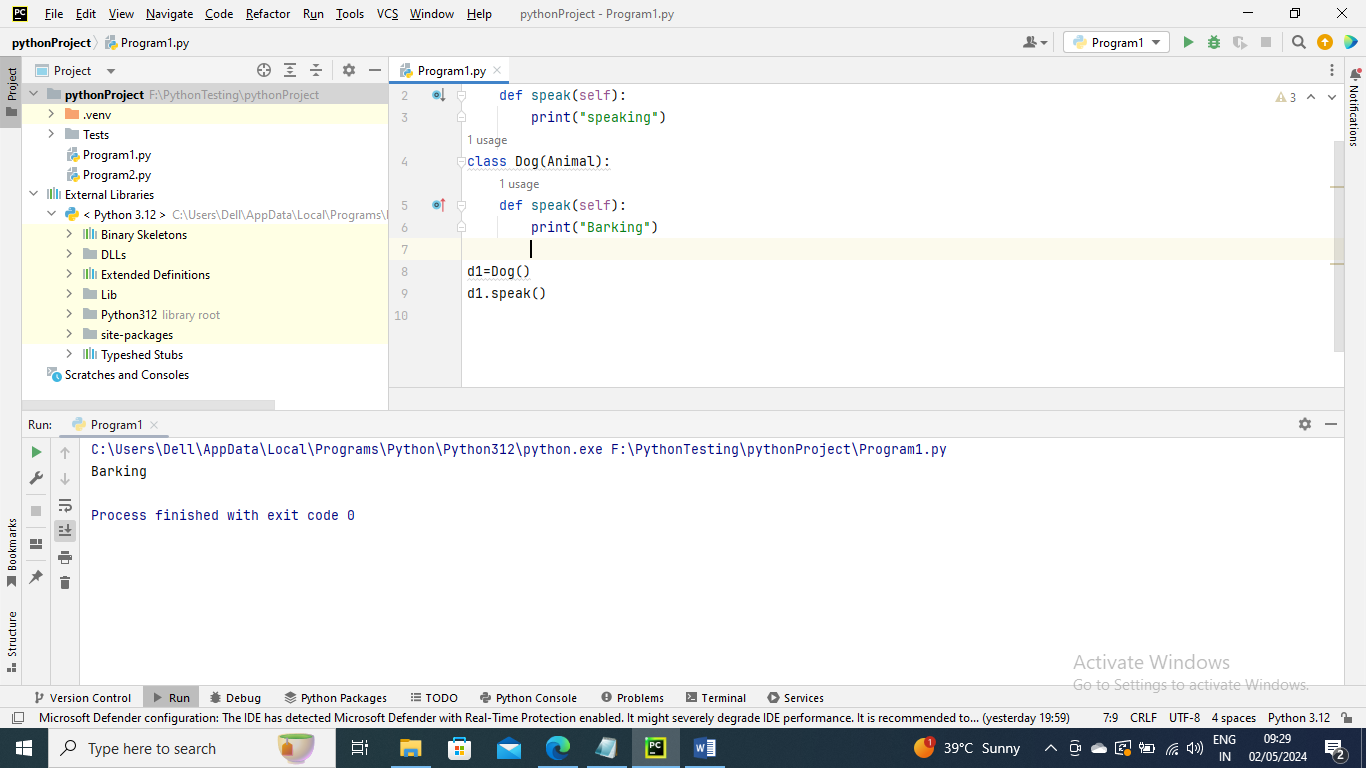
The isinstance(obj,class) method

isinstance(obj,class) method is used to check the relationship between the objects and classes. It return true if the first parameter i.e. obj is the instance of the second parameter





Method Overriding



Data Abstraction –

Abstraction is used to hide internal functionality of the function from the users

Abstraction can be achieved by using abstract classes and interfaces

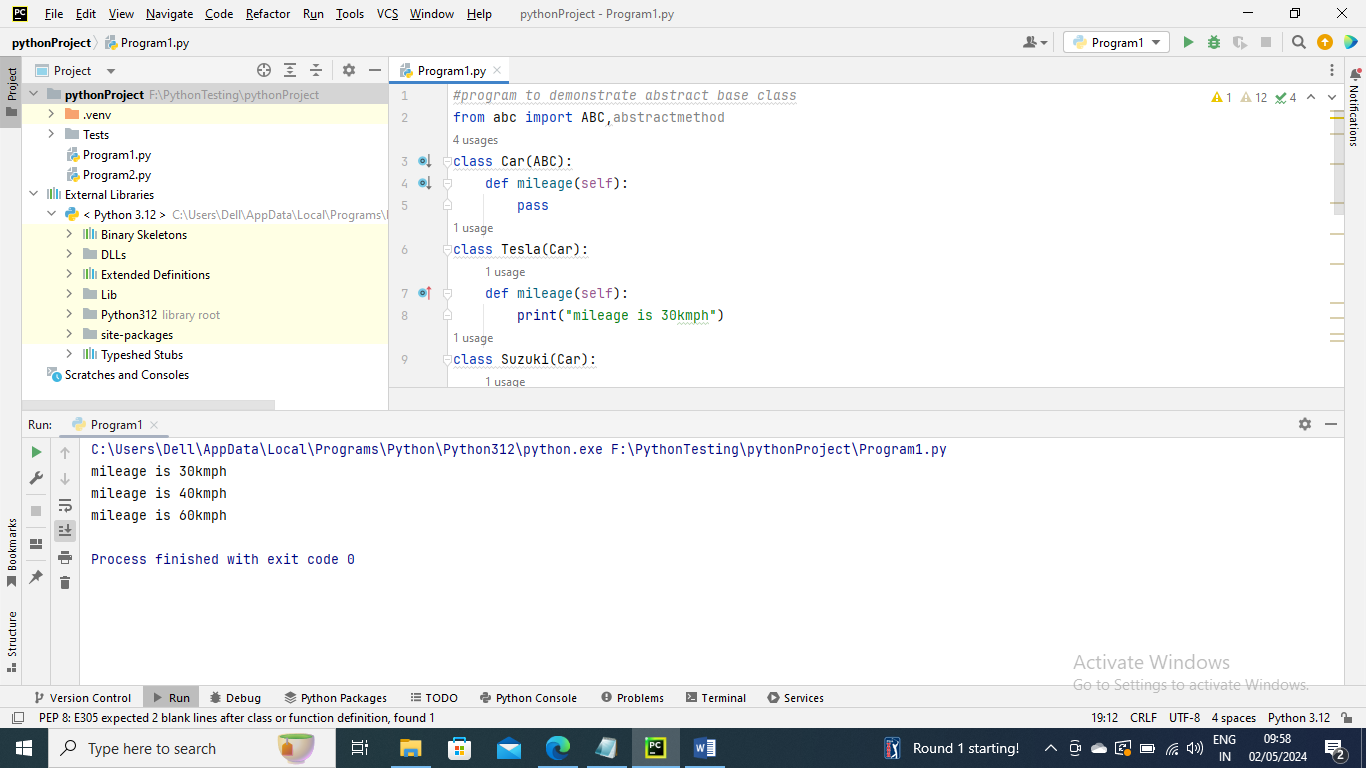
Syntax –

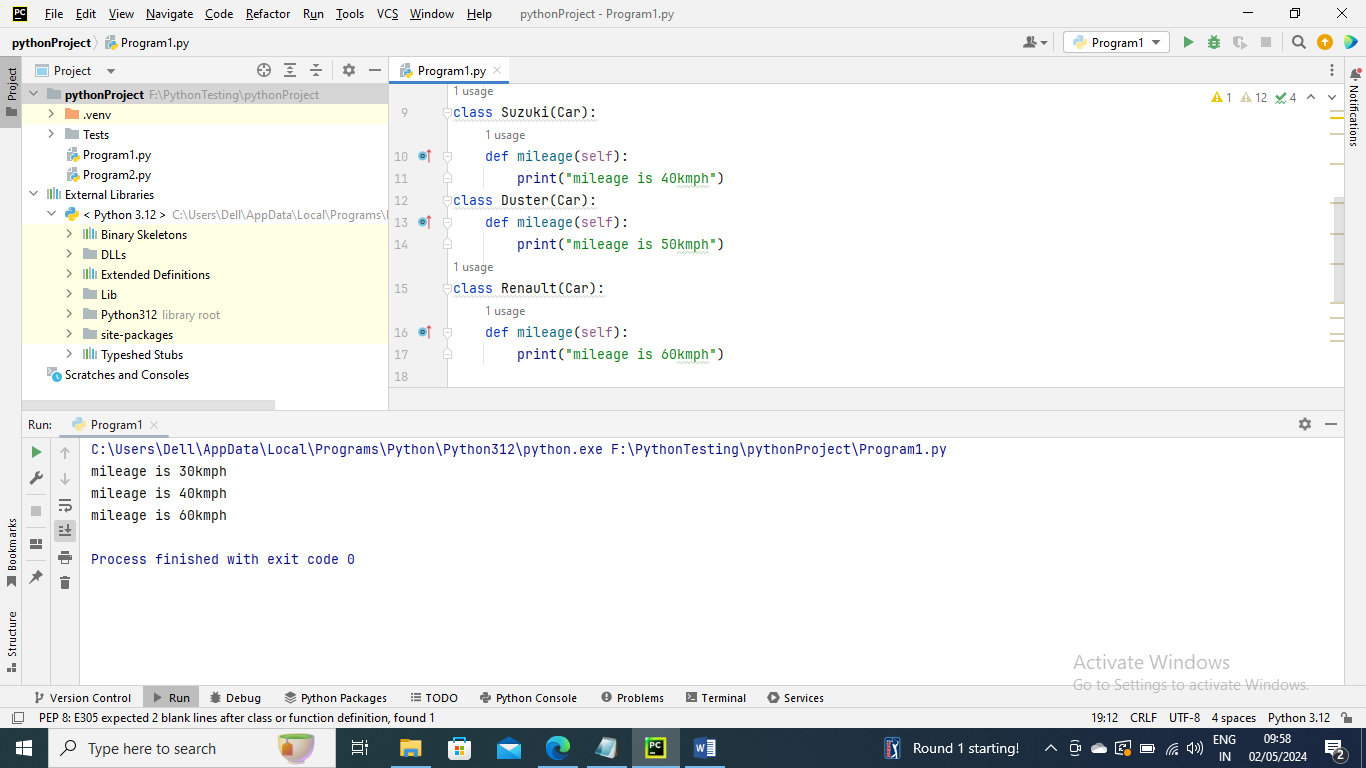
From abc import ABC

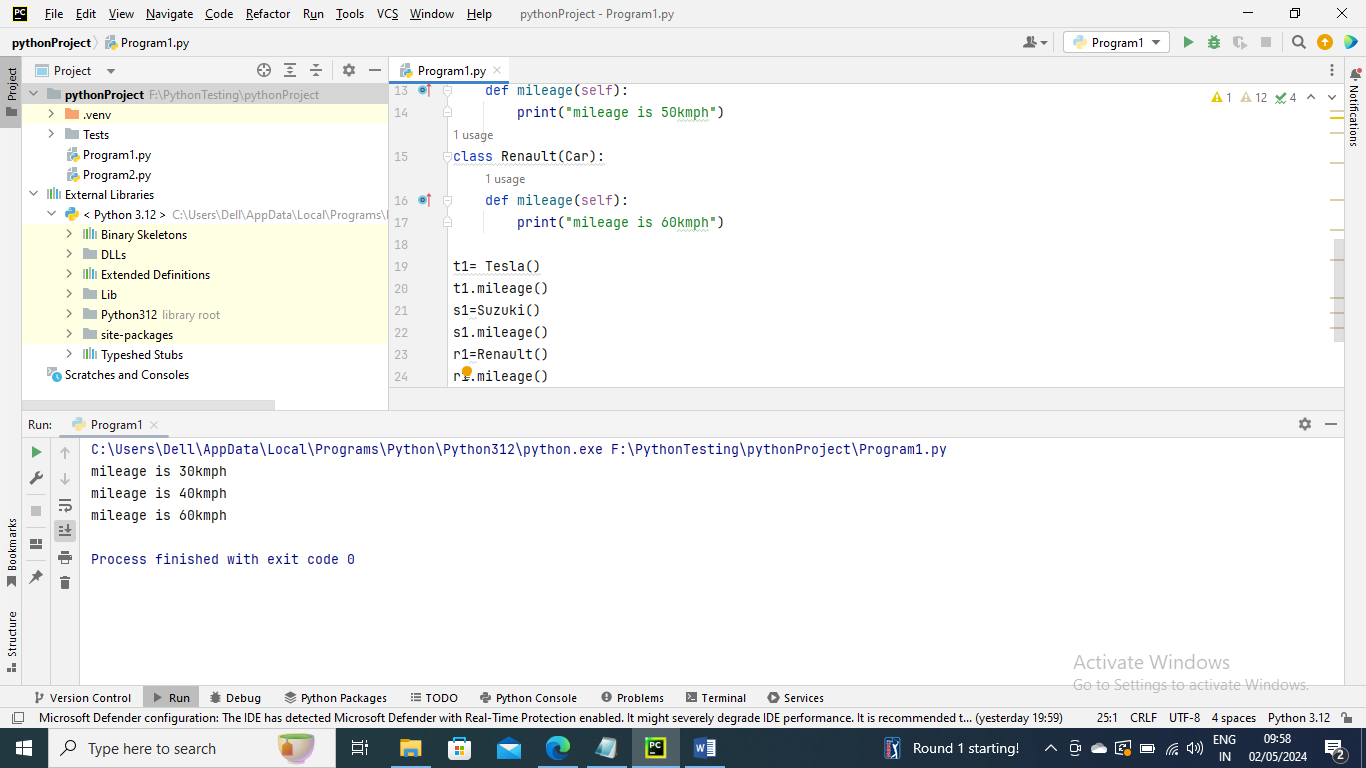
Class classname(ABC):

Abstract class –

We use @abstractmethod to define abstract method or if we don’t procide the definition of method it automatically becomes the abstract method







In the above program we have imported ABC module to create the abstract base class. We created the car class that inherited ABC class and defined abstract method named mileage(). We have then inherited the base class from three different subclasses and implemented the abstract method differently. We created the objects to call the abstract method

We can perform data hiding by adding double underscore \_ \_ as a prefix to the attribute which is to be hidden. After this, the attribute will not be visible outside of the class through object

