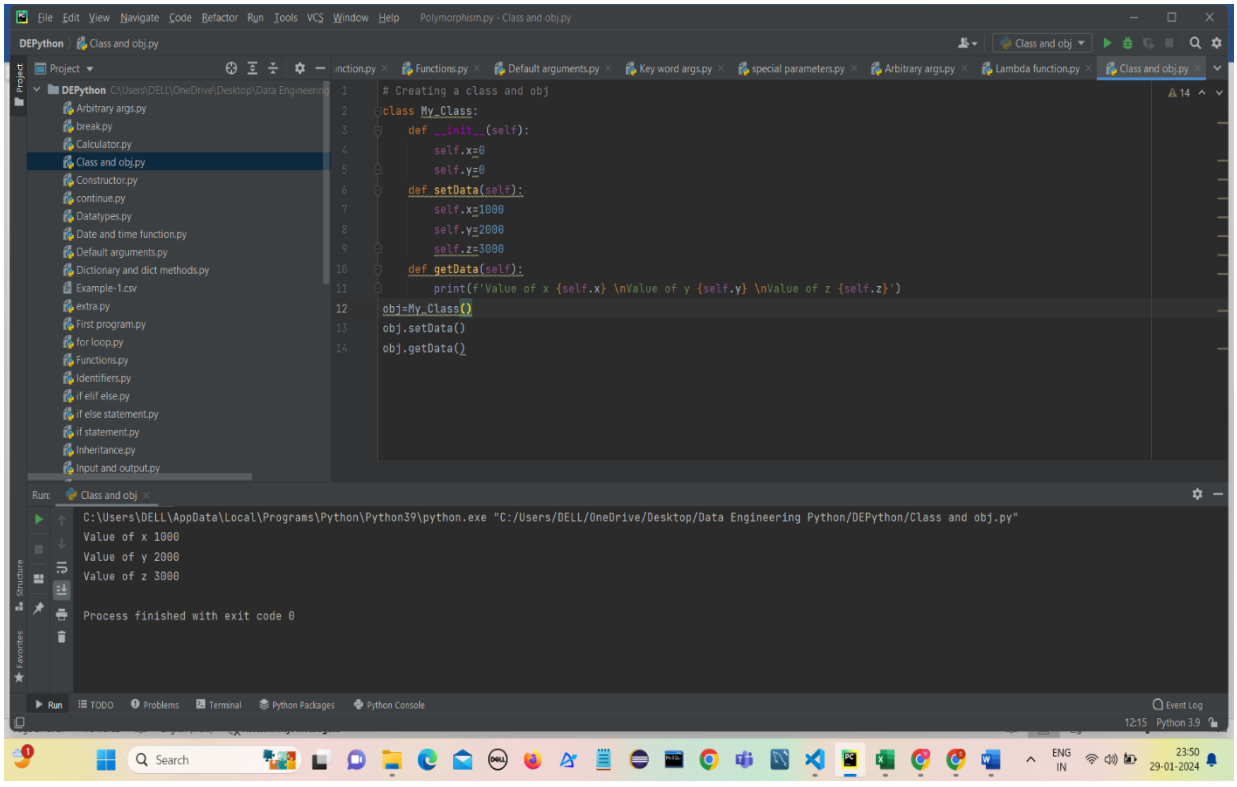
OOPS-Assignment---2

1) Class and Object

Class:

- A class is a user-defined blueprint or prototype from which objects are created.
- Classes provide a means of bundling data and functionality together.
- Creating a new class creates a new type of object, allowing new instances of that type to be made.
- Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by their class) for modifying their state.



Object:

- An Object is an instance of a Class.
- An object consists of:
    1. **State:** It is represented by the attributes of an object. It also reflects the properties of an object.
    2. **Behavior:** It is represented by the methods of an object. It also reflects the response of an object to other objects.
    3. **Identity:** It gives a unique name to an object and enables one object to interact with other objects.

2) Constructor

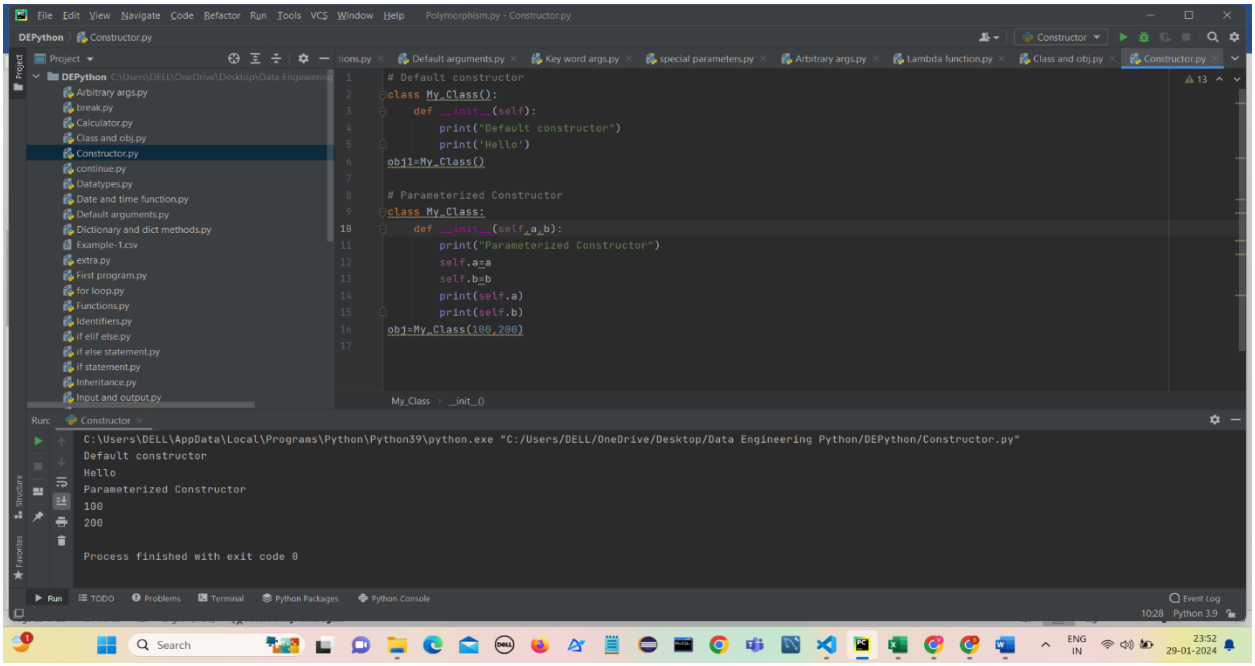
- __init__ method is a constructor in python
- It is a special member of class.
- It is used to initialize data member of class.
- It is automatically called when an object is created.
- It contains instructions that are executed at the time of object creation.
- There are two types of constructors.
  Default constructor
  Parameterized constructor

i) Default Constructor:

Default Constructor definition has only one argument which is reference to instance being constructed.

ii) Parameterized Constructor:

It is a constructor with parameters. The first argument is reference to the instance being constructed known as self and the remaining are provided by programmer.



3) Access specifiers

- Access specifiers are used to restrict the access of the class member variable and methods from outside the class.
- Encapsulation is an OOPs principle which protects the internal data of the class using Access modifiers like Public, Private and Protected.

- Python supports three types of access modifiers which are public, private and protected.
- These access modifiers provide restrictions on the access of member variables and methods of the class from any object outside the class.

i) Public:

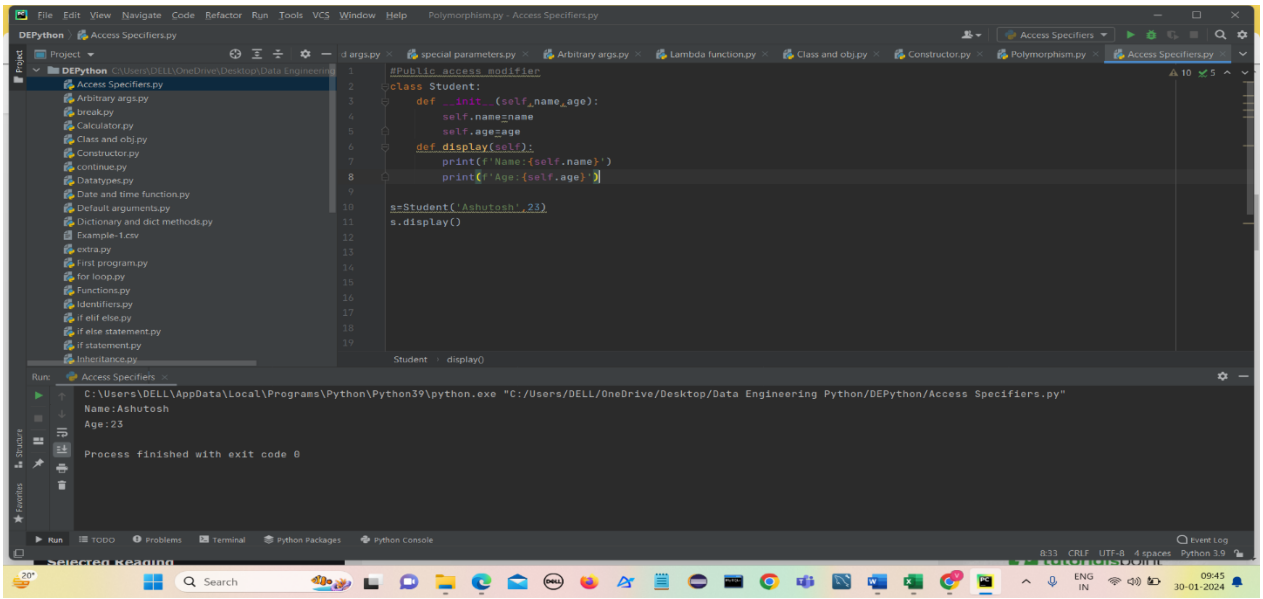
By default the member variables and methods are public which means they can be accessed from anywhere outside or inside the class.

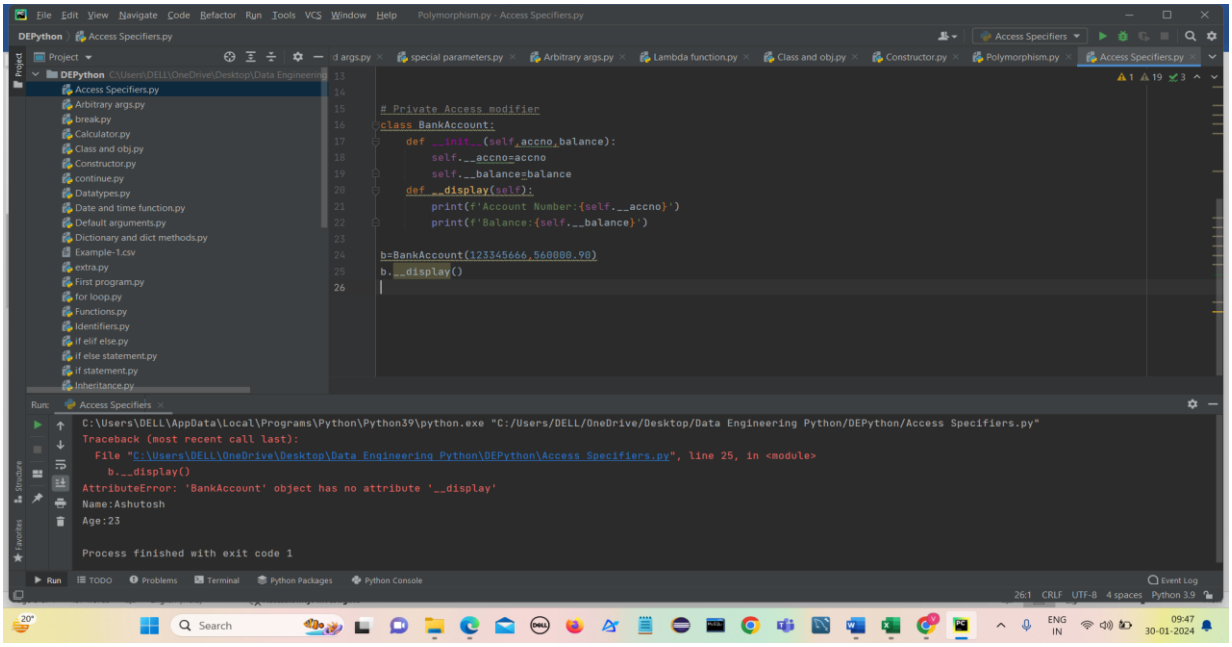
ii) Private:

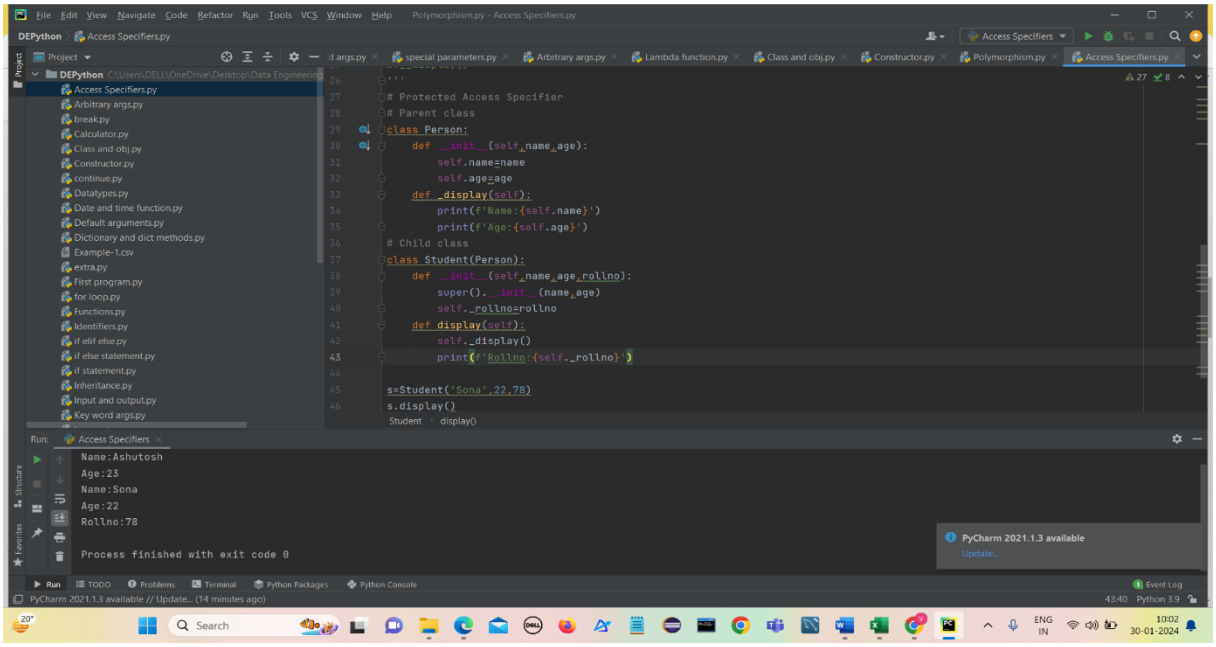
- Class properties and methods with private access modifier can only be accessed within the class where they are defined and cannot be accessed outside the class.
- In Python private properties and methods are declared by adding a prefix with two underscores(‘__’) before their declaration.

iii) Protected:

- Class properties and methods with protected access modifier can be accessed within the class and from the class that inherits the protected class.
- Protected members and methods are declared using single underscore(‘_’) as prefix before their names.

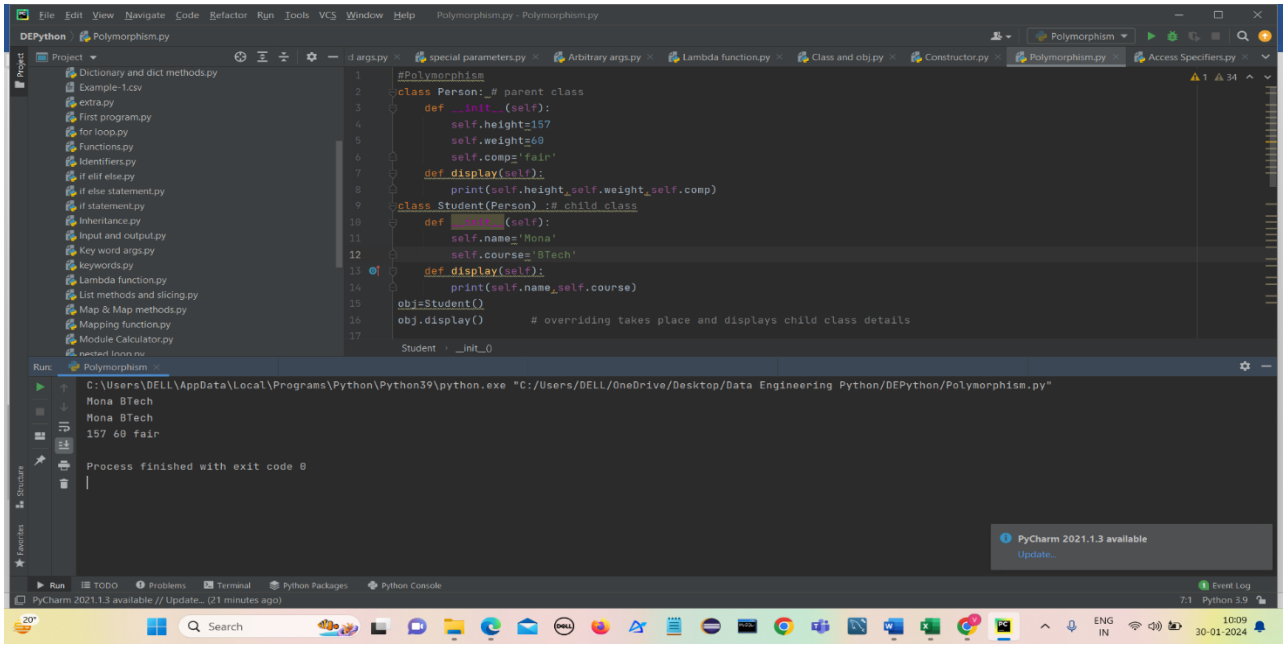


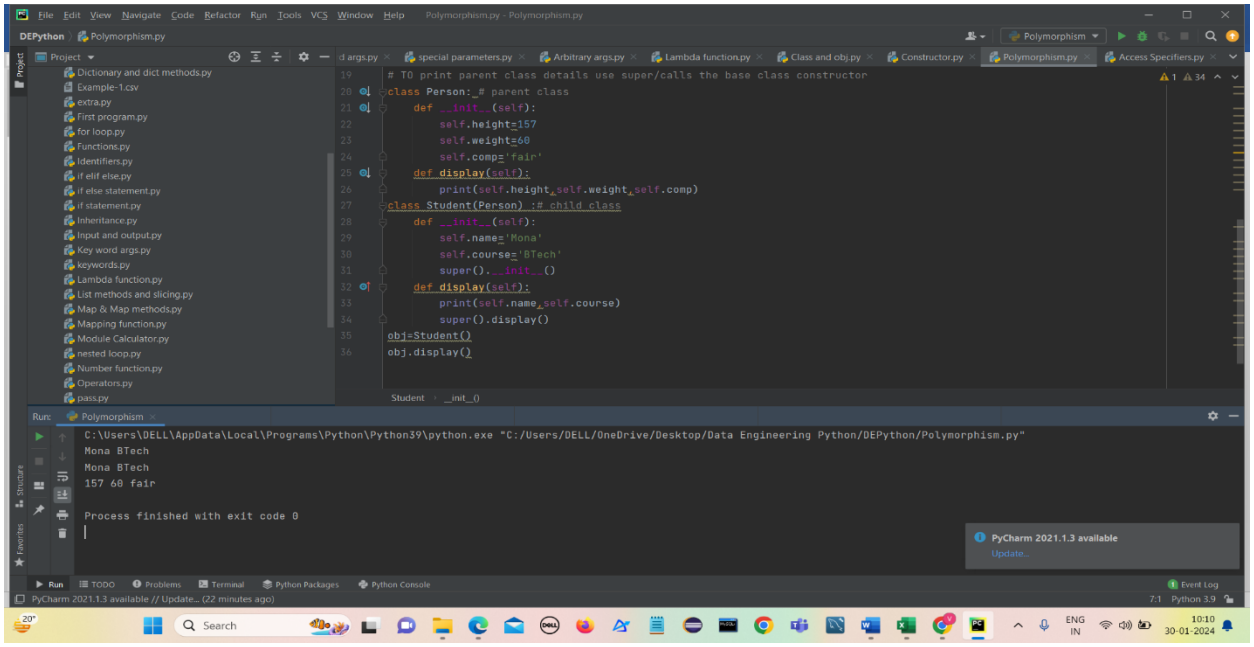




4) Polymorphism

- Polymorphism refers to having multiple forms.
- Polymorphism is a programming term that refers to the use of the same function name, but with different signatures, for multiple types.



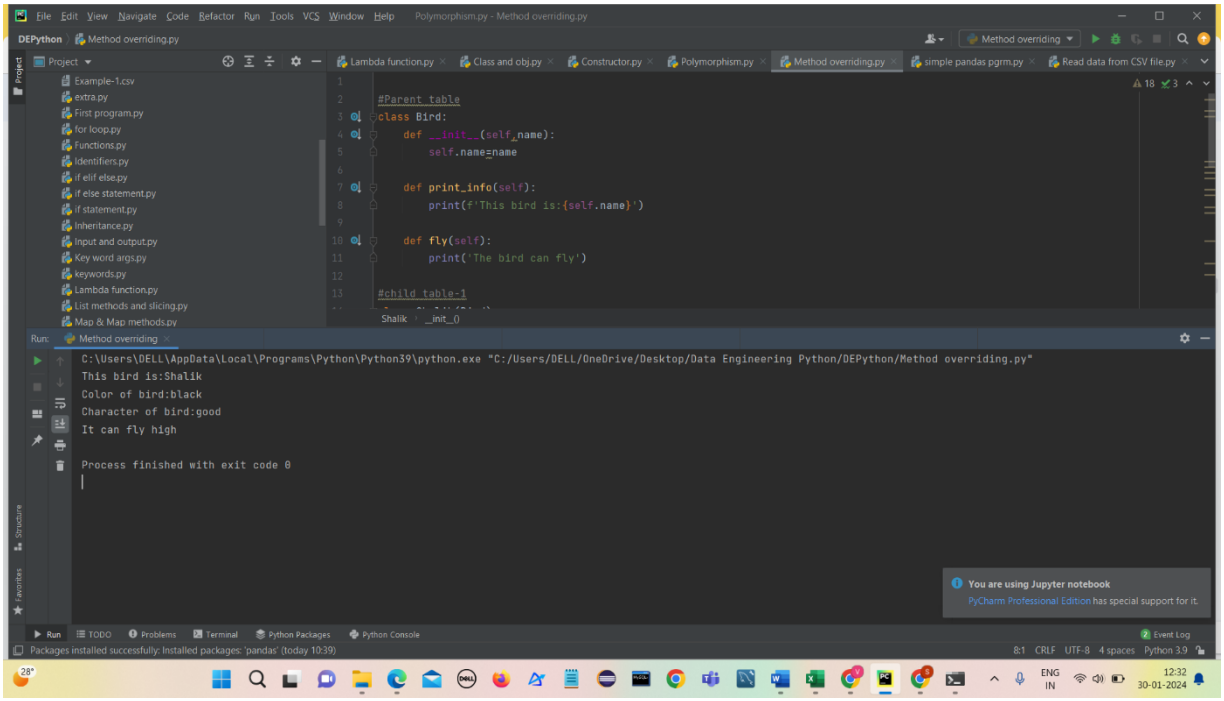


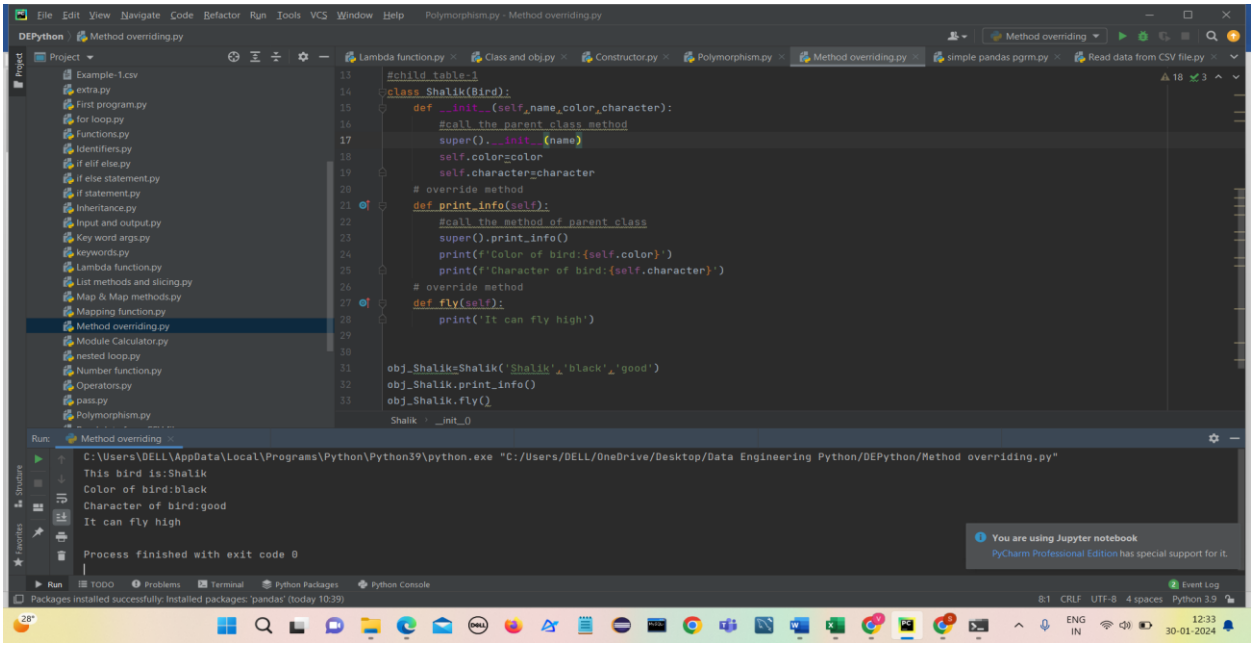
5) Method Overriding

For implementing it you have to use inheritance.

In method overriding you can reimplement the methods of parent class inside child class wherein the methods of child class and parent class have the similarity in the following features:

same name, parameters or signatures and same return type.





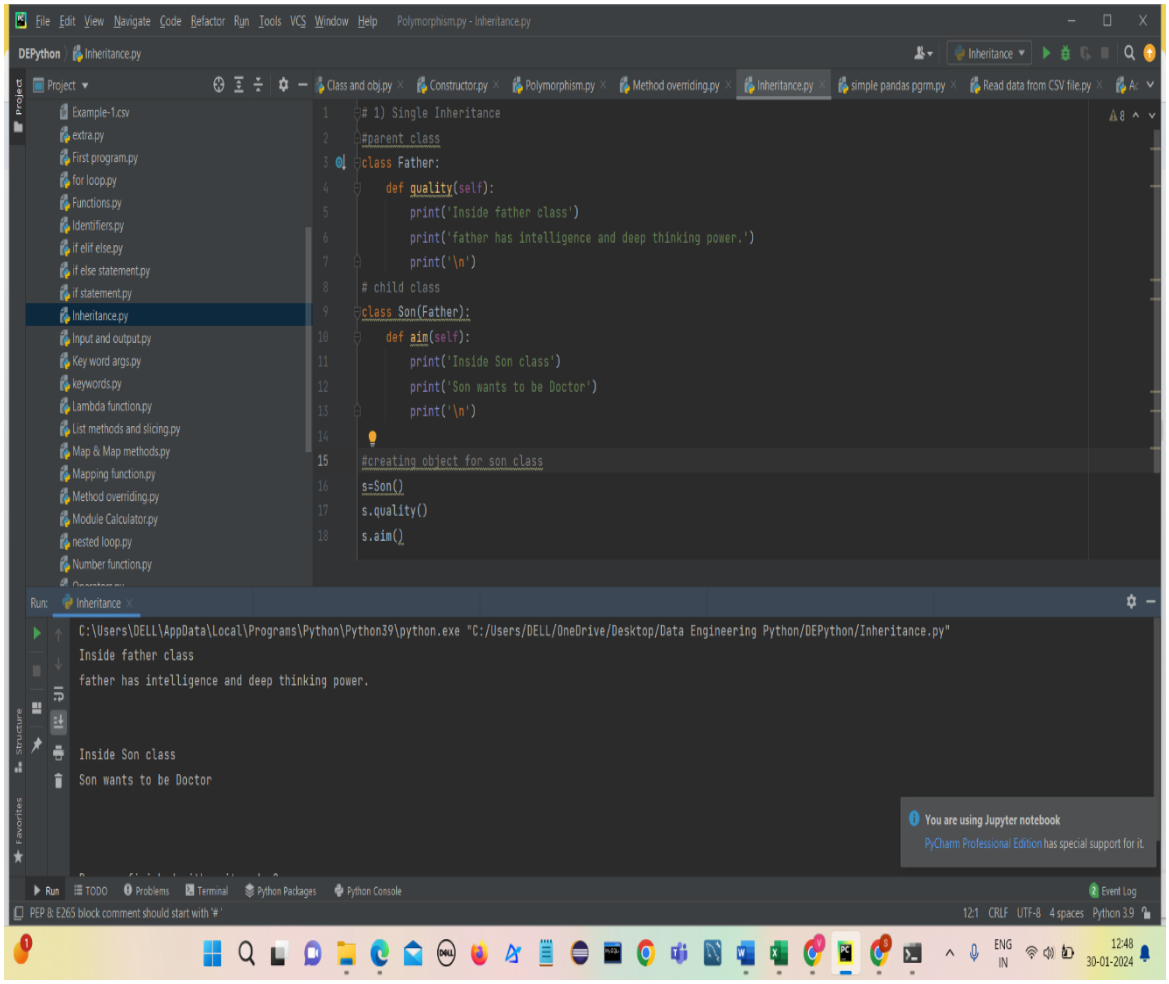
6) Inheritance

- Inheritance is the capability of one class to derive or inherit the properties from another class.
- The class that derives properties is called the **derived class or child class** and the class from which the properties are being derived is called the **base class or parent class.**

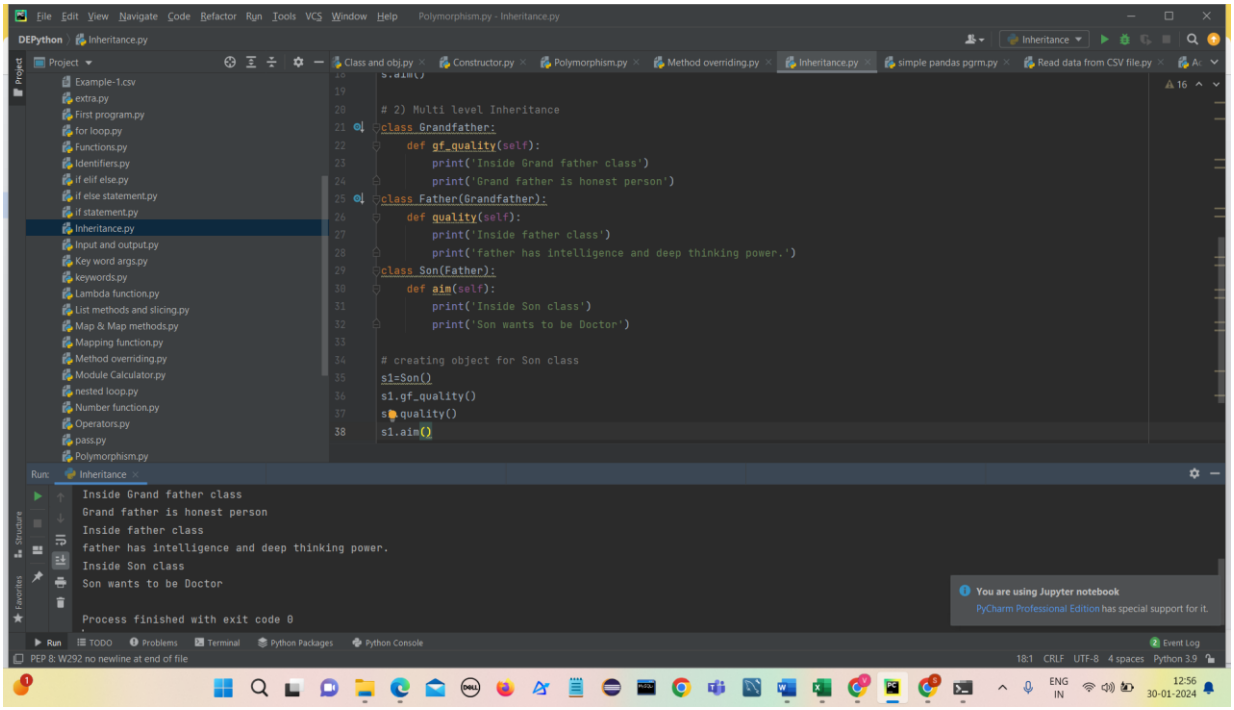
Types of Inheritance

- **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.
- **Hybrid Inheritance:** It is a combination of one or more inheritances.

1) Single Inheritance



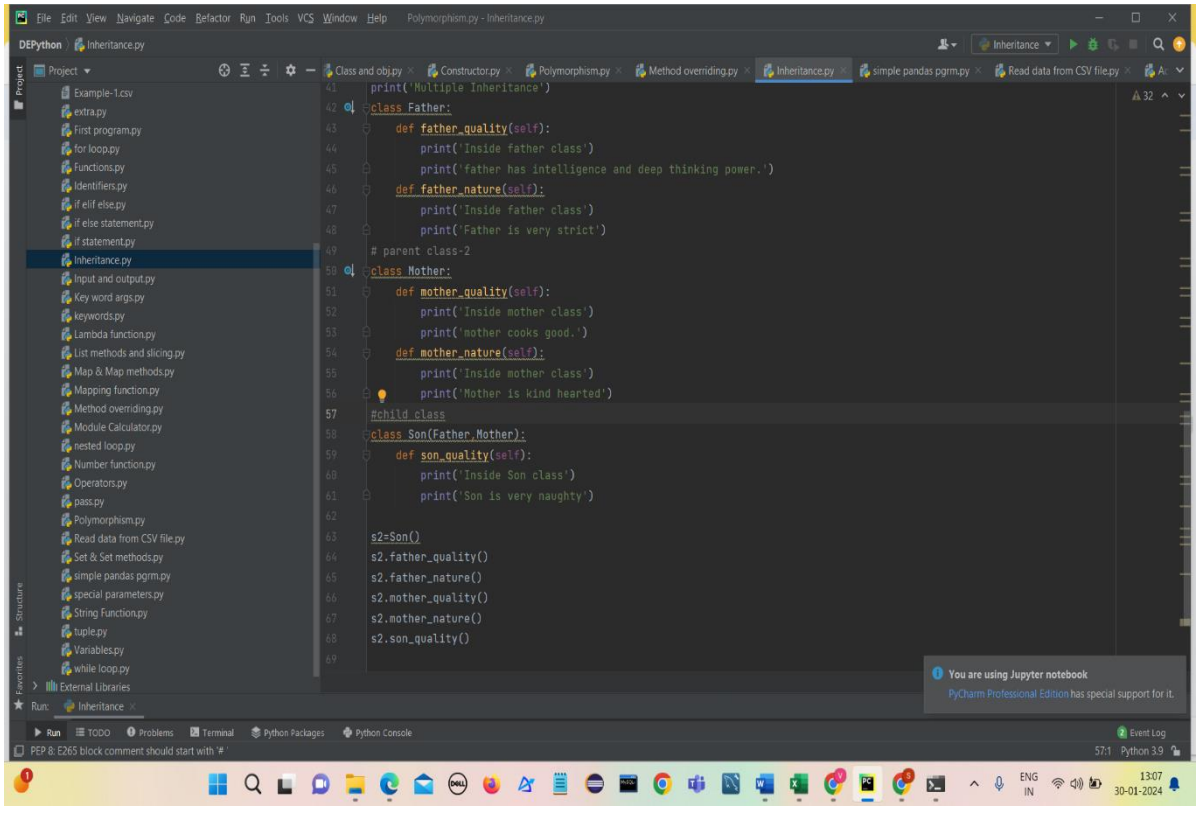
2) Multi level Inheritance



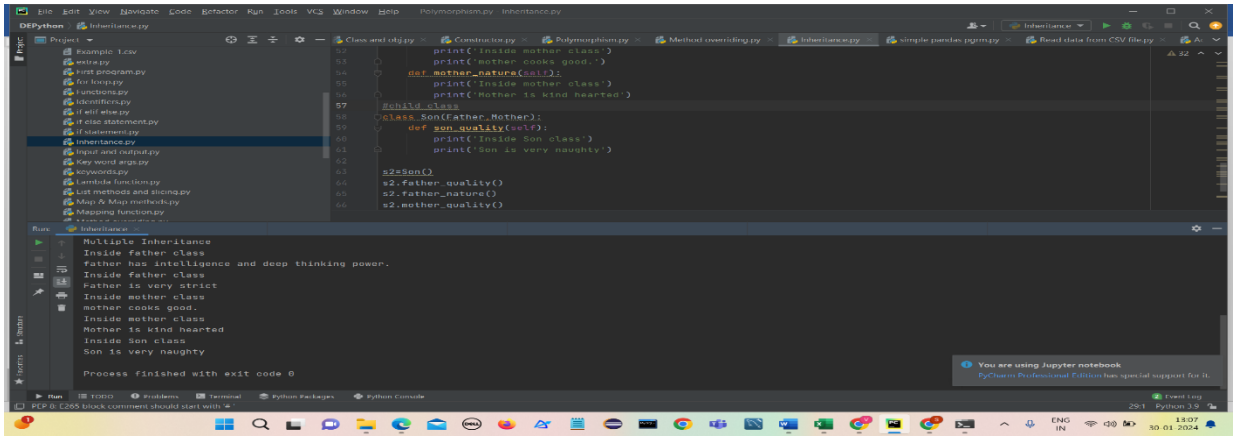
```
# 2) Multi level Inheritance
class Grandfather:
    def gf_quality(self):
        print('Inside Grand father class')
        print('Grand father is honest person')
class Father(Grandfather):
    def quality(self):
        print('Inside father class')
        print('father has intelligence and deep thinking power.')
class Son(Father):
    def aim(self):
        print('Inside Son class')
        print('Son wants to be Doctor')

# creating object for Son class
s1=Son()
s1.gf_quality()
s1.quality()
s1.aim()
```

```
Inside Grand father class
Grand father is honest person
Inside father class
father has intelligence and deep thinking power.
Inside Son class
Son wants to be Doctor

Process finished with exit code 0
```

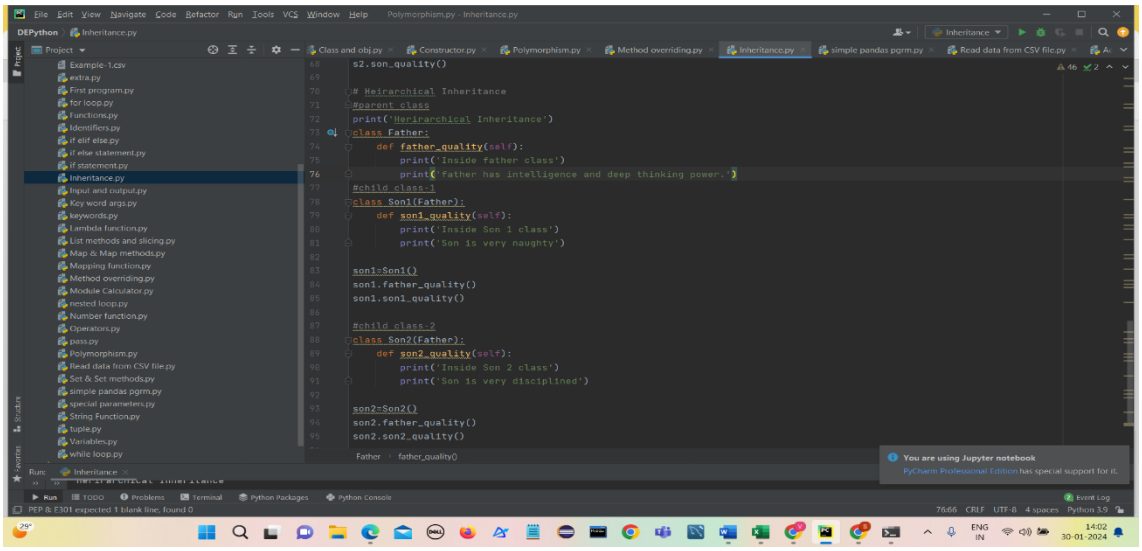
3) Multiple Inheritance



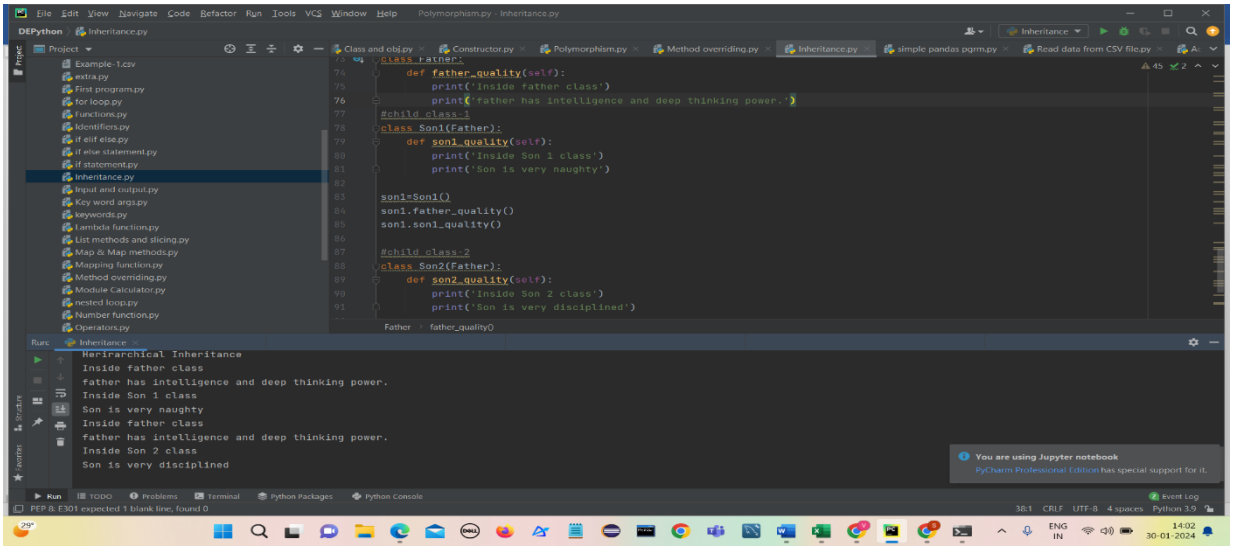
```
print('Multiple Inheritance')
class Father:
    def father_quality(self):
        print('Inside father class')
        print('father has intelligence and deep thinking power.')
    def father_nature(self):
        print('Inside father class')
        print('Father is very strict')
# parent class-2
class Mother:
    def mother_quality(self):
        print('Inside mother class')
        print('mother cooks good.')
    def mother_nature(self):
        print('Inside mother class')
        print('Mother is kind hearted')
#child class
class Son(Father,Mother):
    def son_quality(self):
        print('Inside Son class')
        print('Son is very naughty')

s2=Son()
s2.father_quality()
s2.father_nature()
s2.mother_quality()
s2.mother_nature()
s2.son_quality()
```



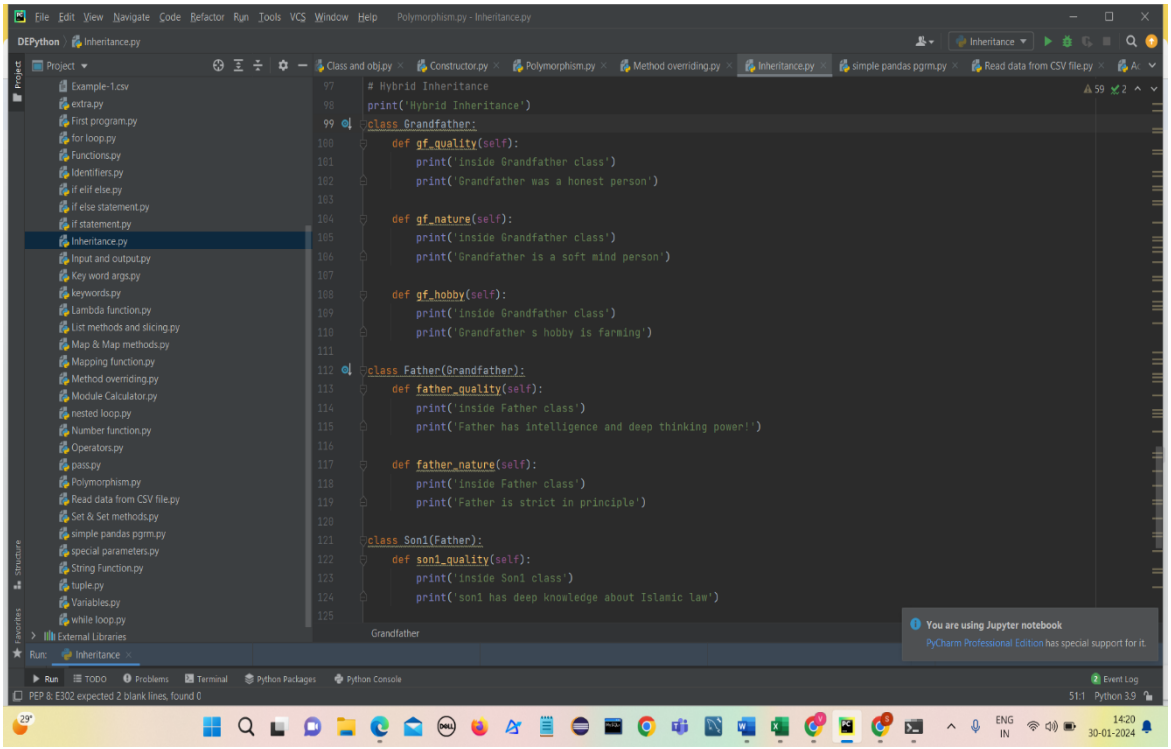
4) Hierarchical Inheritance



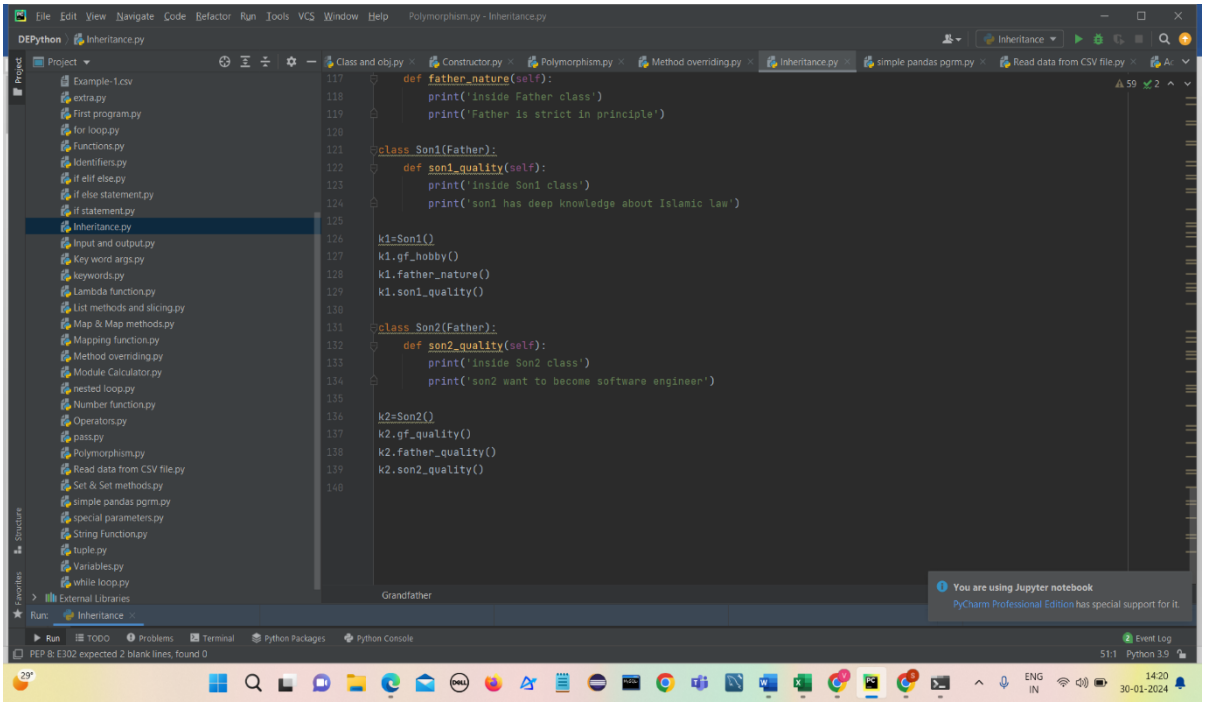
Output:



5) Hybrid Inheritance

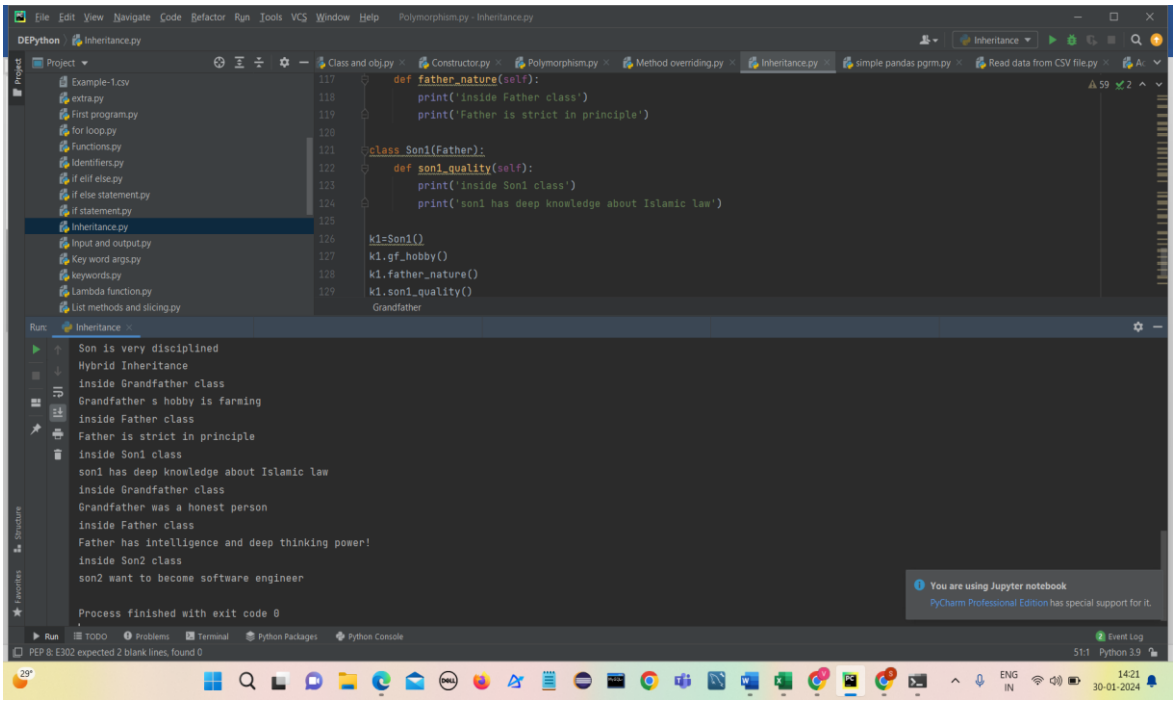


```
# Hybrid Inheritance
print('Hybrid Inheritance')
class Grandfather:
    def gf_quality(self):
        print('inside Grandfather class')
        print('Grandfather was a honest person')

    def gf_nature(self):
        print('inside Grandfather class')
        print('Grandfather is a soft mind person')

    def gf_hobby(self):
        print('inside Grandfather class')
        print('Grandfather s hobby is farming')

class Father(Grandfather):
    def father_quality(self):
        print('inside Father class')
        print('Father has intelligence and deep thinking power!')

    def father_nature(self):
        print('inside Father class')
        print('Father is strict in principle')

class Son1(Father):
    def son1_quality(self):
        print('inside Son1 class')
        print('son1 has deep knowledge about Islamic law')
```



```
    def father_nature(self):
        print('inside Father class')
        print('Father is strict in principle')

class Son1(Father):
    def son1_quality(self):
        print('inside Son1 class')
        print('son1 has deep knowledge about Islamic law')

k1=Son1()
k1.gf_hobby()
k1.father_nature()
k1.son1_quality()

class Son2(Father):
    def son2_quality(self):
        print('inside Son2 class')
        print('son2 want to become software engineer')

k2=Son2()
k2.gf_quality()
k2.father_quality()
k2.son2_quality()
```

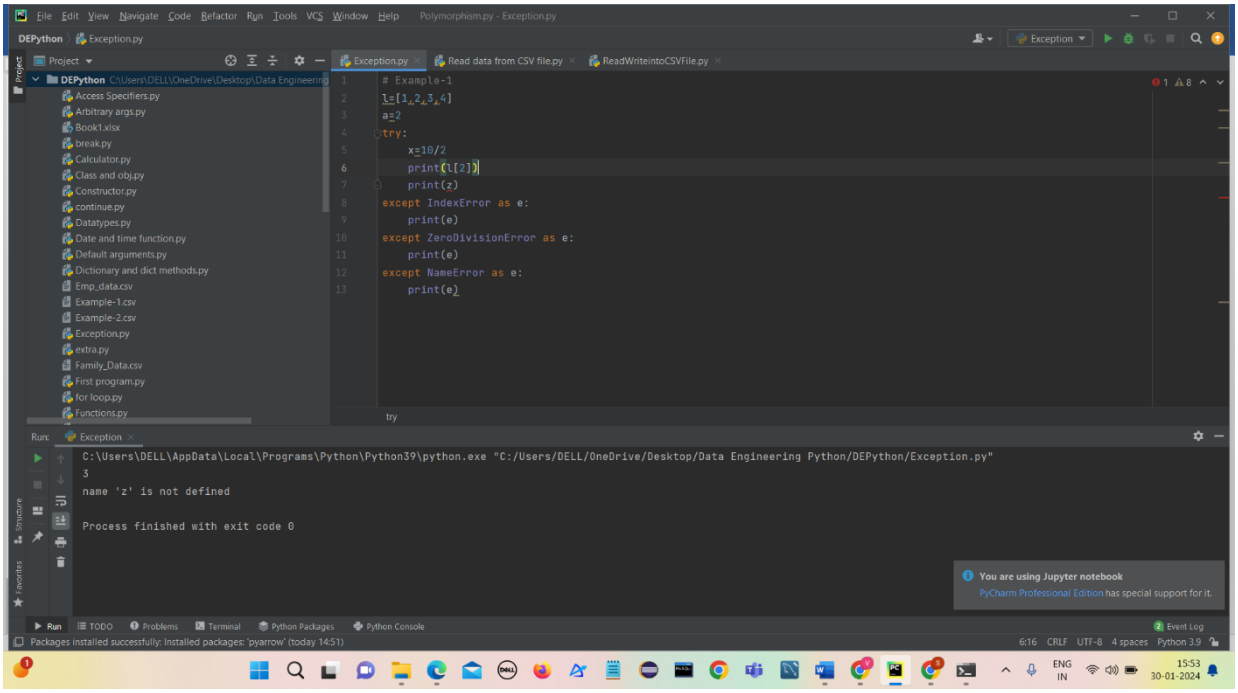
Output:



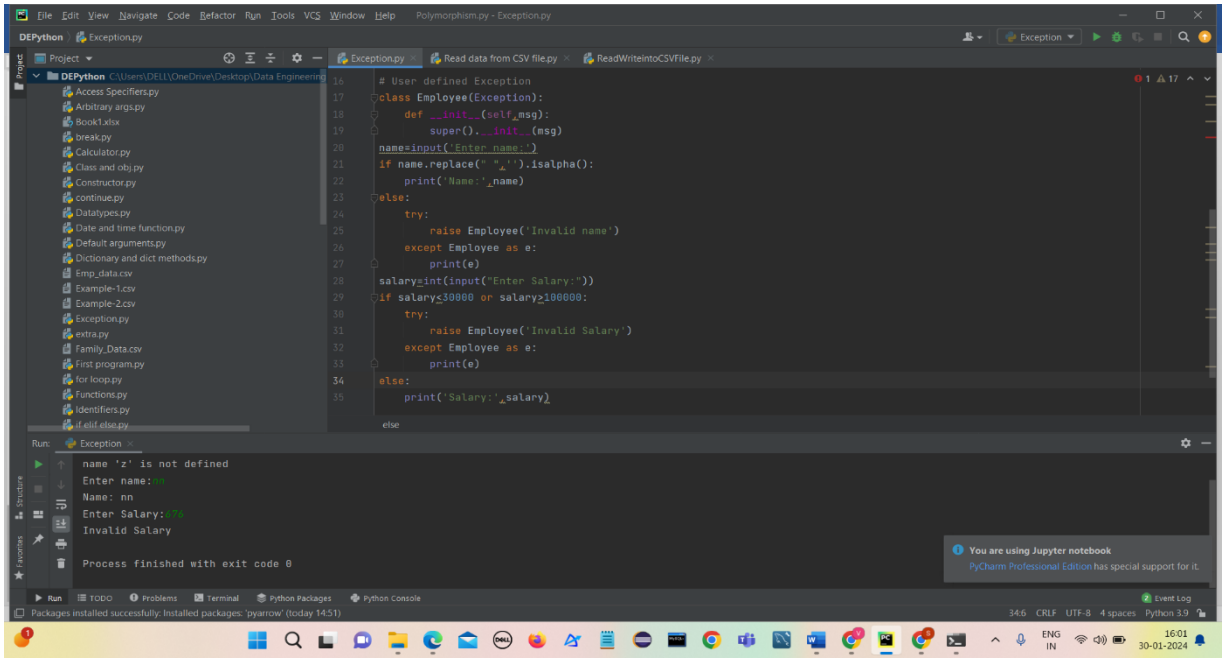
7) Exception Handling

Exception is an event which occurs during the execution of program that distrupts the normal flow of program

Exception handling can be done using try, except, else and finally statements.



User Defined Exception



8) File Handling

Python allows users to handle files i.e., to read and write files along with other operations.

Creating a file in Python:

1) Create a file using open()
2) Two parameters: filename, access mode to create a file

seek() :change or move cursor to place where data must be read or written in the file

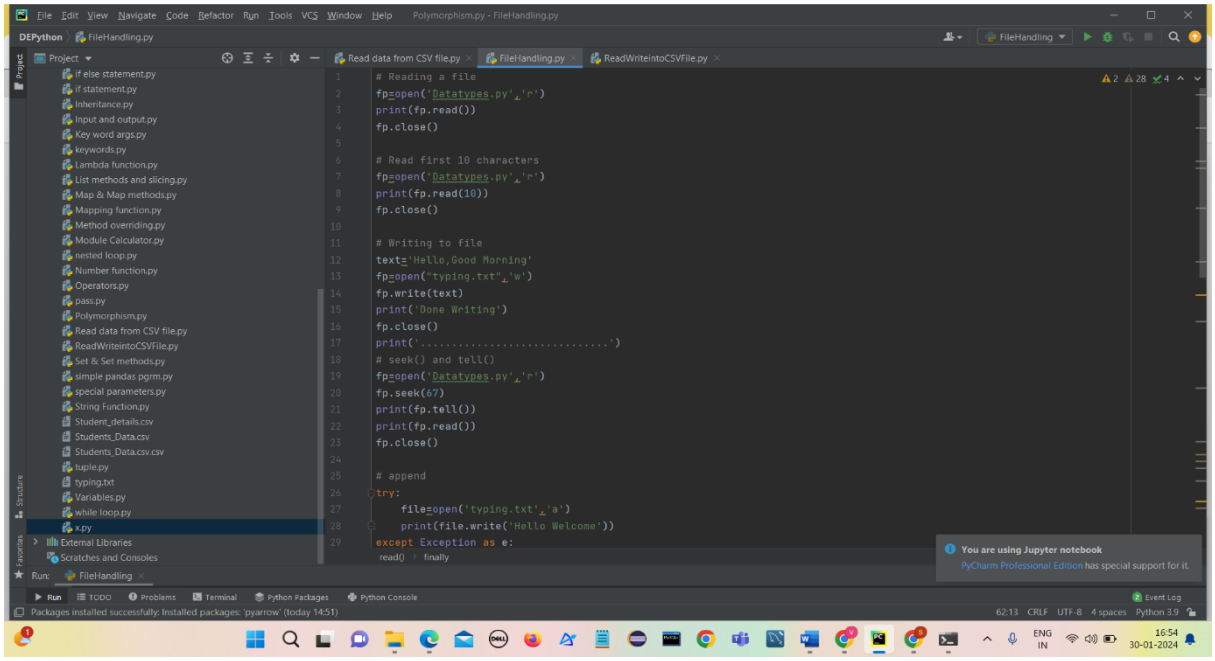
tell():return current position of file pointer from beginning of file

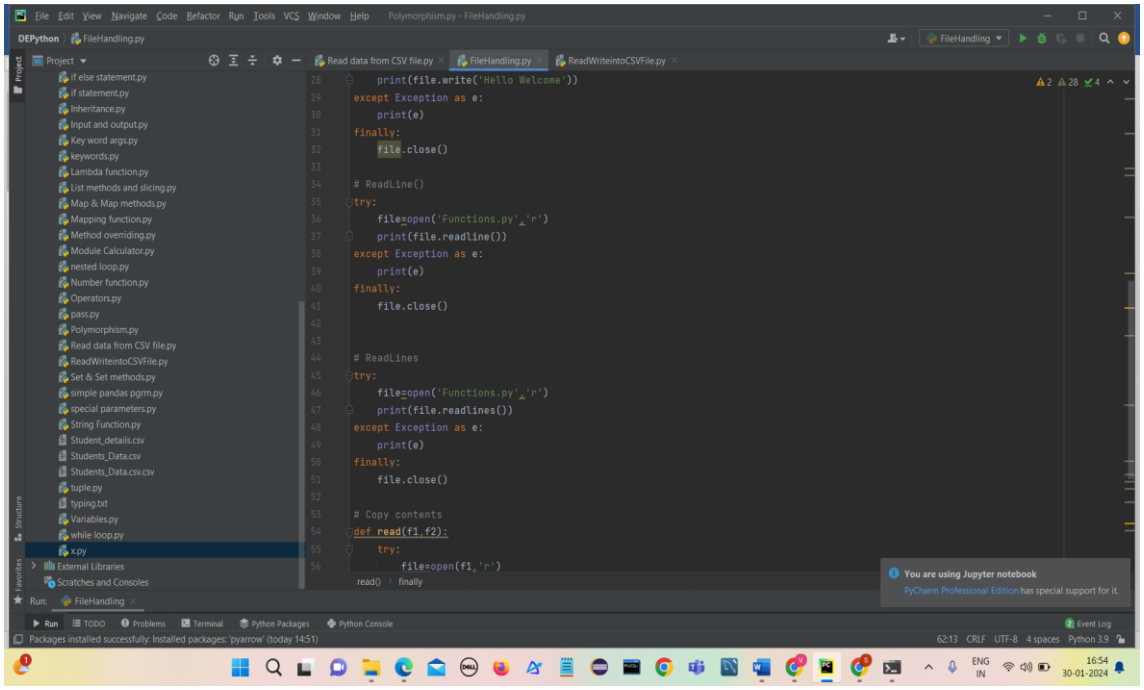
read():returns file content in the form of string

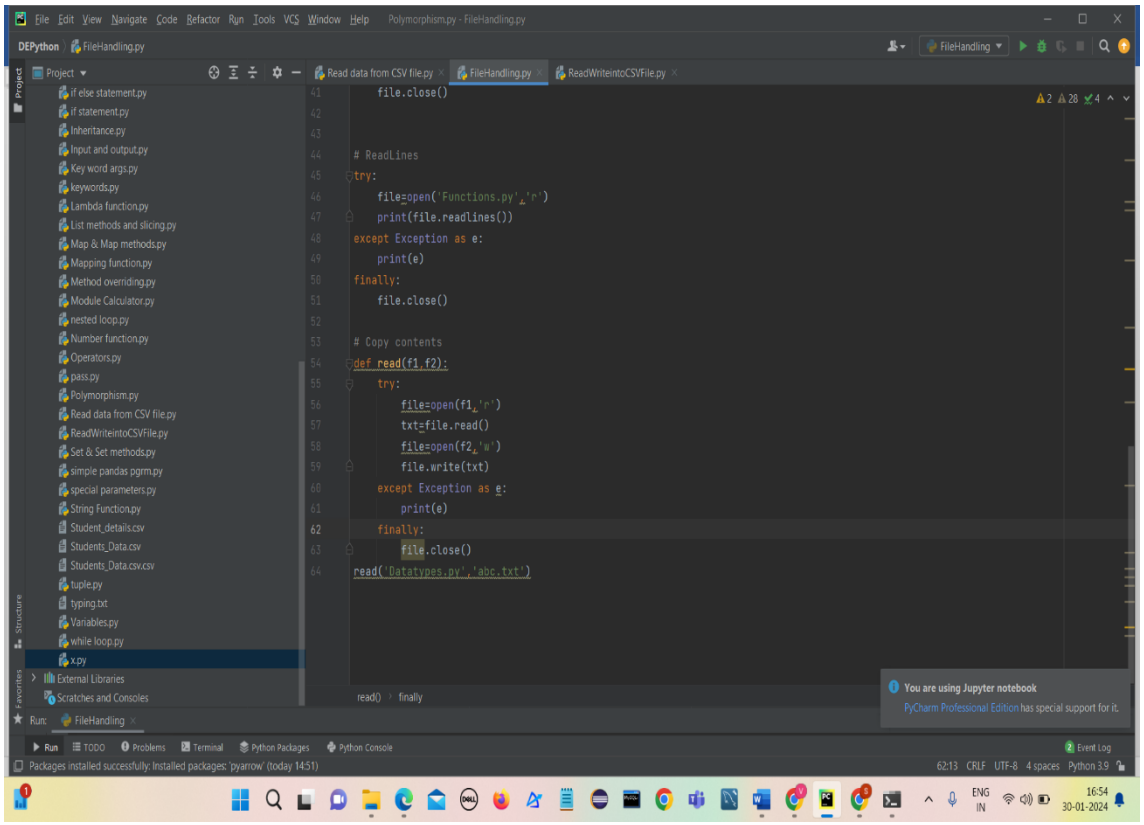
readLine(): Reads single line

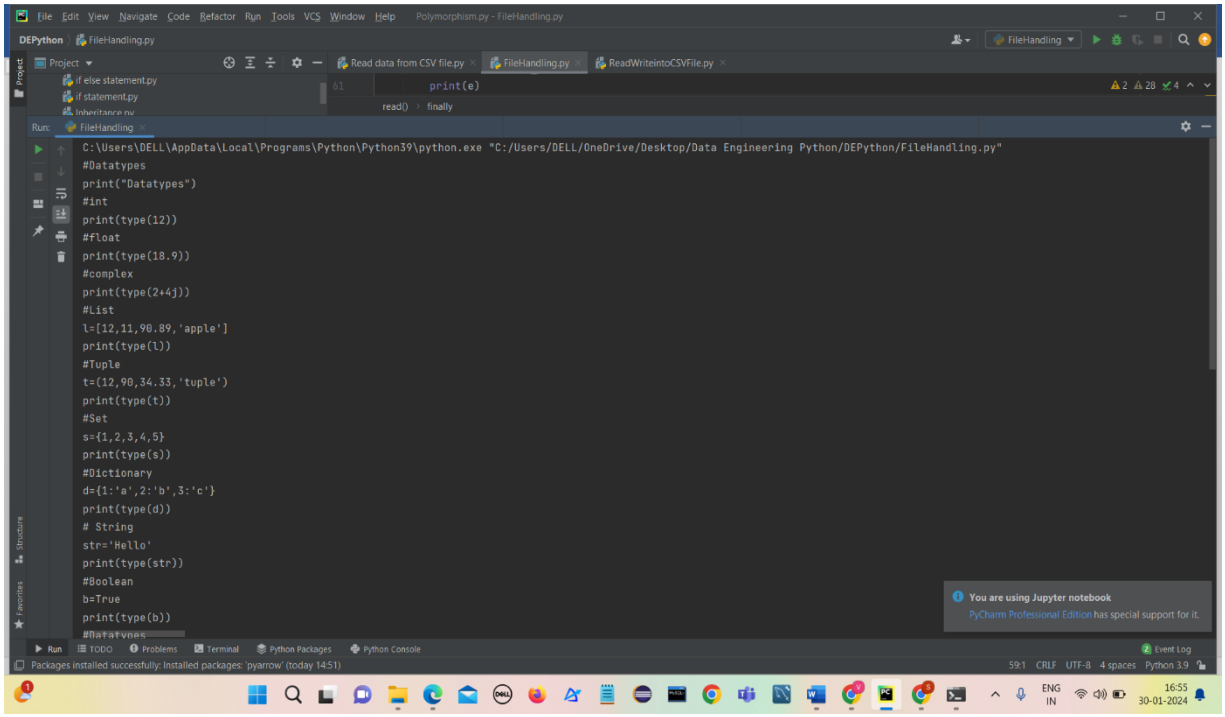
readLines():read file into list

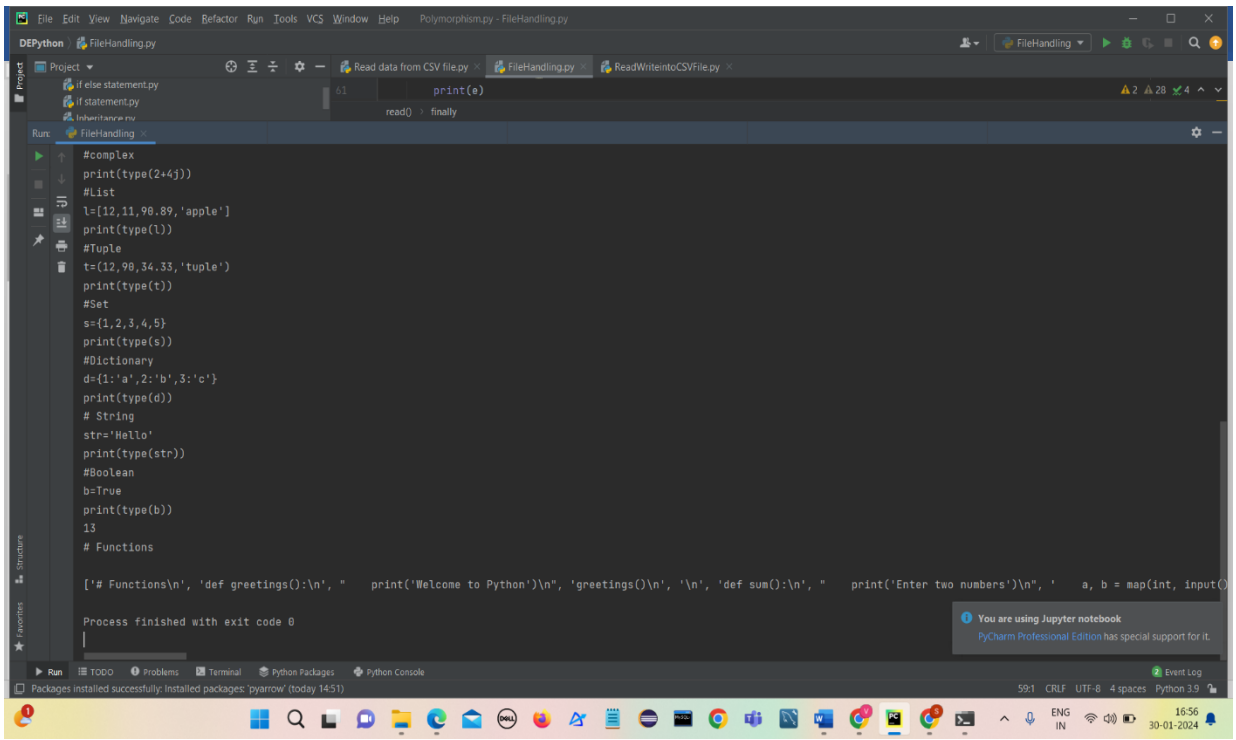
writelines():write list of strings to file


File Edit View Navigate Code Refactor Run Tools VCS Window Help Polymorphism.py - FileHandling.py
DEPython FileHandling.py
FileHandling
Project
Read data from CSV file.py
FileHandling.py
ReadWriteintoCSVFile.py
if else statement.py
if statement.py
Inheritance.py
Input and output.py
Key word args.py
keywords.py
Lambda function.py
List methods and slicing.py
Map & Map methods.py
Mapping function.py
Method overriding.py
Module Calculator.py
nested loop.py
Number function.py
Operators.py
pass.py
Polymorphism.py
Read data from CSV file.py
ReadWriteintoCSVFile.py
Set & Set methods.py
simple pandas pgrm.py
special parameters.py
String Function.py
Student_details.csv
Students_Data.csv
Students_Data.csv.csv
tuple.py
typing.txt
Variables.py
while loop.py
x.py
External Libraries
Scratches and Consoles
# Reading a file
fp=open('Datatypes.py','r')
print(fp.read())
fp.close()
# Read first 10 characters
fp=open('Datatypes.py','r')
print(fp.read(10))
fp.close()
# Writing to file
text='Hello,Good Morning'
fp=open("typing.txt",'w')
fp.write(text)
print('Done Writing')
fp.close()
print('.....')
# seek() and tell()
fp=open('Datatypes.py','r')
fp.seek(67)
print(fp.tell())
print(fp.read())
fp.close()
# append
try:
file=open('typing.txt','a')
print(file.write('Hello Welcome'))
except Exception as e:
read() finally
You are using Jupyter notebook
PyCharm Professional Edition has special support for it.
Run: FileHandling
Run TODO Problems Terminal Python Packages Python Console
Event Log
Packages installed successfully: Installed packages: 'pyarrow' (today 14:51)
62:13 CRLF UTF-8 4 spaces Python 3.9
ENG IN
16:54 30-01-2024


File Edit View Navigate Code Refactor Run Tools VCS Window Help Polymorphism.py - FileHandling.py
DEPython FileHandling.py
FileHandling
Project
Read data from CSV file.py
FileHandling.py
ReadWriteintoCSVFile.py
print(file.write('Hello Welcome'))
except Exception as e:
print(e)
finally:
file.close()
# ReadLine()
try:
file=open('Functions.py','r')
print(file.readline())
except Exception as e:
print(e)
finally:
file.close()
# ReadLines
try:
file=open('Functions.py','r')
print(file.readlines())
except Exception as e:
print(e)
finally:
file.close()
# Copy contents
def read(f1,f2):
try:
file=open(f1,'r')
read() finally
You are using Jupyter notebook
PyCharm Professional Edition has special support for it.
Run: FileHandling
Run TODO Problems Terminal Python Packages Python Console
Event Log
Packages installed successfully: Installed packages: 'pyarrow' (today 14:51)
62:13 CRLF UTF-8 4 spaces Python 3.9
ENG IN
16:54 30-01-2024

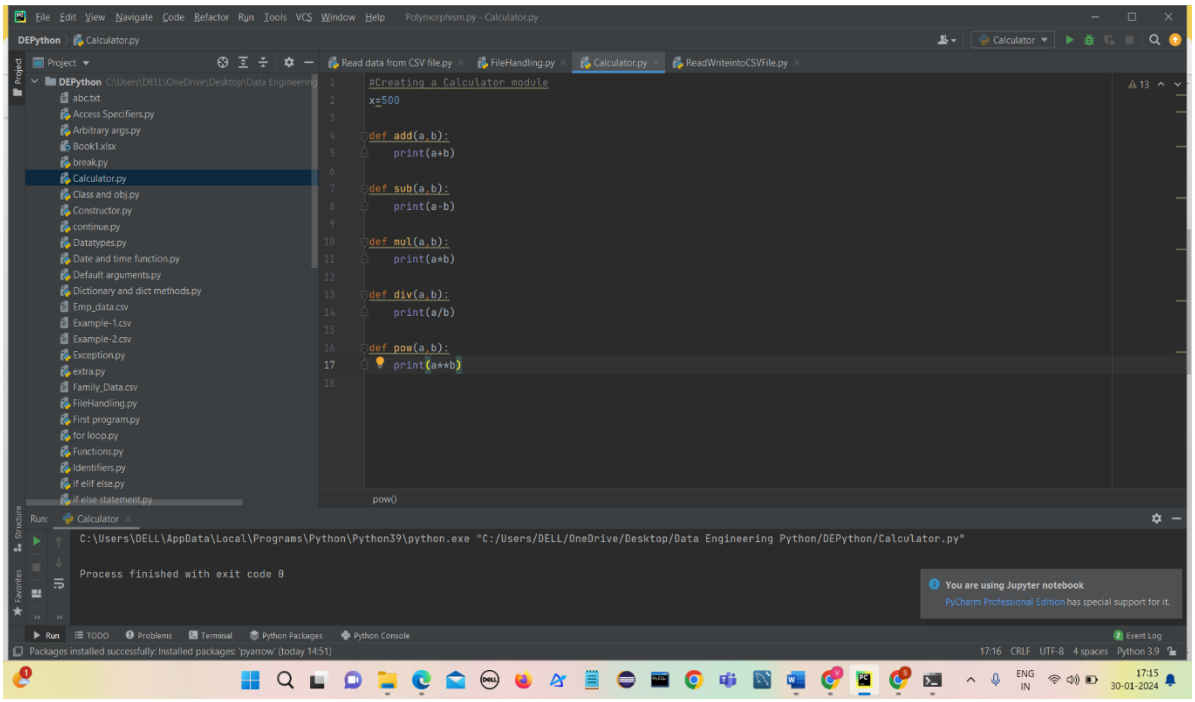

File Edit View Navigate Code Refactor Run Tools VCS Window Help Polymorphism.py - FileHandling.py
DEPython FileHandling.py
FileHandling
Project
Read data from CSV file.py
FileHandling.py
ReadWriteintoCSVFile.py
if else statement.py
if statement.py
Inheritance.py
Input and output.py
Key word args.py
keywords.py
Lambda function.py
List methods and slicing.py
Map & Map methods.py
Mapping function.py
Method overriding.py
Module Calculator.py
nested loop.py
Number function.py
Operators.py
pass.py
Polymorphism.py
Read data from CSV file.py
ReadWriteintoCSVFile.py
Set & Set methods.py
simple pandas pgrm.py
special parameters.py
String Function.py
Student_details.csv
Students_Data.csv
Students_Data.csv.csv
tuple.py
typing.txt
Variables.py
while loop.py
x.py
External Libraries
Scratches and Consoles
file.close()
# ReadLines
try:
file=open('Functions.py','r')
print(file.readlines())
except Exception as e:
print(e)
finally:
file.close()
# Copy contents
def read(f1,f2):
try:
file=open(f1,'r')
txt=file.read()
file=open(f2,'w')
file.write(txt)
except Exception as e:
print(e)
finally:
file.close()
read('Datatypes.py','abc.txt')
read() finally
You are using Jupyter notebook
PyCharm Professional Edition has special support for it.
Run: FileHandling
Run TODO Problems Terminal Python Packages Python Console
Event Log
Packages installed successfully: Installed packages: 'pyarrow' (today 14:51)
62:13 CRLF UTF-8 4 spaces Python 3.9
ENG IN
16:54 30-01-2024


File Edit View Navigate Code Refactor Run Tools VCS Window Help Polymorphism.py - FileHandling.py
DEPython FileHandling.py
Project
Read data from CSV file.py
FileHandling.py
ReadWriteintoCSVFile.py
if else statement.py
if statement.py
print(e)
read() finally
Run: FileHandling
C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/FileHandling.py"
#Datatypes
print("Datatypes")
#int
print(type(12))
#float
print(type(18.9))
#complex
print(type(2+4j))
#List
l=[12,11,90.89,'apple']
print(type(l))
#Tuple
t=(12,90,34.33,'tuple')
print(type(t))
#Set
s={1,2,3,4,5}
print(type(s))
#Dictionary
d={1:'a',2:'b',3:'c'}
print(type(d))
# String
str='Hello'
print(type(str))
#Boolean
b=True
print(type(b))
#Datatypes
You are using Jupyter notebook
PyCharm Professional Edition has special support for it.
Run TODO Problems Terminal Python Packages Python Console
Event Log
Packages installed successfully: Installed packages: 'pyarrow' (today 14:51)
59:1 CRLF UTF-8 4 spaces Python 3.9
ENG IN
16:55
30-01-2024

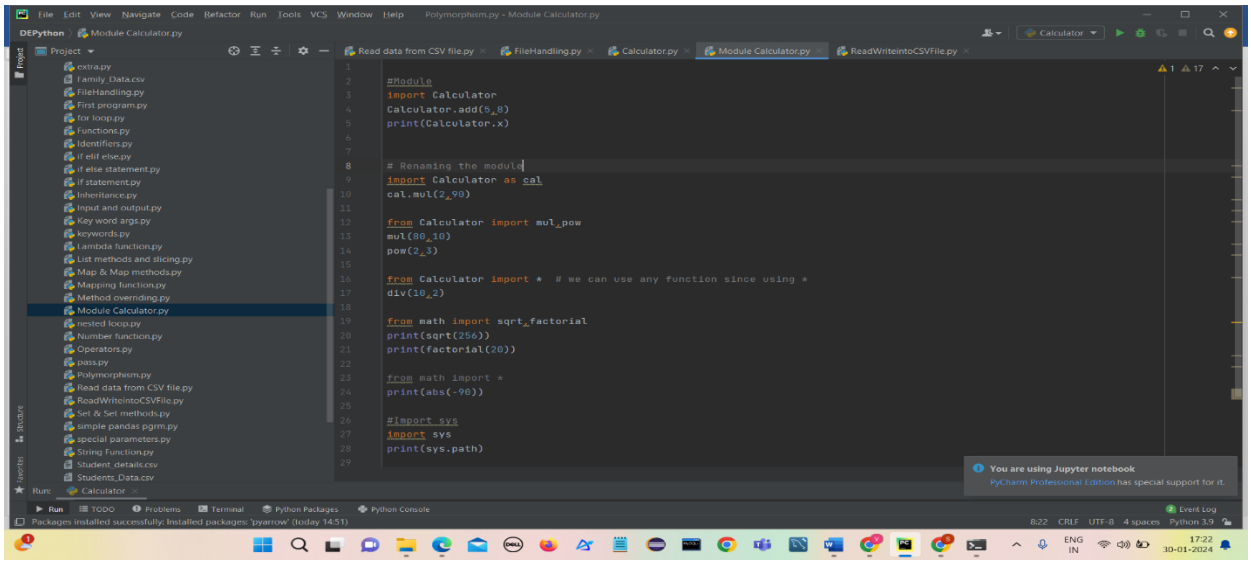

File Edit View Navigate Code Refactor Run Tools VCS Window Help Polymorphism.py - FileHandling.py
DEPython FileHandling.py
Project
Read data from CSV file.py
FileHandling.py
ReadWriteintoCSVFile.py
if else statement.py
if statement.py
print(e)
read() finally
Run: FileHandling
#complex
print(type(2+4j))
#List
l=[12,11,90.89,'apple']
print(type(l))
#Tuple
t=(12,90,34.33,'tuple')
print(type(t))
#Set
s={1,2,3,4,5}
print(type(s))
#Dictionary
d={1:'a',2:'b',3:'c'}
print(type(d))
# String
str='Hello'
print(type(str))
#Boolean
b=True
print(type(b))
13
Functions
['# Functions\n', 'def greetings():\n', "   print('Welcome to Python')\n", 'greetings()\n', '\n', 'def sum():\n', "   print('Enter two numbers')\n", '   a, b = map(int, input()
Process finished with exit code 0
You are using Jupyter notebook
PyCharm Professional Edition has special support for it.
Run TODO Problems Terminal Python Packages Python Console
Event Log
Packages installed successfully: Installed packages: 'pyarrow' (today 14:51)
59:1 CRLF UTF-8 4 spaces Python 3.9
ENG IN
16:56
30-01-2024

9) Modules

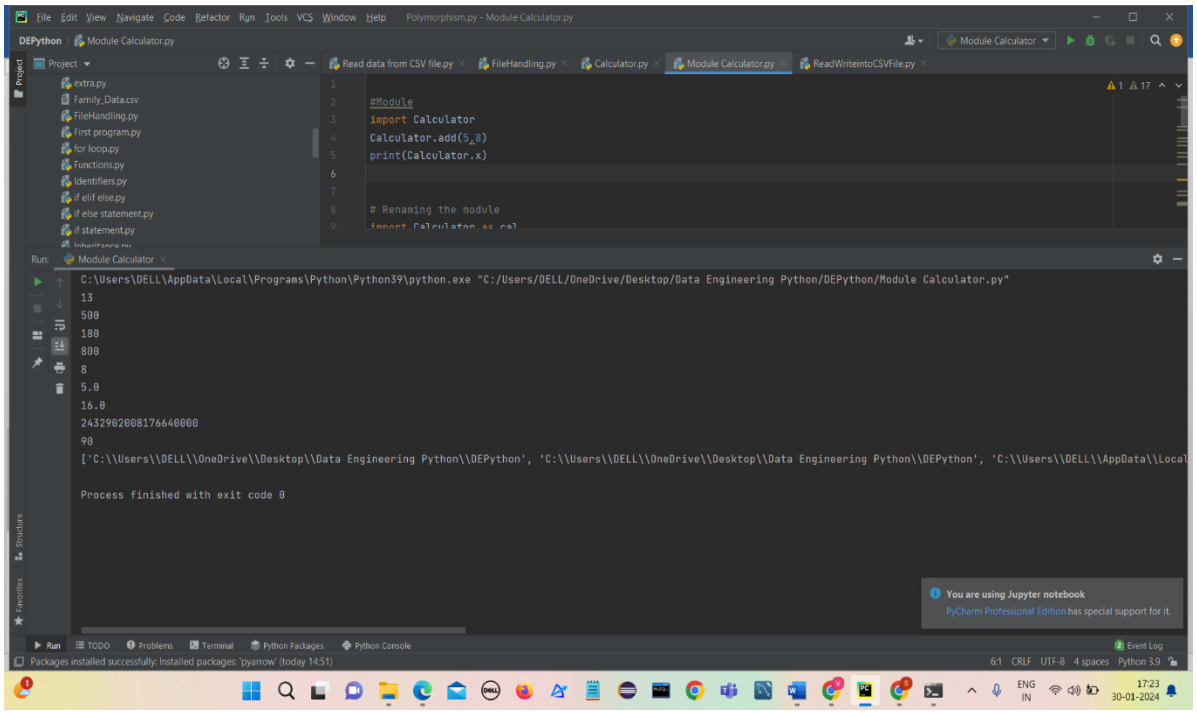
- Module is a file containing Python code.
- The code within a module can define functions, classes, and variables that can be utilized in other Python scripts or modules.
- Modules help organize code into logical units, making it easier to manage and reuse.
- Modules can be shared, reused and collaborated with others.



Calculator.py is a module containing all python functions to add,subtract,multiply,divide and calculate power of two numbers.



The methods in Calculator module can be used by importing the required functions.



Different ways to import:

1) import module_name
2) Renaming a module---------- import module_name as xyz
3) From module_name import fun1
4) To import all functions--------- from module_name import *