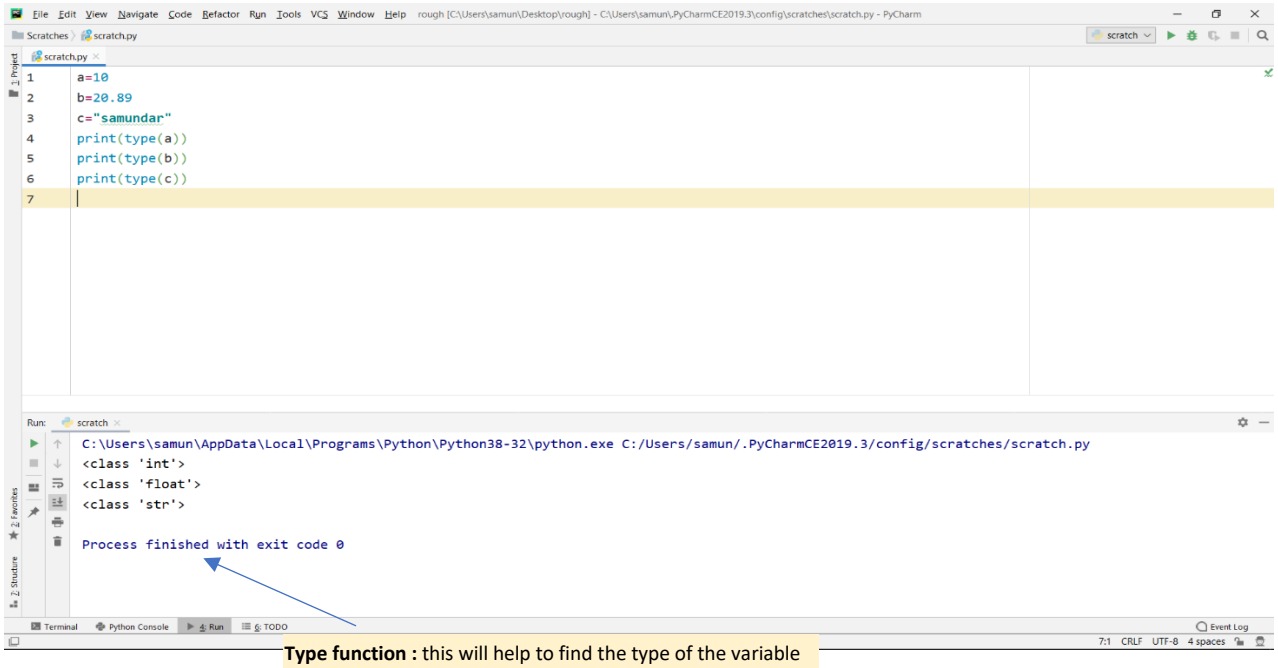
PYTHON PROGRAMMING

Basic of Python:

1.Variables declaration

For integer: int.
For float: float.
For string: str.

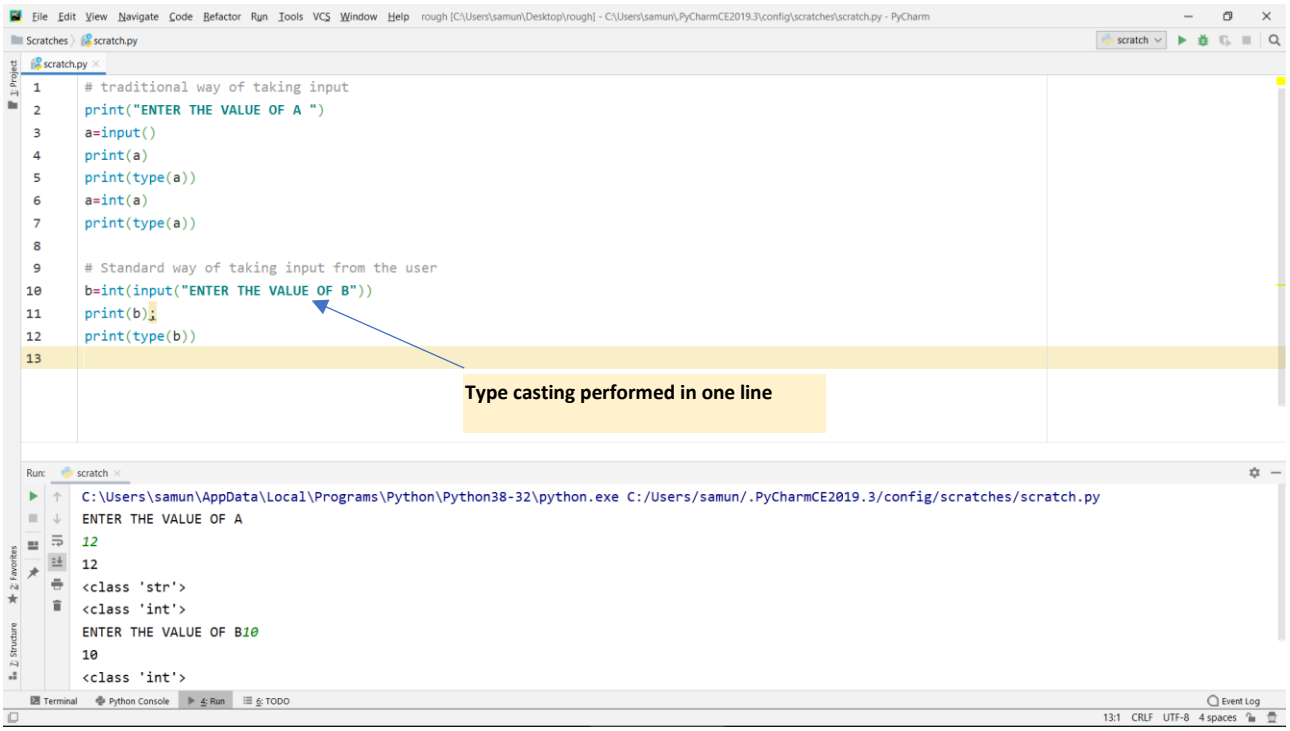
Note: Python automatically initialize the type of variables but when we ask input from user then it only take the input in string format so by using type casting we convert the string according to our requirement.



Type function : this will help to find the type of the variable

2.Taking input from user and perform typecasting

- The **input ()** function is used to take input from the user. By default, it takes string as the input so if we want to take integer or float then we used the typecasting for that



3.List in python

- It is a collection of the data that may be of different types. It will act as an array but the major difference is that an array contains group of similar data type whereas a list contains group of similar as well as different data types.
- The data items present between the **big braces [ ]** and we can access the element using the indices number.
- List are **mutable** (can change).

Methods:

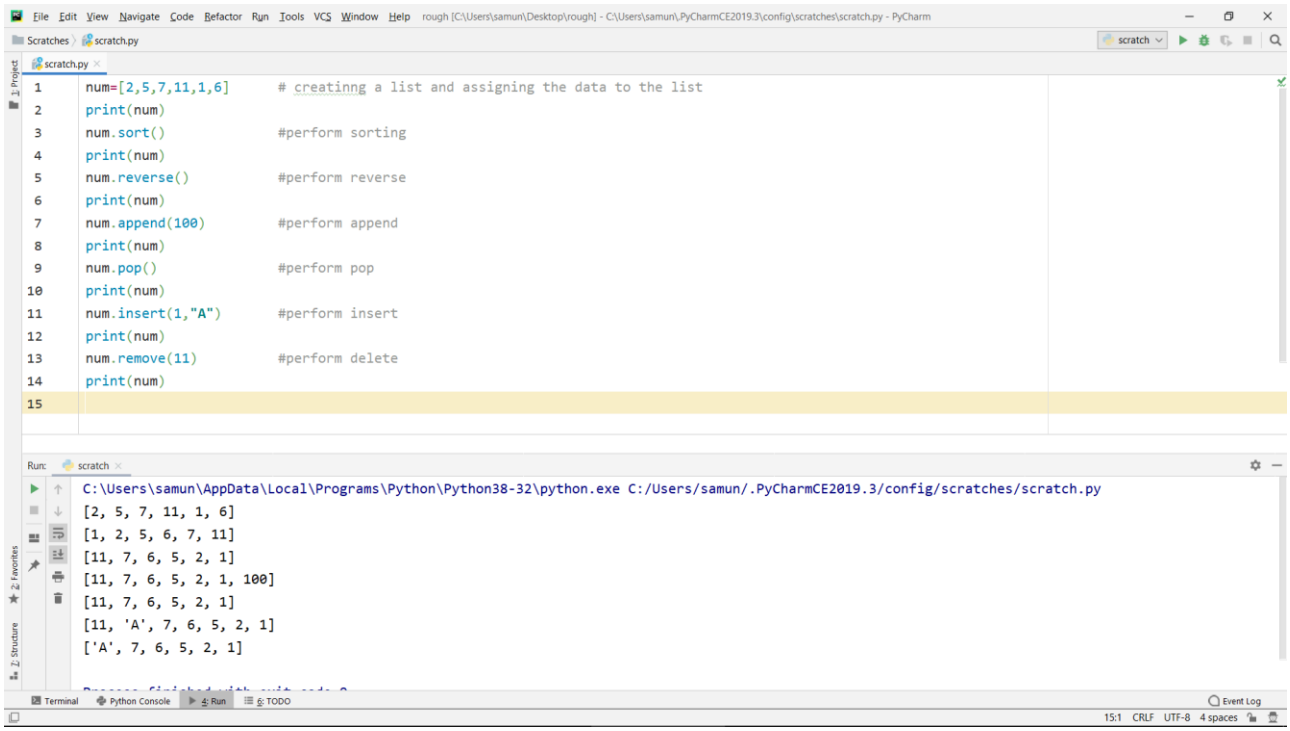
creating a empty list say **number= [ ]** **(creating a empty list)**

Here we are creating a list say **number= [2,7,9,11,3]** **(when data is assigned at same time)**

- **number. Sort()** - it will sort the number.
- **number. reverse()** - it will reverse the list .
- **max(number)** - it will find the maximum number from the list.
- **min(number)** - it will find the minimum number from the list.
- **len(number)** - it will find the length of the list.
-

To alter the data of the list:

- **number append(data)** - it will insert data in list(at the last) .
- **number pop(data)** - it will remove data from the list(at the last) .
- **number. insert(index, data)** - it will insert data in list to the index which is mentioned.
- **number. remove(data)** - it will delete data from list wherever the data is.



4.tuple in python:

- It is a collection of data of different type same as list but it can't have same method as the list have.
- It is immutable in nature means its data can't be altered.

creating a empty list say **number= ()** **(creating a empty tuple)**

Here we are creating a list say **number= (2,7,9,11,3)** **(when data is assigned at same time)**

5.Dictionary in python

- It is a data type in python which contains a key -value pair to store the data.
- The data can be accessed by the help of the key.
- It is mutable in nature.

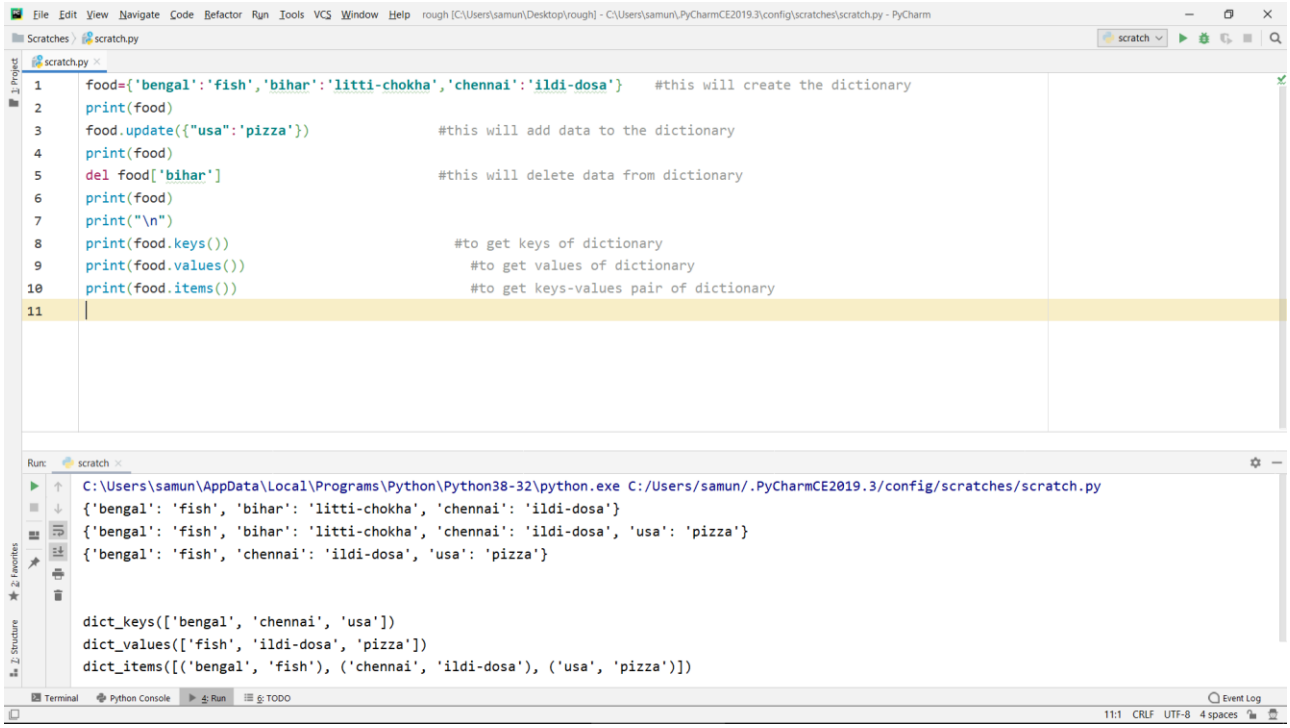
Methods:

creating a empty dictionary say **food= {}** **(creating a empty dictionary)**
here we are creating a dictionary food={“Bengal”: “fish”, “Bihar”:”Litti-chokka”,”Chennai”: “idli-dosa”}

To alter the data of the Dictionary:

- **food.update({key:value})** - it will insert data in dictionary .
- **del food[key]** - it will delete data from dictionary.

- **food.get(key)** - it will fetch the value of key from dictionary.
- **food.keys()** - it will fetch the keys of dictionary.
- **food.values()** - it will fetch the value of key from dictionary.
- **food.items()** - it will fetch the key-value pair from dictionary.



Note: There is a special method called **copy()** use to copy one dictionary and store its value to another we require to use copy() function because when we copy directly one dictionary to other dictionary then if we want some changes in other dictionary then it will also reflect to that dictionary from where the data had been copied

6.Exception handling in python

There are some situation in which the program will throw an error on not based on the user input. if the program run well then there is no problem but if the program has some error then the execution gets stopped at that place where error occur. So avoid such situation and run our program smoothly we use exception handling.

First benefit of using exception handling is that we know where the error occurs and second benefit is that our program doesn't stop while error encounter.

Types of exception:

1.Built-in exception: - These are the exception which are already available in python language.
2.User defined exception: - A programmer can create his own exception called user-defined exception.

Some points to remember:
1.we can write several except block for a single try block.
2.we can write multiple except block to handle multiple exception.
3.we can write try block without any except block.
4.but we can't write except block without any try block.
5.finally block will always executed whether there is an exception or not
6.Else and finally block is optional.

Syntax:

```
try:
    statement……
except Exception class name:
    statement……

else:
    statement……
finally:
    statement……
```

How to write the Exception:

Built-in Exception:

First we have to find what type of possible are going to occur in the program according to which we have to implement the handling procedure .Here some of the exception are mentioned:

1. **type error:** this type of error occur when are performing some operation between two different type of variables (e.g. addition of an integer and a character)

input:

```
try:
    a=int(input("please enter a number : "))
    b=input("enter another number : ")
    c=a+b
except TypeError as e:
    print(e)
print("sucess")
```

Output

```
please enter a number :12
enter another number : rt
unsupported operand type(s) for +: 'int' and 'str'
sucess
```

2. **Zero division error:** this error occur when we perform the division of a number with zero

input:

```
a=int(input("please enter a number"))
b=int(input("please enter a number"))

try:
    a/b
except ZeroDivisionError as e:
    print(e)

print("sucess")
```

Output

```
please enter a number :12
enter another number : rt
unsupported operand type(s) for +: 'int' and 'str'
sucess
```

3. **value error:** this error occur a function receive an argument that has a correct type but value provide to it by user is incorrect (e.g. when user try to insert a string when the input takes only int value)

input:

```
while(1):
    try:
        a=int(input("please enter a
number"))
        b=int(input("enter another number"))
        break
    except ValueError as e:
        print(e)
print("sum is : ",a+b)
```

Output

```
please enter a number: we
invalid literal for int() with base 10: 'we'
please enter a number12
enter another number34
sum is :  46
```

4. **Keyboard interrupt:** suppose if we do not want to execute a particular piece of code than by help of the keyboardinterrupt we can skip that code.

input:

```
try:
    inp=input("enter anything")
    print("this statement run if there is no
keyboard interrupt")
except KeyboardInterrupt:
    print("keybord interrupt occur")

print("SUCCESS")
```

Output

```
enter anything

#after pressing ctrl+c
keyboard interrupt occur
SUCCESS
```

User defined Exception:

There are 3 steps to create a user defined exception:

1. Creating exception class
2. Raising exception
3. Handling exception

input:

```
class MyExcepion(Exception):
    pass
def checkbalance():
    money=10000
    print("you have ",money," in your account")
    withdrwal=int(input("enter the withdrawl amount : "))
    global balance
    balance=money-withdrwal
    if(balance<2000):
        raise MyExcepion("balance insufficient")
try:
    checkbalance()
    print("transanction sucessfull available balance : ",balance)
except MyExcepion as e:
    print(e)
print("this is going to work every time to show the exception handling flow")
```

Output

```
you have  10000  in your account
enter the withdrawl amount : 9000
balance insufficient
this is going to work every time to show the exception handling flow
```

7.File handling in python

Python help us to read and write or create a file. The various Acess Modes are:

- "r" : opens a file for read only.
- "w" : opens a file for write only.
- "r+": opens a file for both read and write. Does not overwrite (doesn't delete data when executed)
- "w+": opens a file for both read and write overwrite the file if file exist otherwise creates a new file.
- "a" : it is append mode used to only append any data at the last of file not for read.
- "a+": it will append data and also allow to read the data.

The open() function:

To open any file we use open() function.

Syntax:

fileobj=open("*file name*", "*access mode* ")

NOTE: Here we create a object which helps us in using the various method related to file handling

Method for reading and writing file:

1.) **read() Method** : This method is used to read a string from a open file
syntax: content=fileobject.read()
2.) **write()Method** :this method is used to write string to an open file.The access mode must be in write mode.
syntax: fileobject.write(string)
3.) **Close()method** :it will close the file after which the data cannot be written to the opened file
syntax: fileobject.close()

NOTE: We will perform iteration and able to access each word from the text file and later we will modify it

input:

```
fileobj=open('sam.txt',"r+")       #it will create a file object from which other method is called
content=fileobj.read()
print(content)
fileobj.write("\n hello samundar how are you ?")
fileobj.close()

#again we are acessing the file

print("-----------again Acesss the same file -----------")
obj2=open("sam.txt","r+")
content2=obj2.read()
print(content2)
```

Output:

```
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe
C:/Users/samun/.PyCharmCE2019.3/config/scratches/scratch.py
hello my name is samundar singh
i am a good boy
i am from jharkhand

-----------again Acesss the same file -----------
hello my name is samundar singh
i am a good boy
i am from jharkhand


 hello samundar how are you ?

Process finished with exit code 0
```

8.Recursion in python

When a function call itself then this process called a recursion. since the function call itself so there are infinite calling if we do not stop the calling of function so there must be some way which interrupt the calling of function. This interrupt logic is called **Base case**. Generally in base case there is a **if statement** which return some value and thus the calling breaks.

Syntax:

```
def function(argument):
    if(argument==0):
        return 1
    else:
        function(logic)

function()
```

Function defined

Base case

Here we write the logic which call the function recursively

Here the function called

For better understanding a example to show the implementation is shown here:

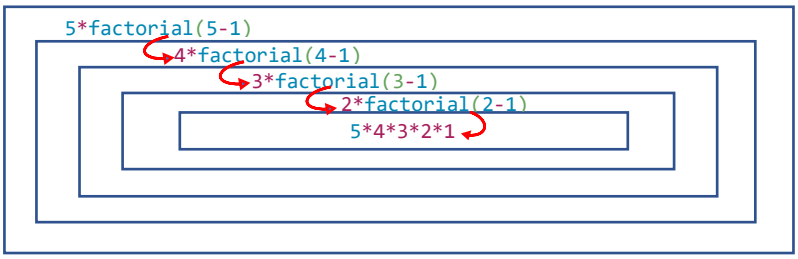
Example for finding factorial of a number using recursion

input

```
def factorial(n):
    if(n==1):
        return 1
    else:
        return n*factorial(n-1)

n=int(input("enter the number : "))
result=factorial(n)
print("factorial of ",n," is ",result)
```

Output

```
enter the number : 5
factorial of  5  is  120
```



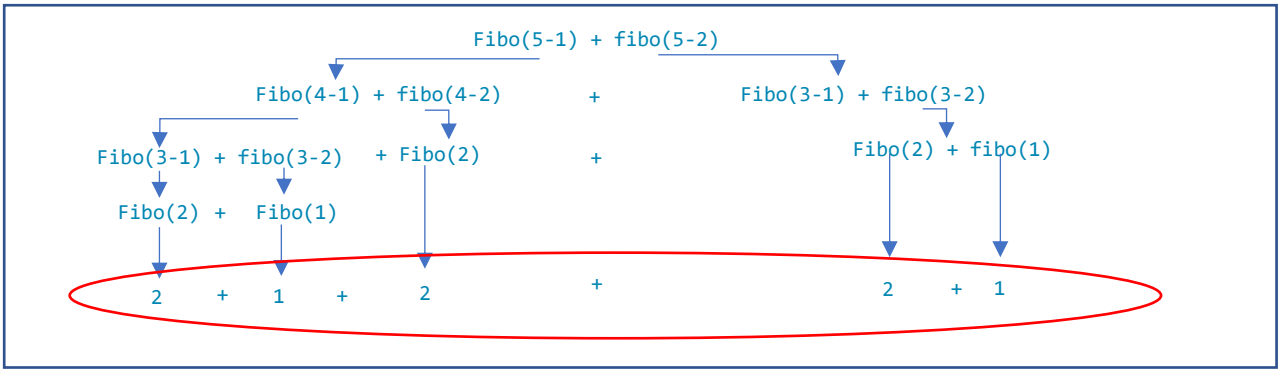
Example for finding Fibonacci series using recursion

input

```
def fibo(n):
    if(n==1):
        return 1
    elif(n==2):

        return 2
    else:
        return fibo(n-1)+fibo(n-2)

n=int(input("enter the number : "))
```

Output

```
enter the number : 5
fibbonaci of  5  is  8
```



9.F-string in python

- This function is used when we have to insert variable in a string specially in between coding
- Before declaring a sting we have to write “f” before the quotes.
- This f string is useful in data insert and retrieve from data base

input

```
a=(input("enter name :"))
b=int(input("enter age : "))
c=input("enter the address : ")

print(f"my name is {a} and i am {b} years old. i live in {c}")

print('my name is %s and i am %d years old. i live in %s'%(a,b,c))
```

Using f sting

Normal method

10.passing multiple argument in function

- We use *var to take multiple argument in function.
- The *var store the multiple data inside the function as a tuple of data.
- Always write the *var as the last argument of the function.
- We can pass data in function in any type like list, tuple, multiple declare variables etc.
- We can pass the dictionary also in the function by using ** var2.
- The *var is always written before the **var2.

Example : passing multiple argument :

input

```
#creating a list of student
def student(*data1):
    print('the student are :')
    for i in data1:
        print(i,end=" ")
student("samundar","rahul","rakul")
```

Output

```
the student are :
samundar rahul rakul
```

11.The Enumerate function

There are some situation in which we need the index value as well as the data of a given list ,in such situation we have to use two loop one for the index and another for the value in order to skip the loop creation we use enumerate function it will return the index as well as value.

input

```
li={"apple","mango","banana","litchi"}
for index,value in enumerate(li):
    print(f"The index is {index} and the value is {value}")
```

Output

```
The index is 0 and the value is litchi
The index is 1 and the value is banana
The index is 2 and the value is mango
The index is 3 and the value is apple
```

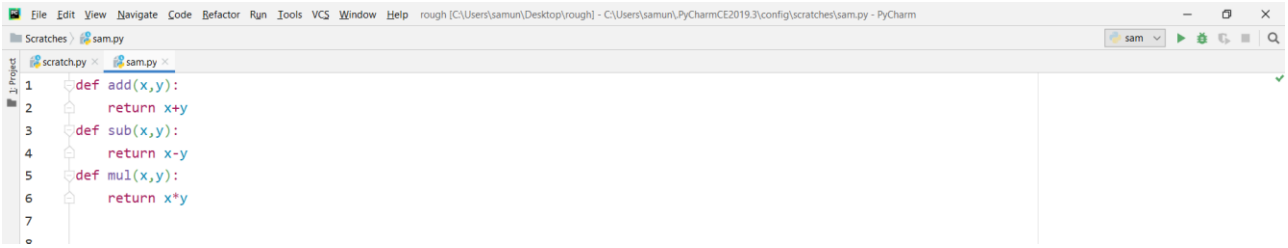
NOTE: In enumerate function the index is recorded from the end it means that the end element has index 0 and the first element has indexed last

12.Modules in python

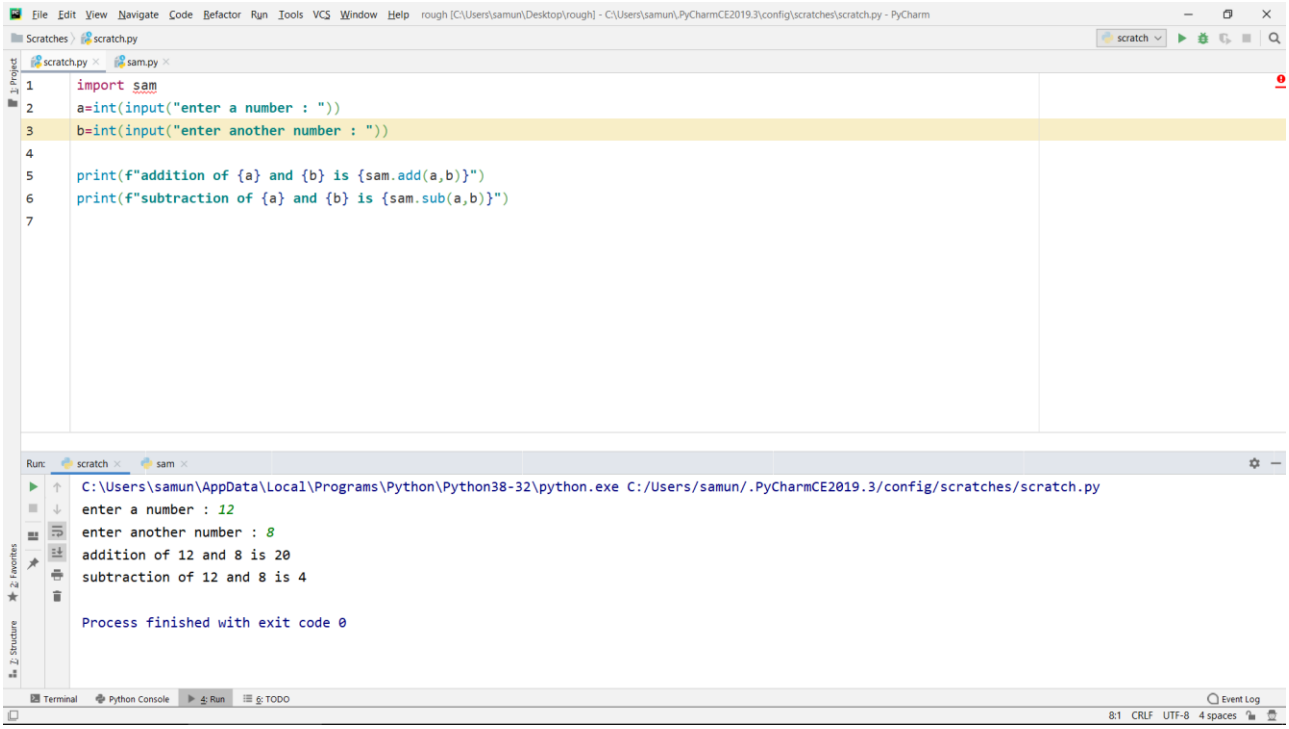
We can create our own modules (user-defined module) in python.

- First, we create a file name with the extension.
- Then we write **"import file name" in** file name mention the name of file from where function is imported
- we can access the function of the another file by using (.)dot e.g. module name. function name().

This is the file where module is defined:



This is the file where we accessed the module:



**V.imp

NOTE: There are situation that if there is another code except the function in the file then whenever we import the file then the program will execute all the lines of imported file first but we do not want that we only want to use a particular function so here we use this:

- Simply type main in pycharm and press enter it will write this code: **if__name__== '__main__'**
- we have to write code inside this block in order to avoid the above problem.
- This sentence will just an if block which execute only when you are in the same file in which you executing the program, if you are in another file then the code that is written inside these main block will not executed.

13.The Join function in python

There are various scenario in which we use the join function suppose we fetch data from data base and try to make in such a format which can be easily loaded in Hadoop and other database.in such case we can modify our fetched data and made it into a simple format.

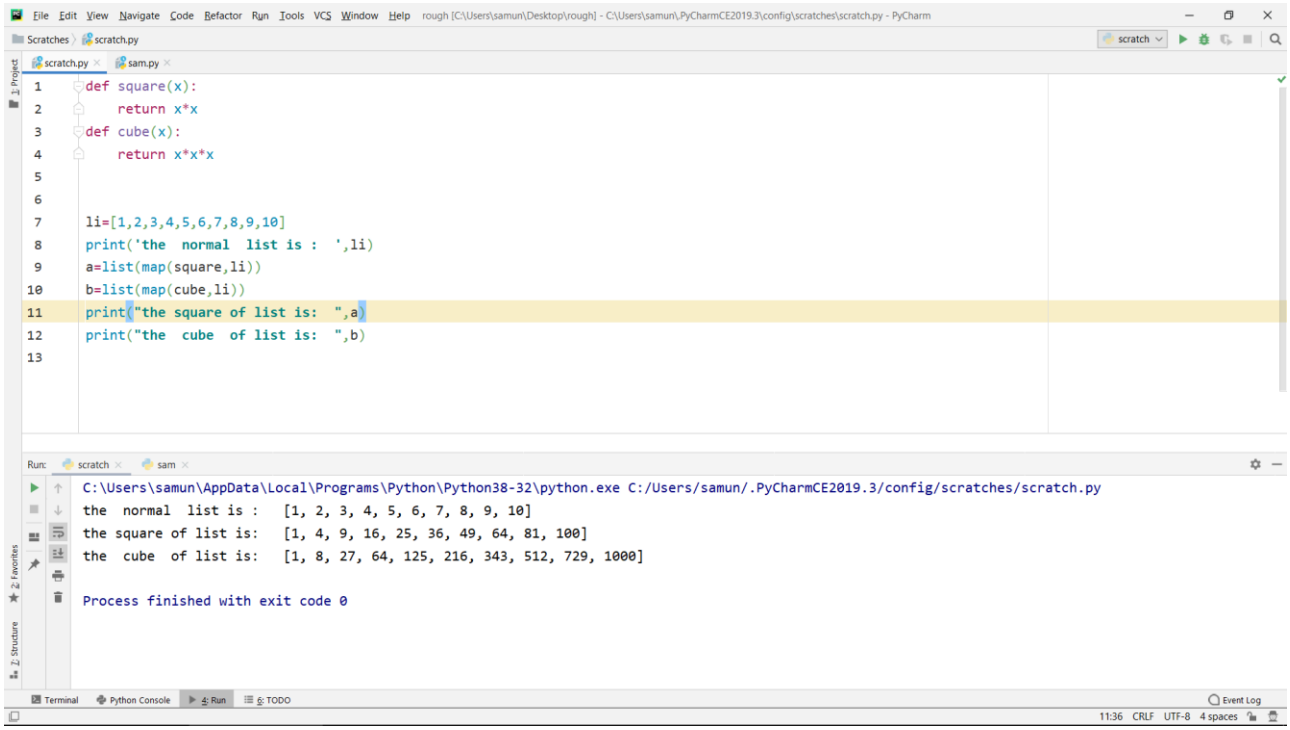
13.The map function in python

If there is a list and we want to modify each and every data of list then this can be done by map function.

Syntax: var = list(map(fun,list_name))

!warning: do not put braces here

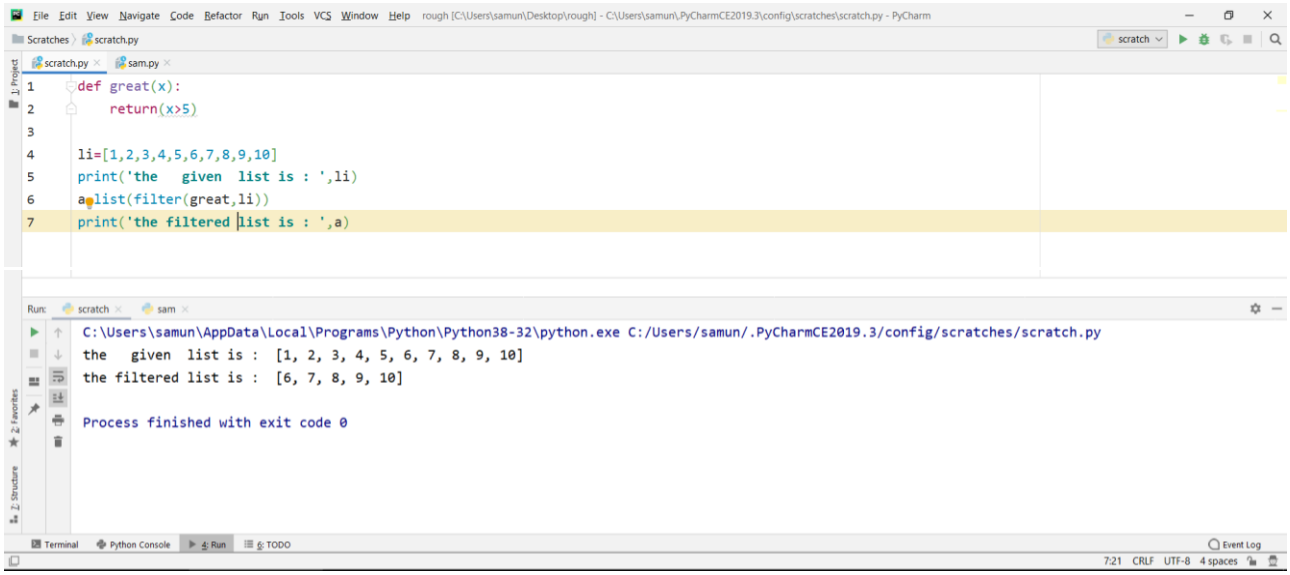
- First, we have to create a function which changes the data of the function note the function must take argument and also return the result.
- The map function put each data in that function and whatever the function return data stored in another list.
- The map function always returns object data type so we have to typecast it into the list format.
- The result must be stored in another variable (or list).it means it create a separate list with modified data.



13.The filter function in python

If there is a list and we want some filtered data from it then we use filter function. Except of returning value this function return true or false, so it can only filter the data not modified it.

- First, we have to create a function which Checks the data of the function, note the function must take argument and also return the result in form of true or false.
- The filter function put each data from the list and send it to the function where if the function retun true then only it keep the data and stored it into a new list.
- It cannot modify the data of the list



OOPS in Python:

1. Class in python

Creating a class in python helps in reusability of code. we can also define function inside the class and call it whenever needed.

- In class we can mention the structure or blueprint of the object and also help to set default value
- In class we can also create a number of functions which can be accessed by creating an object and calling that particular function
- The data that is defined as a default inside the class cannot be changed using the object, it can only be changed when we use class name
- If we add anything in the class using object or try to change then the changes only occur for that object only it cannot throw any error it just get added to that object.
- If we define any function inside the class the as a default parameter it pass a parameter ***self*** which helps to call the function by passing argument as the self object as show in this fig at line(5). We did not have to pass the object name as argument it automatically passed the object.

```
1  class student:
2      roll=0
3      name=""
4      std=0
5      def detail(self):
6          print(f"Roll.no   :{self.roll}\nName      :{self.name}\nclass     :{self.std} ")
7
8  sam=student()                          # creating object
9  rahul=student()                        # creating object
10
11 sam.name="samundar singh"
12 sam.roll=100
13 sam.std=10
14 print(sam.__dict__)                    #inbuilt function that print the data in a dictionary format
15 print("\n")
16 sam.detail()
```

```
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/samun/.PyCharmCE2019.3/config/scratches/scratch.py
{'name': 'samundar singh', 'roll': 100, 'std': 10}


Roll.no   :100
Name      :samundar singh
class     :10

Process finished with exit code 0
```

Accessing the function inside of the class:

```
1  class cal:
2      def add(x,y):
3          return x+y
4      def sub(x,y):
5          return x-y
6
7  a=int(input("enter a number : "))
8  b=int(input("enter another number : "))
9
10 print(f"sum of number {cal.add(a,b)}")          # direct access the function of class without object
11 print(f"subtract of number {cal.sub(a,b)}")     # direct access the function of class without object
```

```
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/samun/.PyCharmCE2019.3/config/scratches/sam.py
enter a number : 12
enter another number : 5
sum of number 17
subtract of number 7

Process finished with exit code 0
```

2. Constructor in python

A constructor is a special type of class method or function which is used to initialize the member of the class when an object of class is created.

- **it is automatically called whenever an object is created.**

  Syntax: ***def __init__(self,argument1,argument2…..)***

```
class employee:
    #creating constructor that initalize the data
    def __init__(self):
        id=int(input("enter your employee id : "))
        name=input("enter your name : ")
        address=input("enter your address : ")
        phone=int(input("enter your phone : "))

        self.name=name
        self.id=id
        self.address=address
        self.phone= phone

#data insrttion through creating single object
ram=employee()
print(ram.__dict__)

#data insertion by creating object through loop
for i in range(1,4):
    print(f"\t employee : {i}  through loop ")
    i=employee()
    print(i.__dict__)
```

output

```
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/sa
enter your employee id : 100
enter your name : samundar
enter your address : lohardaga
enter your phone : 89898989898
{'name': 'samundar', 'id': 100, 'address': 'lohardaga', 'phone': 89898989898}
     employee : 1  through loop
enter your employee id : 101
enter your name : rahul
enter your address : patna
enter your phone : 4567
{'name': 'rahul', 'id': 101, 'address': 'patna', 'phone': 4567}
     employee : 2  through loop
enter your employee id : 110
enter your name : johnny dep
enter your address : usa
enter your phone : 4564323
{'name': 'johnny dep', 'id': 110, 'address': 'usa', 'phone': 4564323}
     employee : 3  through loop
enter your employee id :
```

3. Changing class member data or property using object in python

The traditional way is that if we have to change the data of class then we can only change by means of class name but now by using **@classmethod property** we can change data by object also.

Syntax: @classmethod

objectname. function(cls, newdata)

By default this created

Mention object name

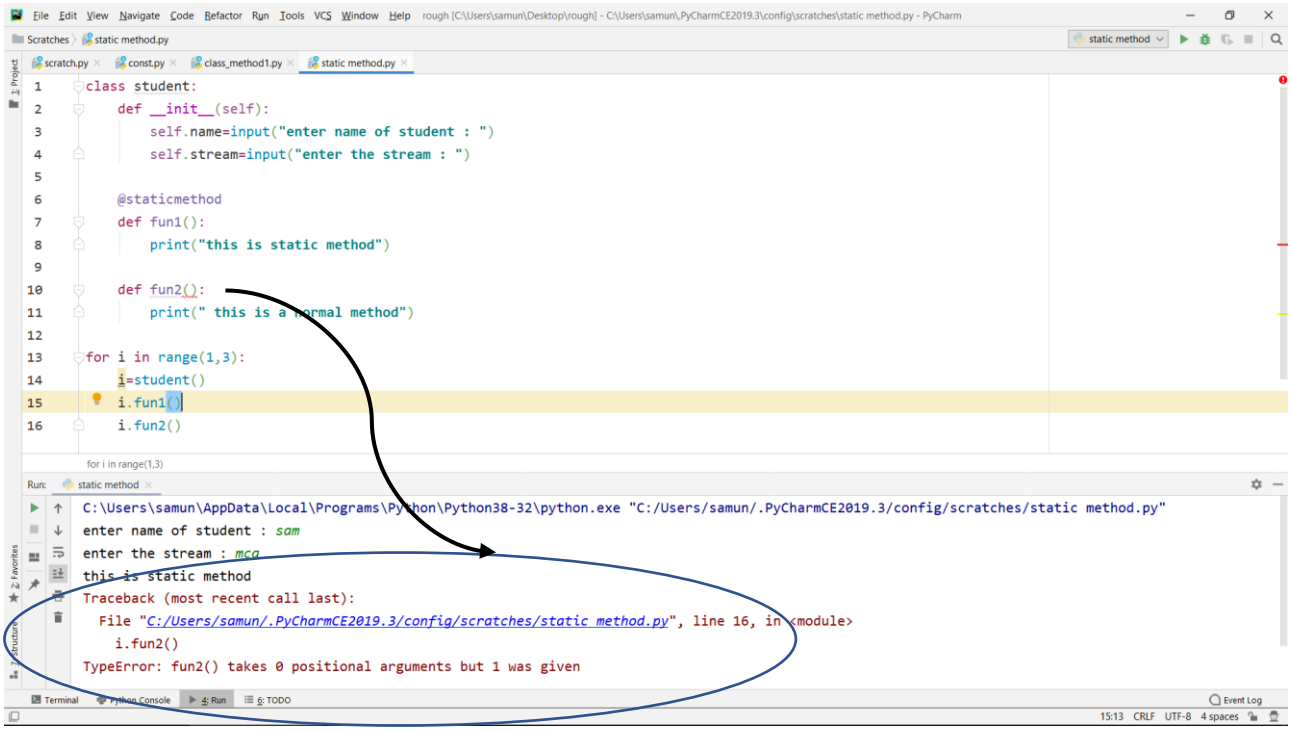
Mention function name that is written below the classmethod

```
class employee:
    basic=5000
    HRA=1000
    def __init__(self):
        self.basic=employee.basic
        self.HRA=employee.HRA
    @classmethod
    def changebaisc(cls,newbasic):
        cls.basic=newbasic

ram=employee()
print("ram salary detail : ",ram.__dict__)
print(f"\t\t\t\t the basic_pay in employee class before change is {employee.basic}")

ram.changebaisc(10000)      #using object we will change the data of the class successfully

mohan=employee()
print("mohan salary detail : ",mohan.__dict__)
print(f"\t\t\t\t the basic_pay in employee class After change is {employee.basic}")
```

```
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/samun/.PyCharmCE2019.3/config/scratches/class_method1.py
ram salary detail :  {'basic': 5000, 'HRA': 1000}
                 the basic_pay in employee class before change is 5000
mohan salary detail :  {'basic': 10000, 'HRA': 1000}
                 the basic_pay in employee class After change is 10000
```

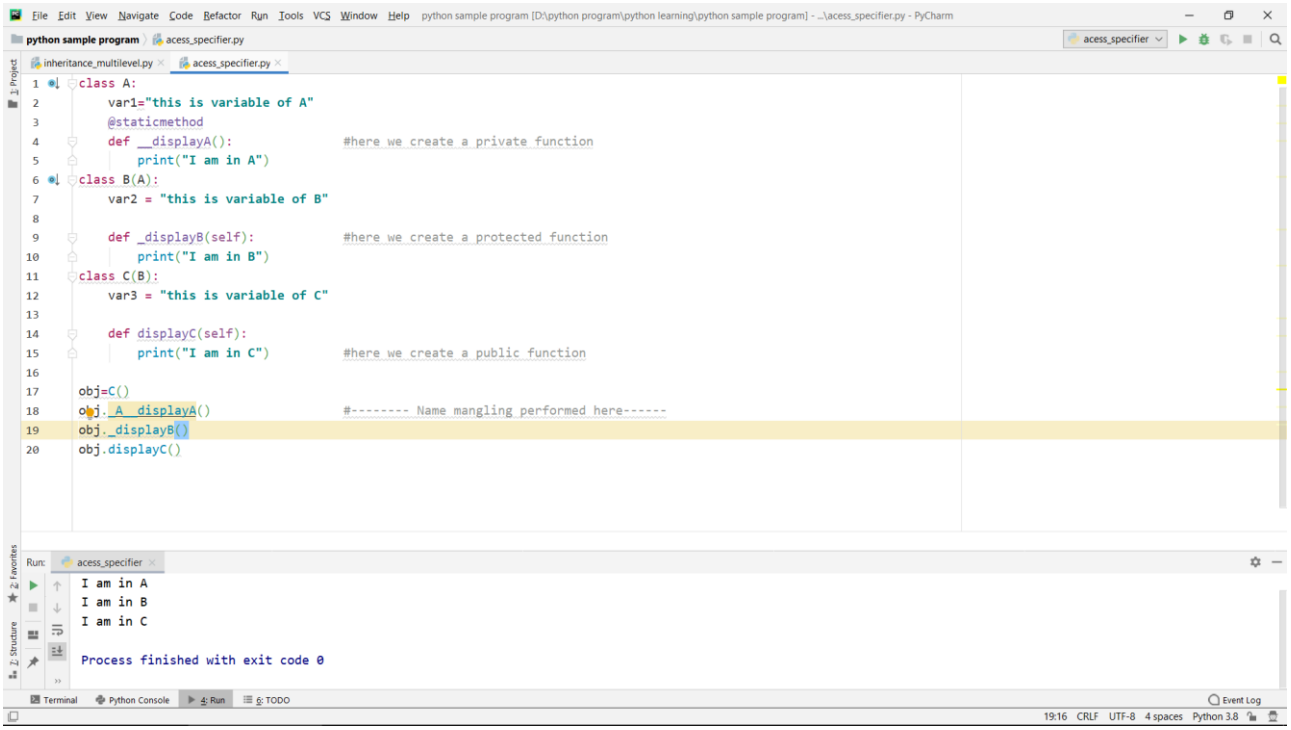
4. Use of static method in python

There are such situation in which we have to create a function inside the class but did not want the ***self*** argument although we can create the same function by neglecting the ***self*** argument but this is not efficient so we create a static method. The function that is defined below this method does not need any argument.



5. Access specifier in python

- **Public:** The data that is public can access by any anyone. There is no particular indication for it.
- **Protected:** The data that is protected can be accessed by the class objects and its child class. It is denoted by a single underscore(_func1) and called by placing the same single underscore(_func1).
- **Private:** The data that is private can only accessed by the class members or child class under some conditions. It is denoted by a double underscore(__func1) and called by placing a single underscore class name double underscore function or variable name(_A__func1).



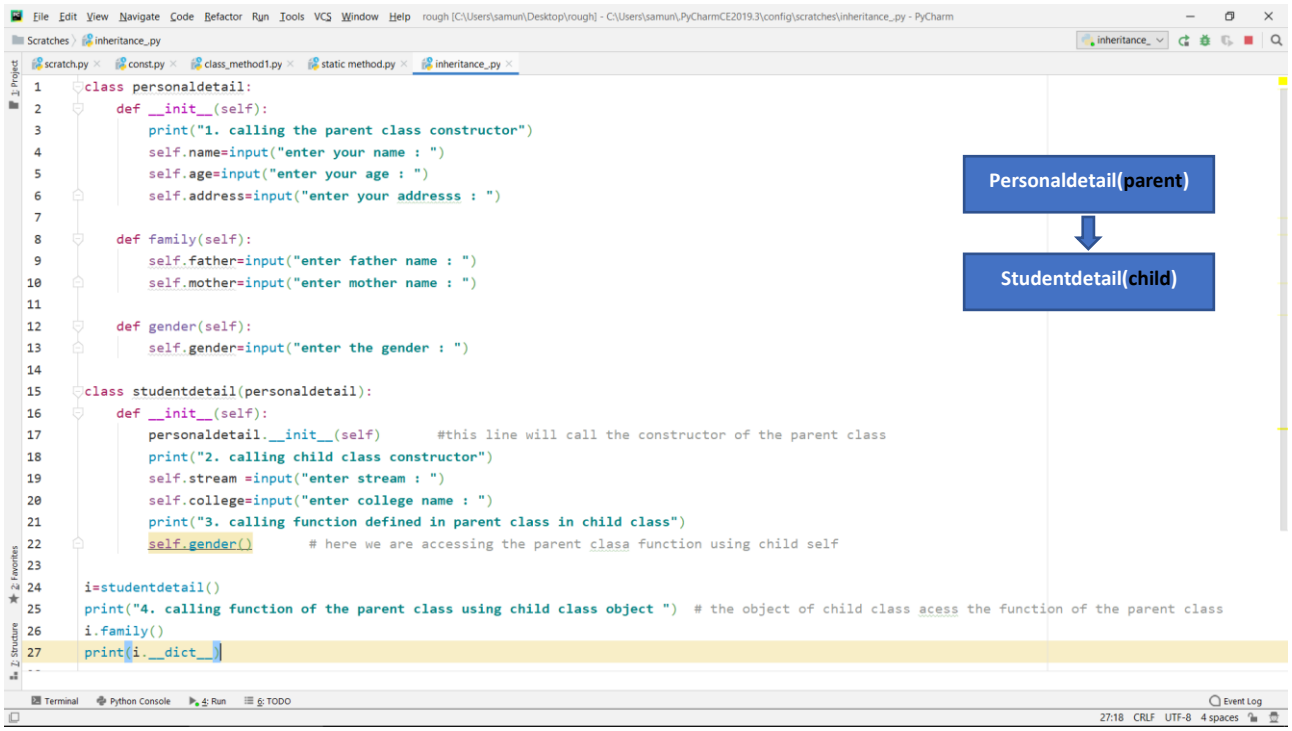
6. Inheritance in python

We can inherit the property of one class to another class using inheritance.

Inheritance are of many types:

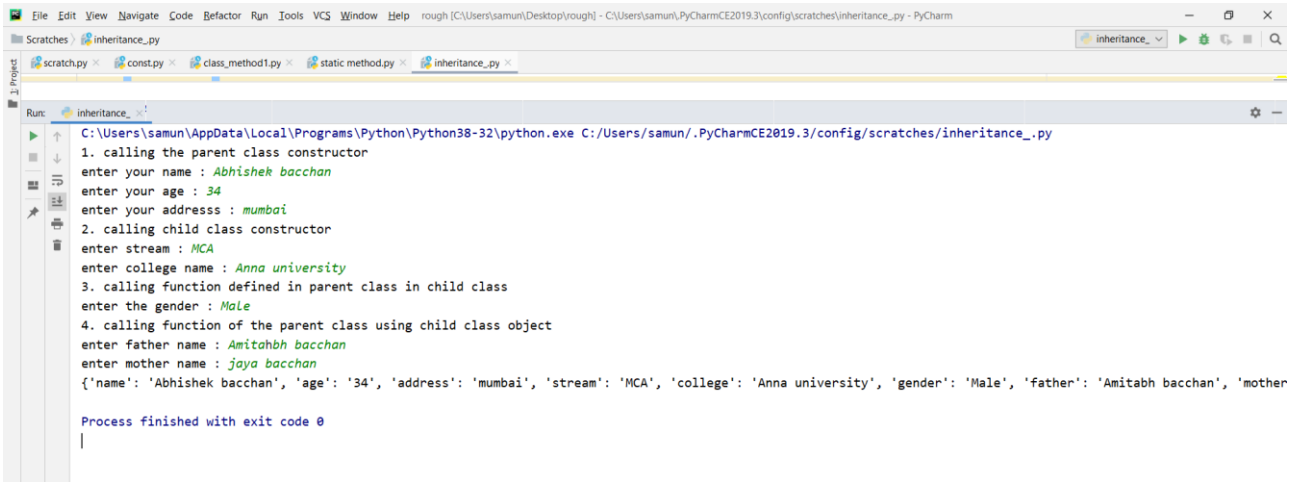
1. **Single level Inheritance:**
   - In single level inheritance there is one parent class and one child
   - The child can inherit with only one parent the child may have child but there is no grandparent relation.
   - We can also create the constructor of both the child and parent class and also access the function of the parent class with the help of child class object

Example to show single level inheritance:




```
class personaldetail:
    def __init__(self):
        print("1. calling the parent class constructor")
        self.name=input("enter your name : ")
        self.age=input("enter your age : ")
        self.address=input("enter your addresss : ")

    def family(self):
        self.father=input("enter father name : ")
        self.mother=input("enter mother name : ")

    def gender(self):
        self.gender=input("enter the gender : ")

class studentdetail(personaldetail):
    def __init__(self):
        personaldetail.__init__(self)       #this line will call the constructor of the parent class
        print("2. calling child class constructor")
        self.stream =input("enter stream : ")
        self.college=input("enter college name : ")
        print("3. calling function defined in parent class in child class")
        self.gender()       # here we are accessing the parent clasa function using child self

i=studentdetail()
print("4. calling function of the parent class using child class object ")  # the object of child class acess the function of the parent class
i.family()
print(i.__dict__)
```

Output:



```
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/samun/.PyCharmCE2019.3/config/scratches/inheritance_.py
1. calling the parent class constructor
enter your name : Abhishek bacchan
enter your age : 34
enter your addresss : mumbai
2. calling child class constructor
enter stream : MCA
enter college name : Anna university
3. calling function defined in parent class in child class
enter the gender : Male
4. calling function of the parent class using child class object
enter father name : Amitabh bacchan
enter mother name : jaya bacchan
{'name': 'Abhishek bacchan', 'age': '34', 'address': 'mumbai', 'stream': 'MCA', 'college': 'Anna university', 'gender': 'Male', 'father': 'Amitabh bacchan', 'mother

Process finished with exit code 0
```

2). **Multi-level Inheritance:**

- In multilevel inheritance we can access the grand parent data using child class object.
- In multilevel there is more than two class and the class are interconnected to each other.
- We can either access the parent class data in child class or we can call the function of the parent class by help of the child class object.

Syntax:

Class A:

Statements…….

Class B(A):

Statements…….

Class C(B):

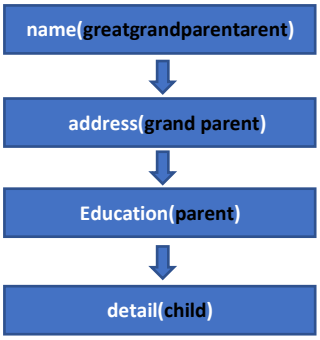
Statements…….

Class D(C):

Statements……..

Object=class D is created to access all the function of classes

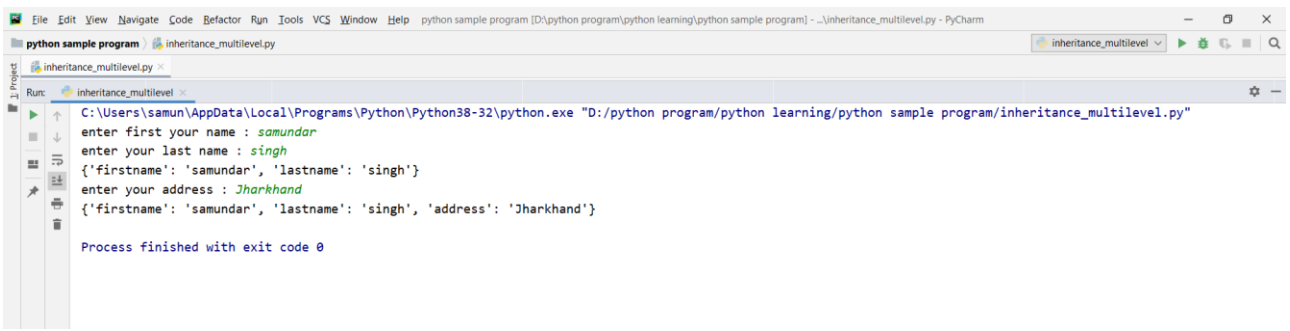
Example:
multilevel inheritance



Example to show Multi level inheritance:

```
class name:
    def names(self):
        self.firstname=input("enter first your name : ")
        self.lastname=input("enter your last name : ")
class address(name):
    def address(self):
        self.address=input("enter your address : ")

class education(address):
    def educationdetail(self):
        self.stream=input("enter your stream : ")

class detail(education):
    def __init__(self):
        self.names()            #acessinng a function of great grand parent using grandchild object


    def printdata(self):
        print(self.__dict__)

ram=detail()
ram.printdata()                 #Accessing the  child class function
ram.address()                   #accesing grand parent function using child object
ram.printdata()
```

Output:

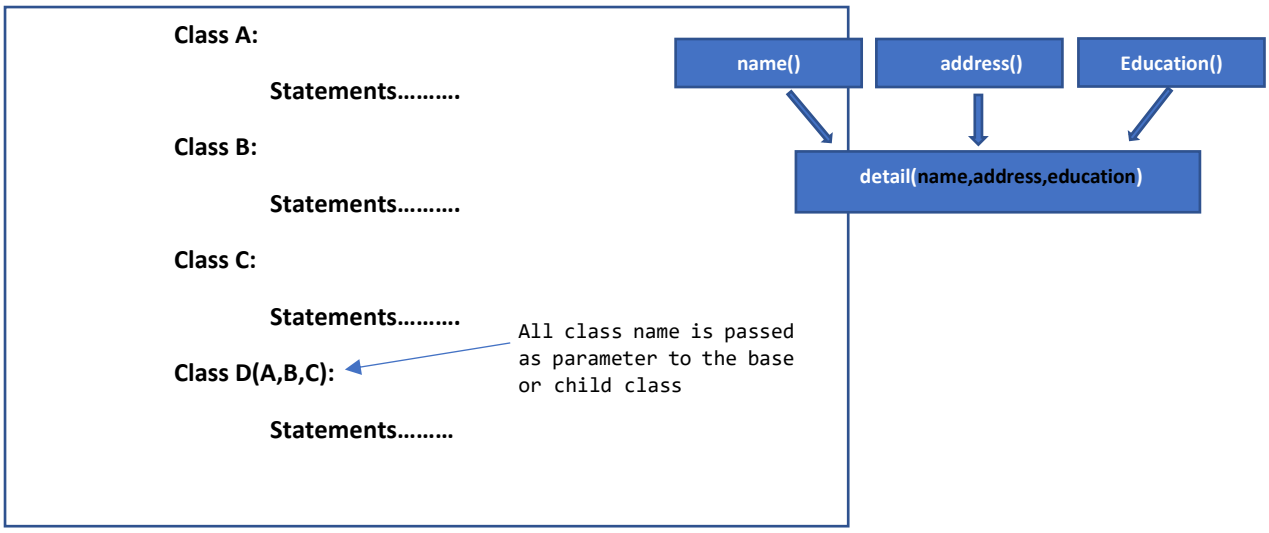


```
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe "D:/python program/python learning/python sample program/inheritance_multilevel.py"
enter first your name : samundar
enter your last name : singh
{'firstname': 'samundar', 'lastname': 'singh'}
enter your address : Jharkhand
{'firstname': 'samundar', 'lastname': 'singh', 'address': 'Jharkhand'}

Process finished with exit code 0
```

3). **Multiple Inheritance:**

- In multiple inheritance one class can inherit property of other class.
- There is no need that the classes are interconnected to each other.

Syntax:



Example to show Multiple inheritance:

```
class name:
    def names(self):
        self.firstname = input("enter first your name : ")
        self.lastname = input("enter your last name : ")

class address:
    def addressdetail(self):
        self.address = input("enter your address : ")

class education:
    def educationdetail(self):
        self.stream = input("enter your stream : ")

class detail(name, address, education):         # acessinng a function of other class by passing the class name in the argument
    def __init__(self):
        self.names()
        self.addressdetail()
        self.educationdetail()

    def printdata(self):
        print(self.__dict__)

ram = detail()
ram.printdata()
```

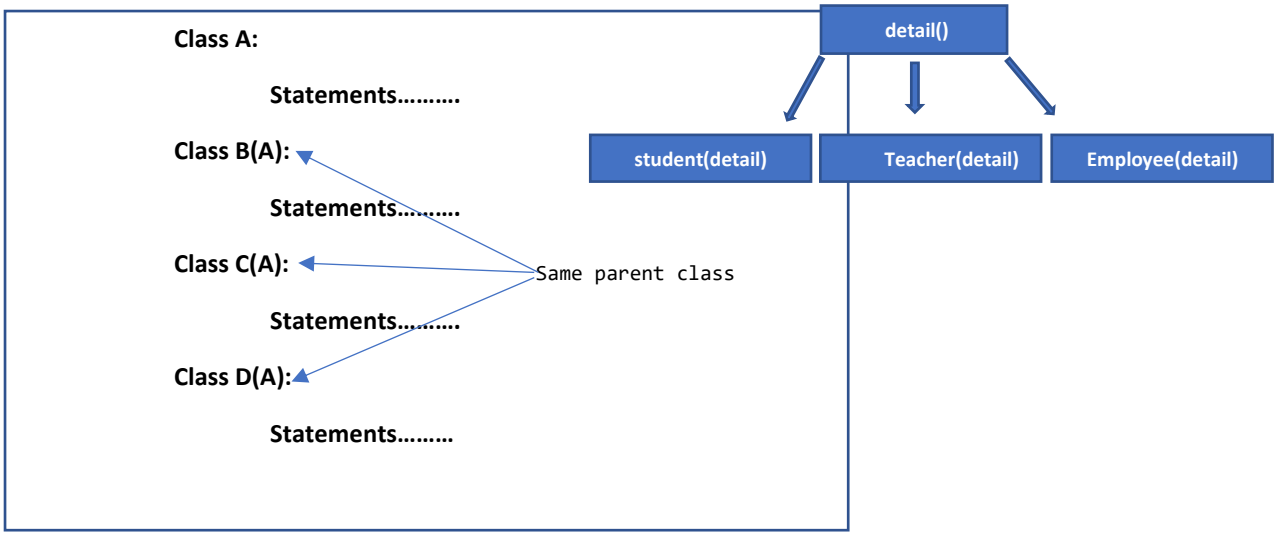
Output:

```
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe "D:/python program/python learning/python sample program/inheritance_multiple.py"
enter first your name : samundar
enter your last name : singh
enter your address : Jharkhand
enter your stream : Mca
{'firstname': 'samundar', 'lastname': 'singh', 'address': 'Jharkhand', 'stream': 'Mca'}

Process finished with exit code 0
```

2). **Hierarchical Inheritance:**

- In hierarchical inheritance there is only one parent class and multiple child class all the child class inherit property of the parent class.

Syntax:



Example to show Hierarchical inheritance:

```
class detail:
    def __init__(self):
        self.name=input("enter your name : ")
        self.age=input("enter your age : ")
        self.address=input("enter your address : ")
# the below classes have same parent that is detail
class student(detail):
    def printstuddent(self):
        print("student data : ",self.__dict__)
class teacher(detail):
    def printteacher(self):
        print("teacher data : ",self.__dict__)
class employee(detail):
    def printemployee(self):
        print("employee data : ",self.__dict__)


print(" select option : \n  1.student    \n  2.teacher   \n  3.employee ")
choice=int(input("enter your choice to insert the data"))

if (choice==1):
    obj=student()
    obj.printstuddent()
elif(choice==2):
    obj = teacher()
    obj.printteacher()
elif(choice==3):
    obj = employee()
    obj.printemployee()
else:
    print("invalid choice")
```

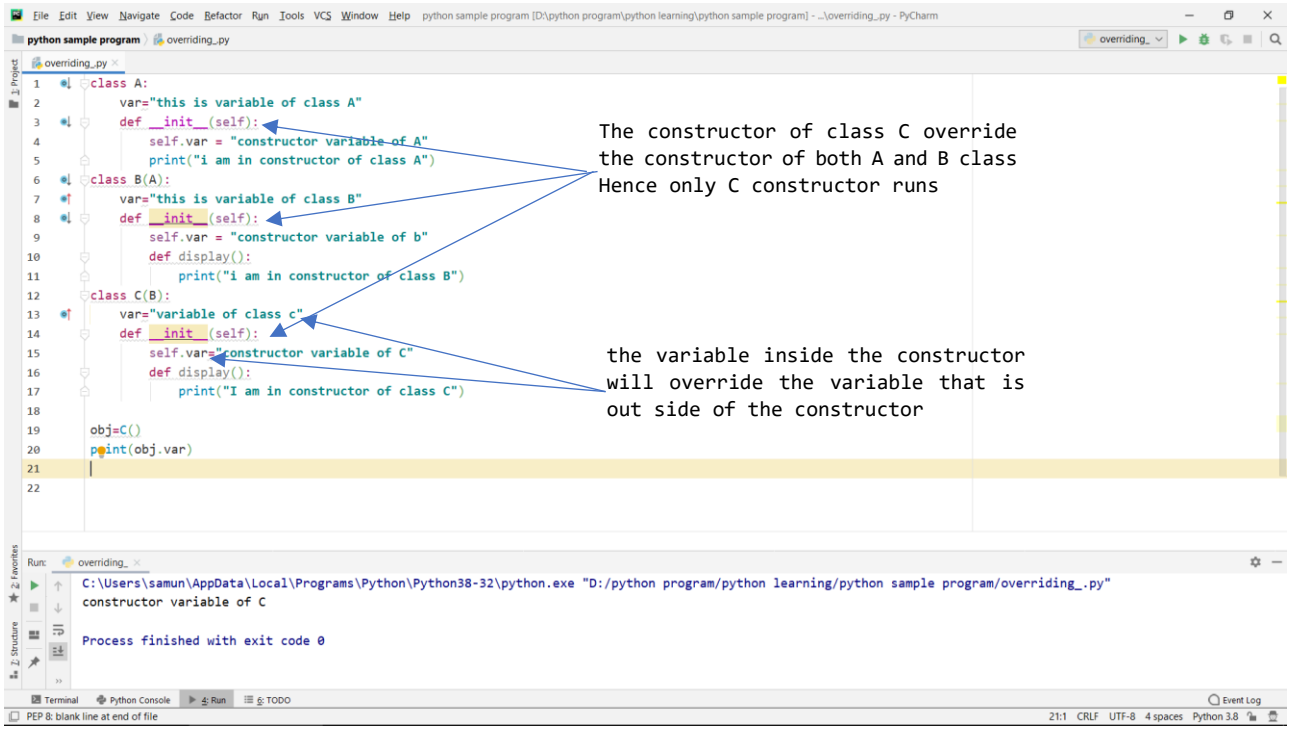
Output:

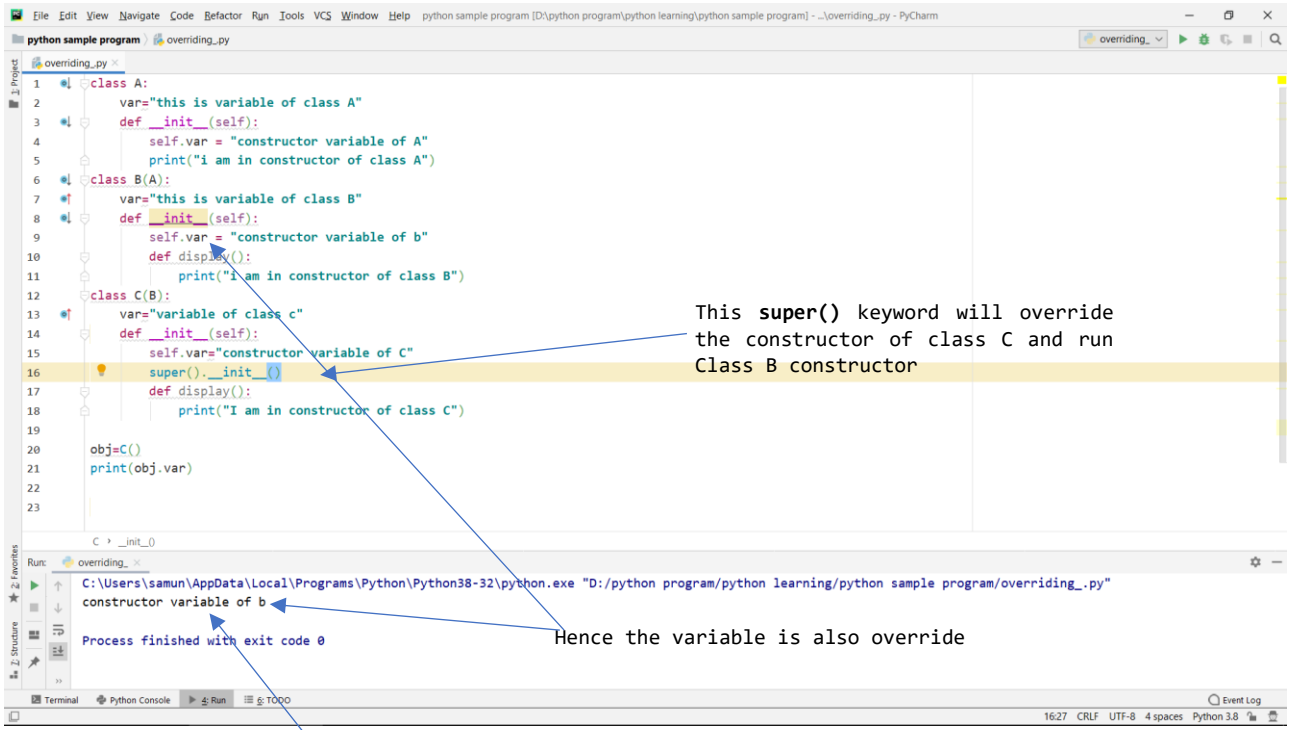
```
C:\Users\samun\AppData\Local\Programs\Python\Python38-32\python.exe "D:/python program/python learning/python sample program/inheritance_hierarchy.py"
 select option :
  1.student
  2.teacher
  3.employee
enter your choice to insert the data 2
enter your name : sanjay pandey
enter your age : 45
enter your address : Patna
teacher data :  {'name': 'sanjay pandey', 'age': '45', 'address': 'Patna'}

Process finished with exit code 0
```

7. Method overriding and super() keyword

Before starting this first we have to understand the flow of execution of the class.

When we create a class and perform inheritance on it then whenever the object of child class is created than the constructor of the child class is first executed and if any variable or method is called through the object than first it find the variable in the constructor then outside of constructor and after then its parent class…and so on





When the object of child class created then the constructor of the child class is executed if the requested variable is inside the constructor then it simply assign that value at the called object but later we invoked super() key thus the value of var is override by the variable of it parent class that's why instead of printing the variable of child class, it will print the value of parent class.

8. Function Caching

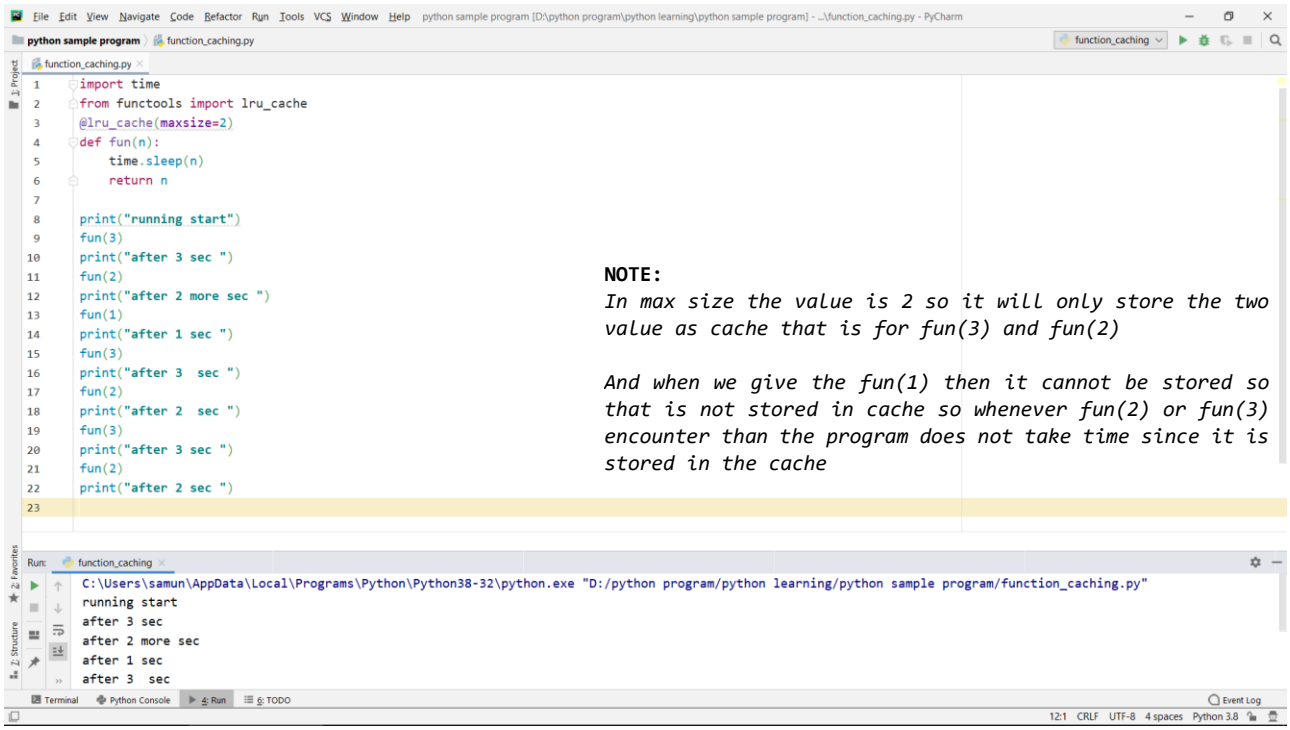
There are such situation in which a function will require time to process the data for example if we want some work related to database then it is not ideal to open the database again and again because it consume time suppose if a function take 3 sec to open a database and we call that function in our program 5 times than the execution time will take 3 x 5=15 sec to complete the task so there is a special method in python which will reduce this effort.

To achieve the function caching we have to import the lru_cache from the functools this lru_cache is a decorator which accept maxsize as a argument in which we have to mention that how much this will cache the data of function with different argument.

Syntax:

```
from functools import lru_cache
@lru_cache(maxsize=2)
def fun(n):
    statement…….
    return n
```

Note: in max size we have to mention that for how many various cases the function will store the data



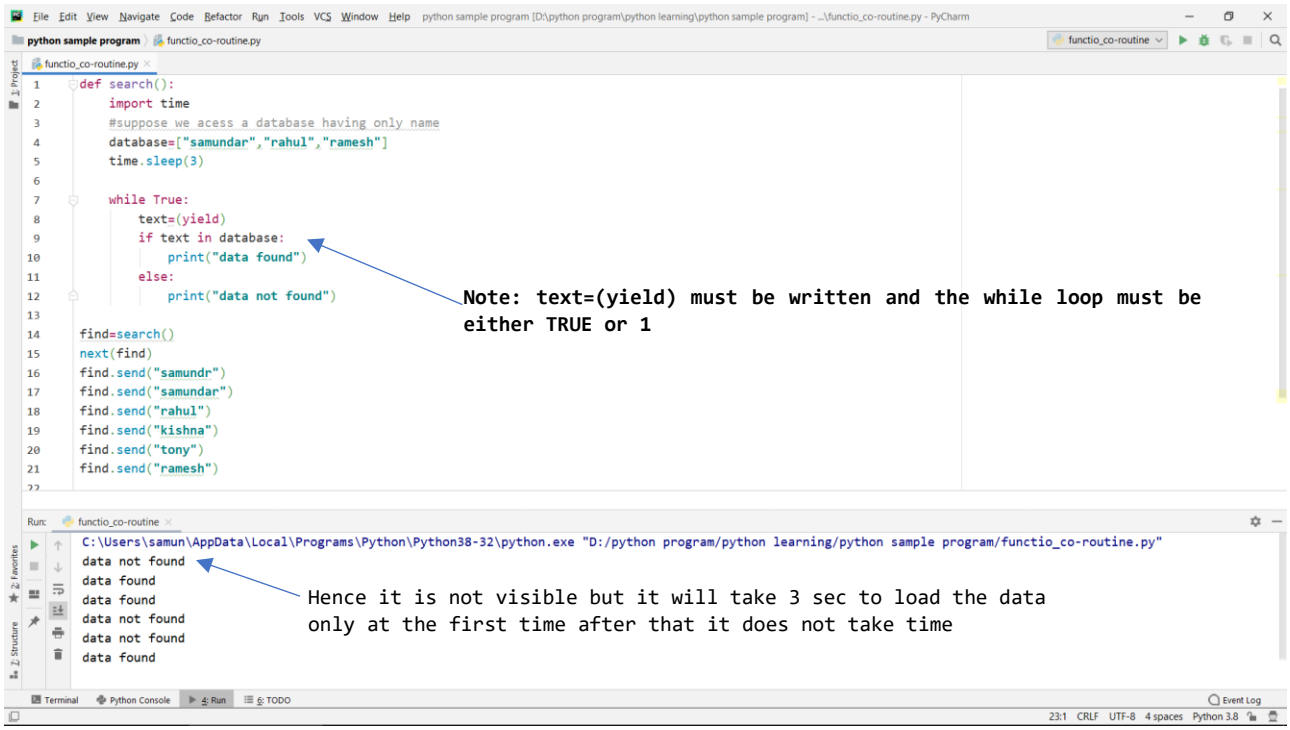
9. Function Co-routine

There are situations in which a function will take time to complete it execution and we require that when we call that function again than instead of executing the whole function we execute some part of the function. That time we use co-routine.

For example if we have to accesses a database and we search a particular data in table and we create a function that access a database and search a data in table if we want to again search a data than the same table then same process executed means again we have to access the database and start search table in database which is not ideal

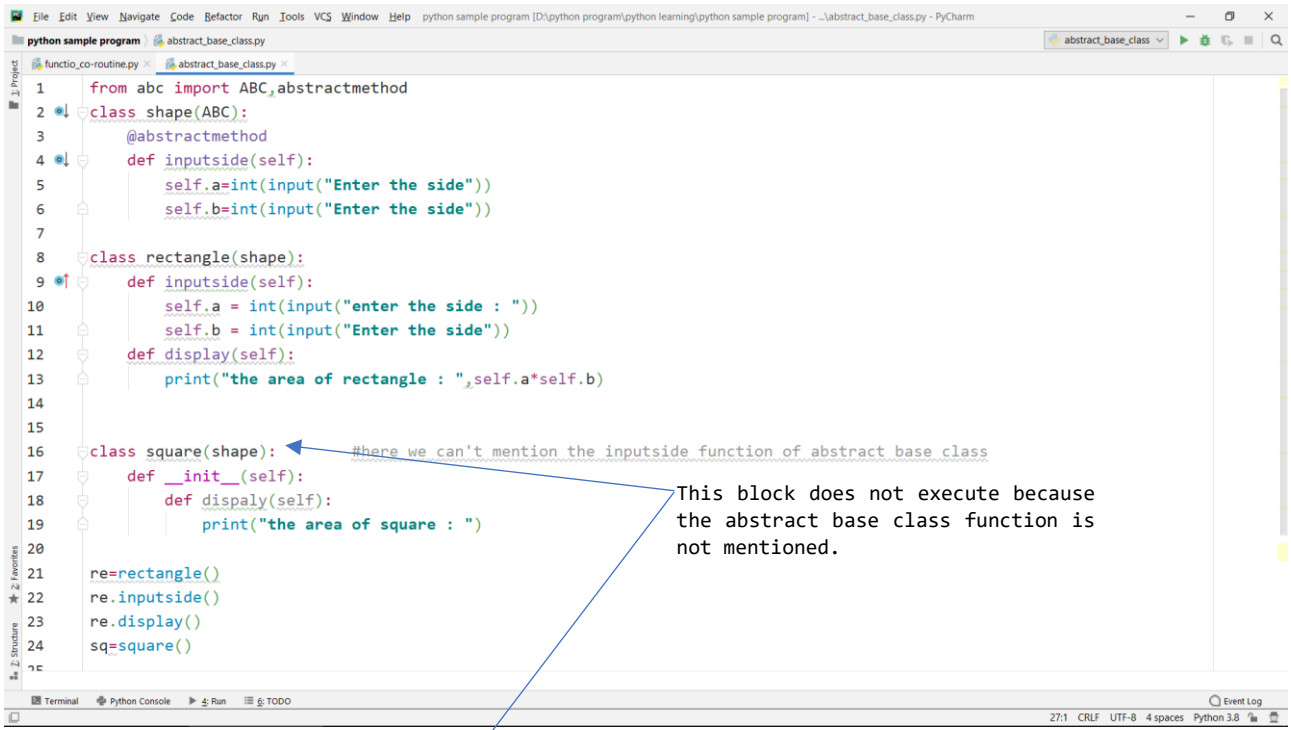
Here co-routine helps, the ideal approach is that if we access the database and table only once and search the data whenever needed

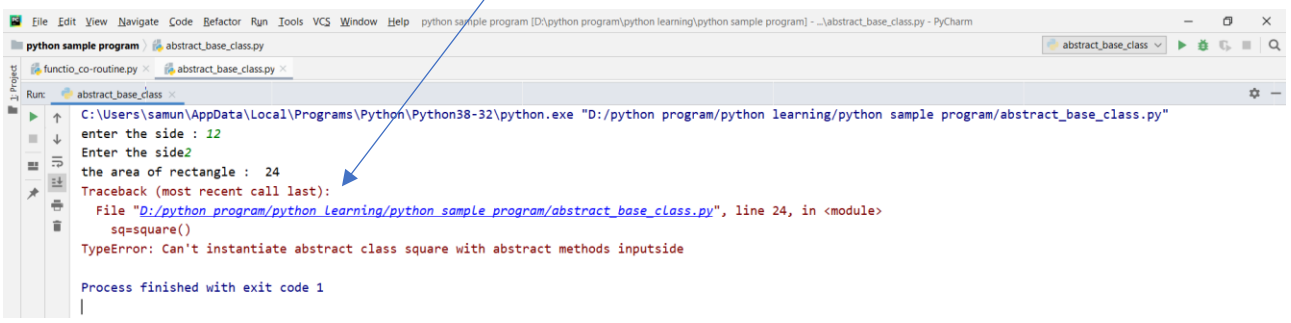
Here the scenario is represented by means of time module. Here we create database and a sleep method to give it a real feel of fetching data from the database.



10. Abstract base class in python

This is the class that enforces the child class to have the same function that is in the base class. If the function is not mentioned in the child class then it will show some error.
To achieve abstract base class we have to import ABCmeta from abc class.





11. OS module in python

It is a built in module of python and the various methods are:

1.) **Rename():** To change the name of the file we use rename() method:
Syntax: os.rename(current_file_name, new_file_name)
2.) **Remove():** To remove the file we use remove() method:
Syntax: `os.remove(file_name)`
3.) **mkdir():** To create folder we use mkdir() method:
Syntax: `os.mkdir("newdir")`
4.) **chdir():** To change the current directory we use chdir() method:
Syntax: `os.chdir("newdir")`
5.) **getcwd():** To get current working directory we use getcwd() method:
Syntax: os.getcwd()

This module is helpful when we create any app and that app can create a folder and store data in it or to search a particular file in a particular location.

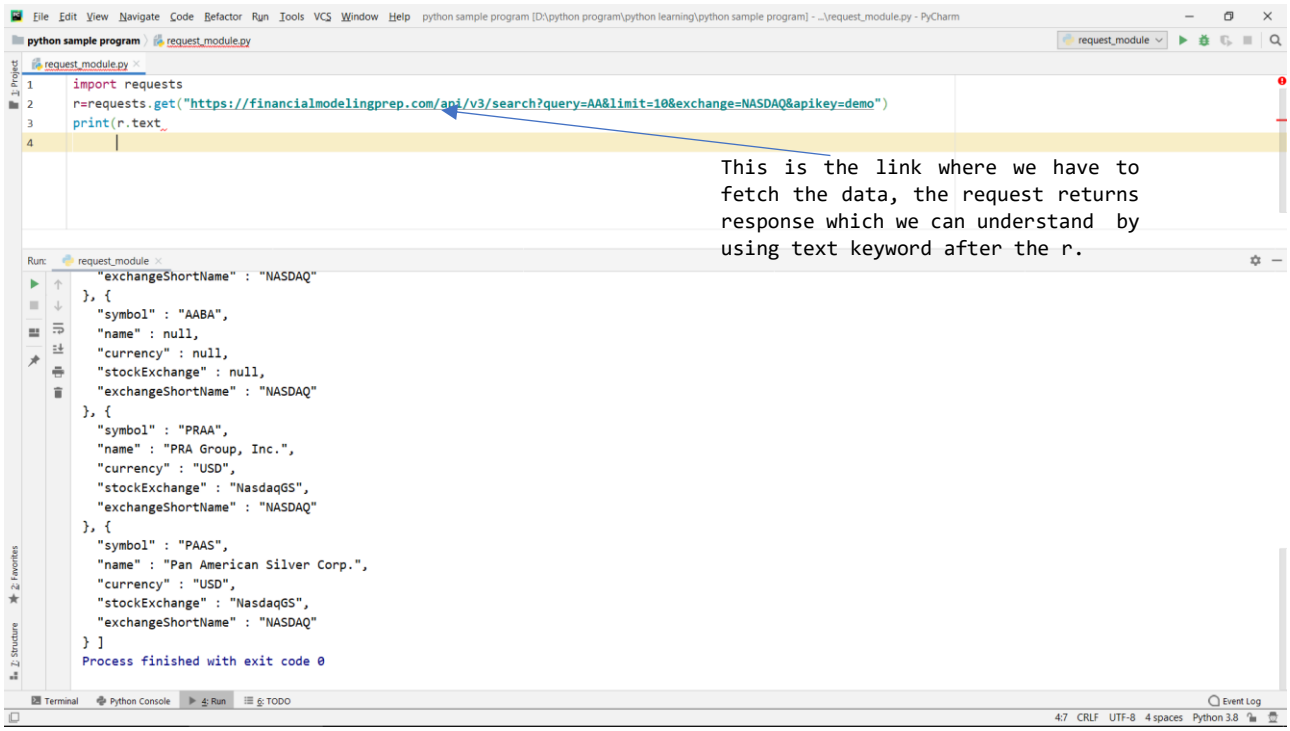
12. request module in python

When we create any app that will communicate with any website than in order to establish communication between the app and the website, we use request method

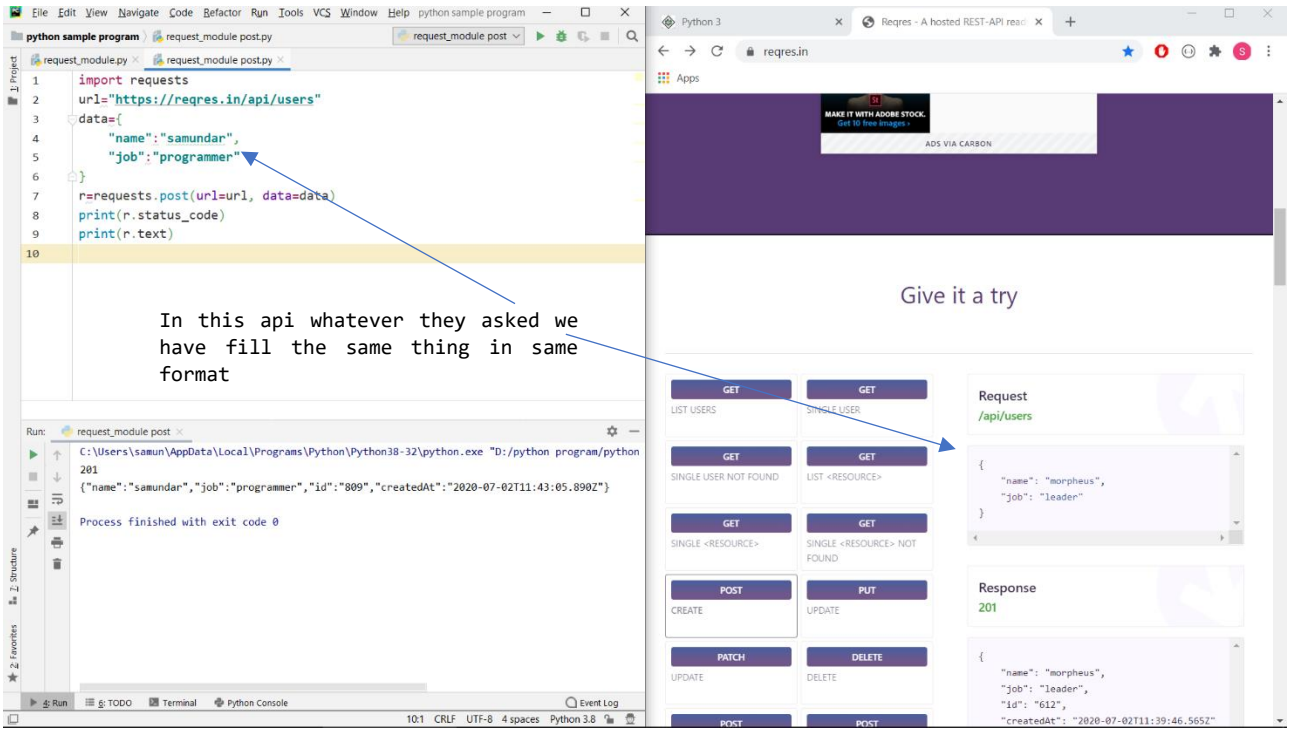
Note: first install the request method using the command : ***pip install requests***

Here by using the get method we are going to hit an api which is already created we get that API from here:

This site have In-built API :- Financial Modeling Prep API Documentation



For post method we have to use a dictionary to send the data, here again we use a API to perform the post method in post method of request two argument are given that is the url where data have to be sent and the data



13. JSON module in python

There is an in-built module n python that support the json encoding and decoding function.

Some of the important method of JSON module are as:

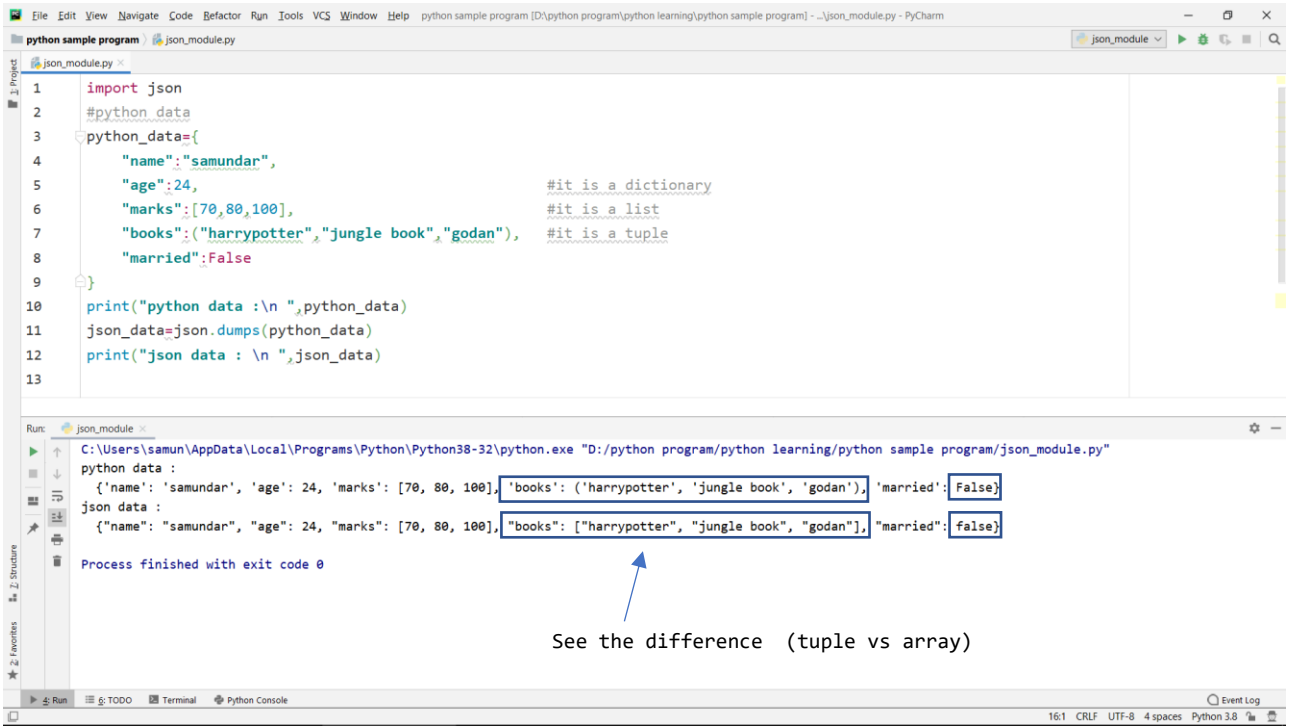
1. **Python to JSON (encoding) using dumps:**

   The following python object converted to json object as given below:

Python	JSON
dict	Object
list	Array
unicode	String
number - int, long	number – int
float	number – real
True	True
False	False
None	Null

Python object is converted to JSON object with the help of **dumps()** method . **dumps()** method converts dictionary object of python into JSON string data format.The converted data cannot be written to any file. If we want to write the converted data into a .json file then we use **dump()** function.

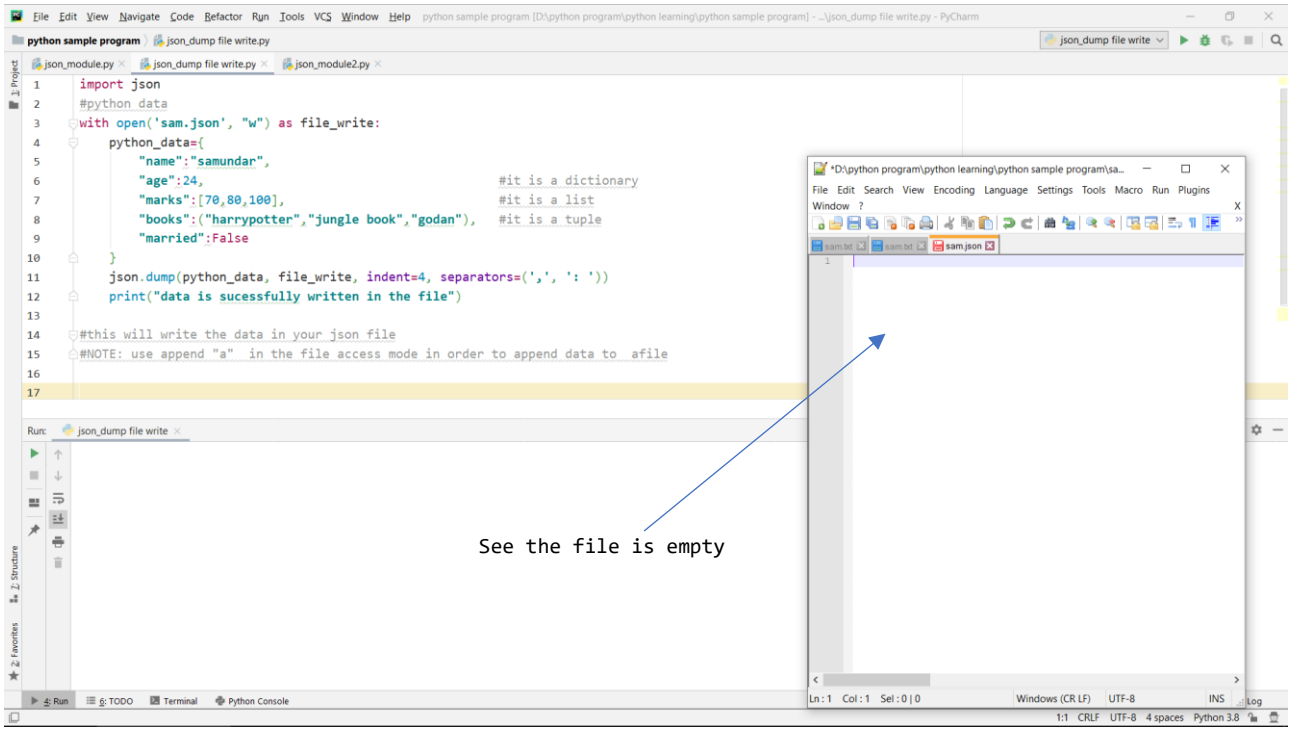
Program to convert python object to json object (using dumps):

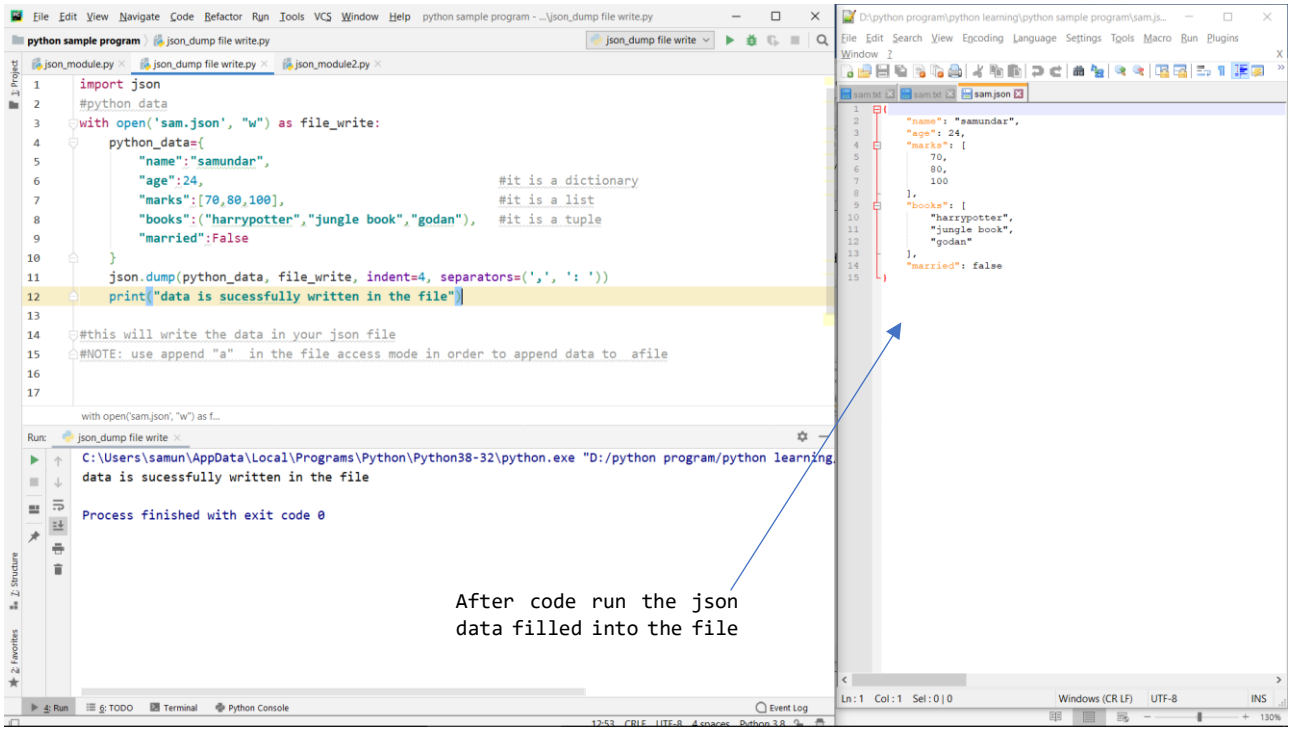


If we want to write the above data into a json file then we use dump() method show in this program below:

Syntax: **import json**
With open("your_json_filename.json", " Access mode"**) as ANY_NAME**
Python_dictionary_name={ ……..

---}
Json.dump(Python_dictionary_name**,** ANY_NAME,indent=4,seperator(' , ' , ' : ') **)**





NOTE: **1.** The **indent = 4** and **separator = (" , " , " : ")** parameter is just to show the data in pretty format
2. In order to append the data we use **"a "** append method in access mode

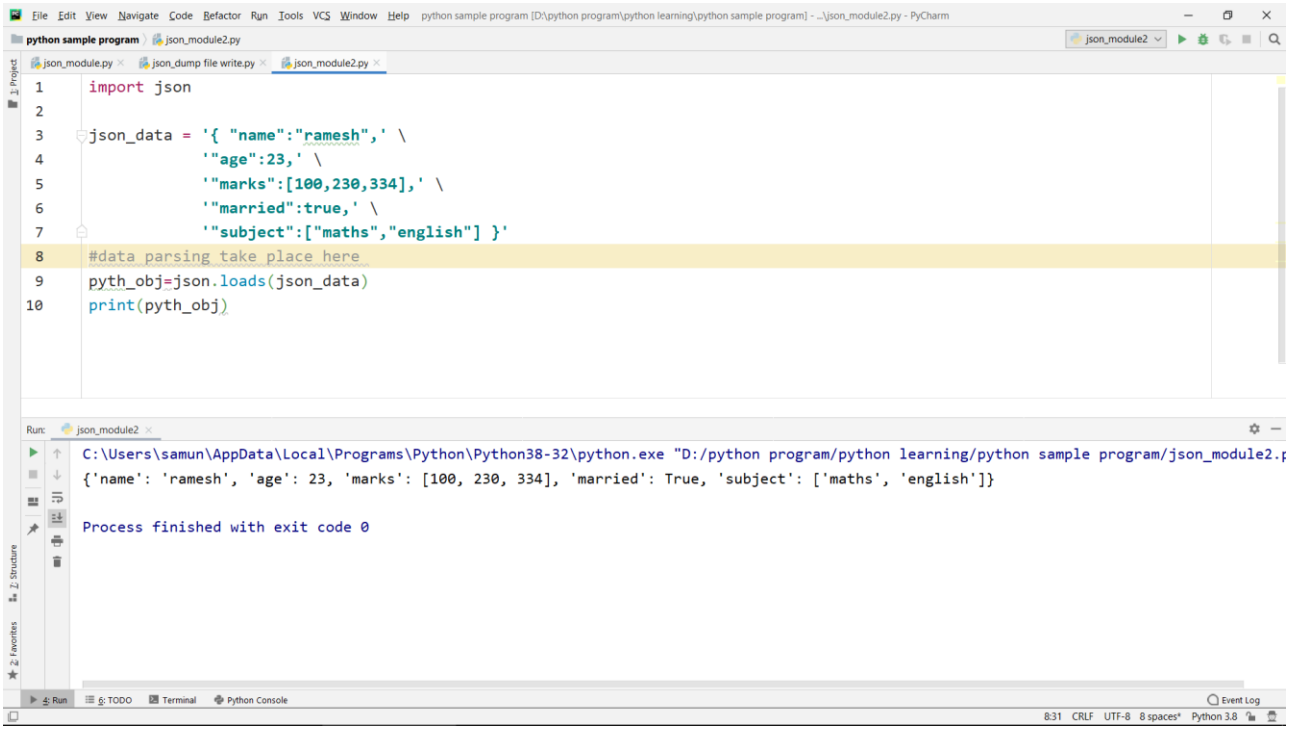
2.) JSON to Python (decoding or parsing):

The following JSON object converted to python object as given below

JSON	Python
Object	dict
Array	list
String	unicode
number – int	number - int, long
number – real	float
True	True
False	False
Null	None

JSON string decoding is done with the help of inbuilt method **loads()** & **load()** of JSON library in Python.
load() is used parse json data to python data it cannot write any data to the file

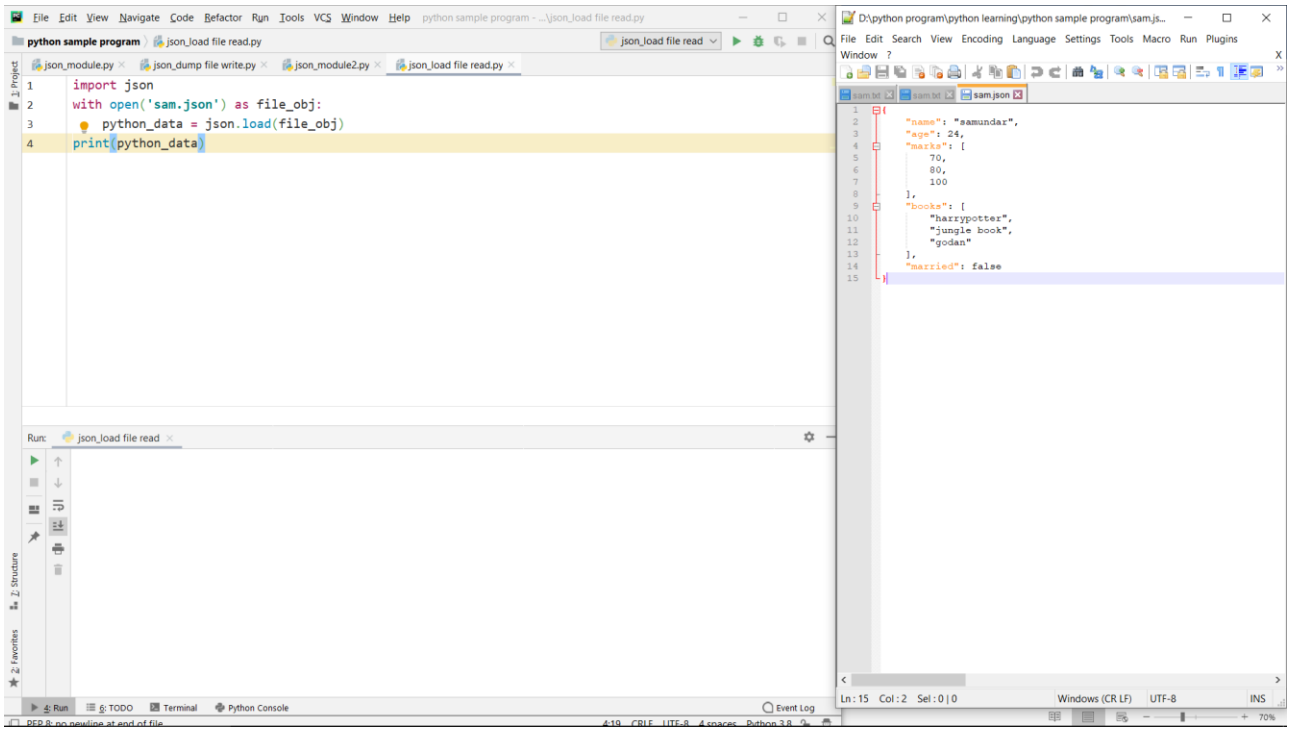
program to convert python object to json object (using dumps):

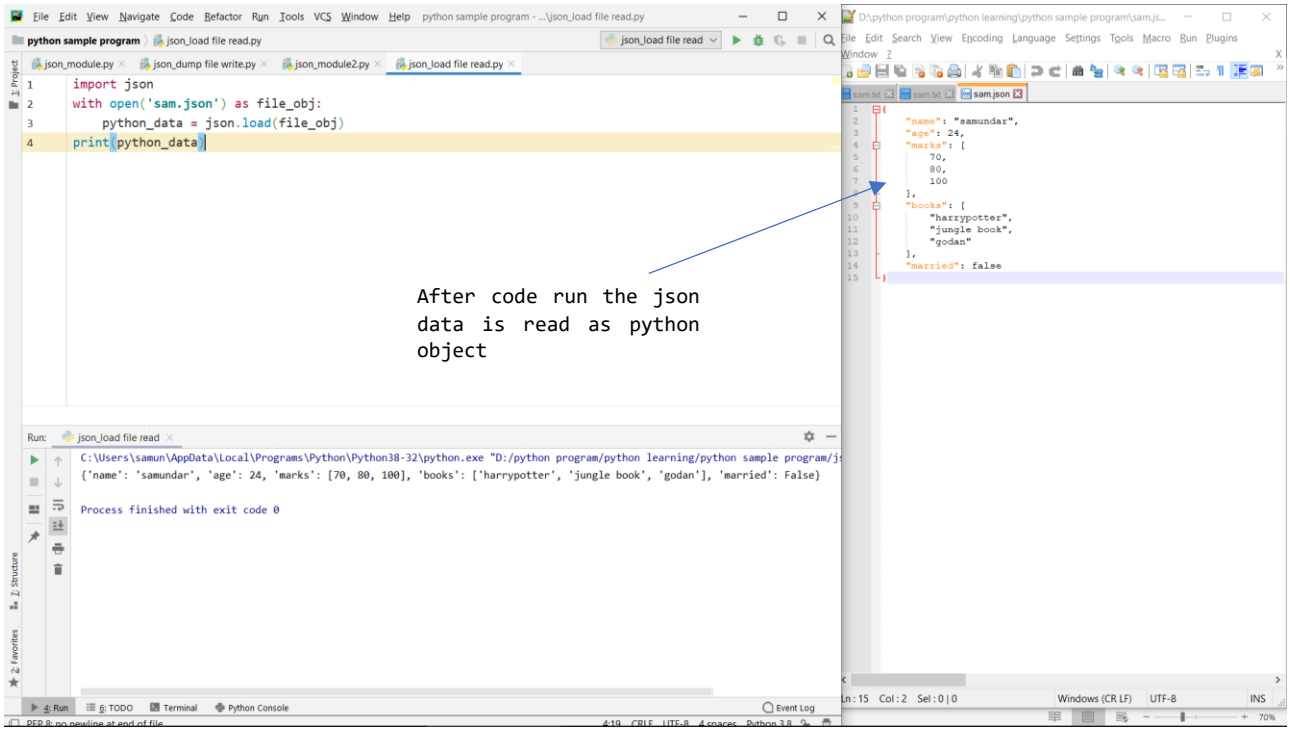


Note: 1. Whatever data is given as json data must be given as string.
2.Since the data is in json format it means it is either in key value pair or array so it can be converted to dictionary or list.

PYTHON PROGRAMMING

If we want to read data from a json file then we use load() method show in this program below:





TO BE CONTINUE……