Task-5

**Task:-5**

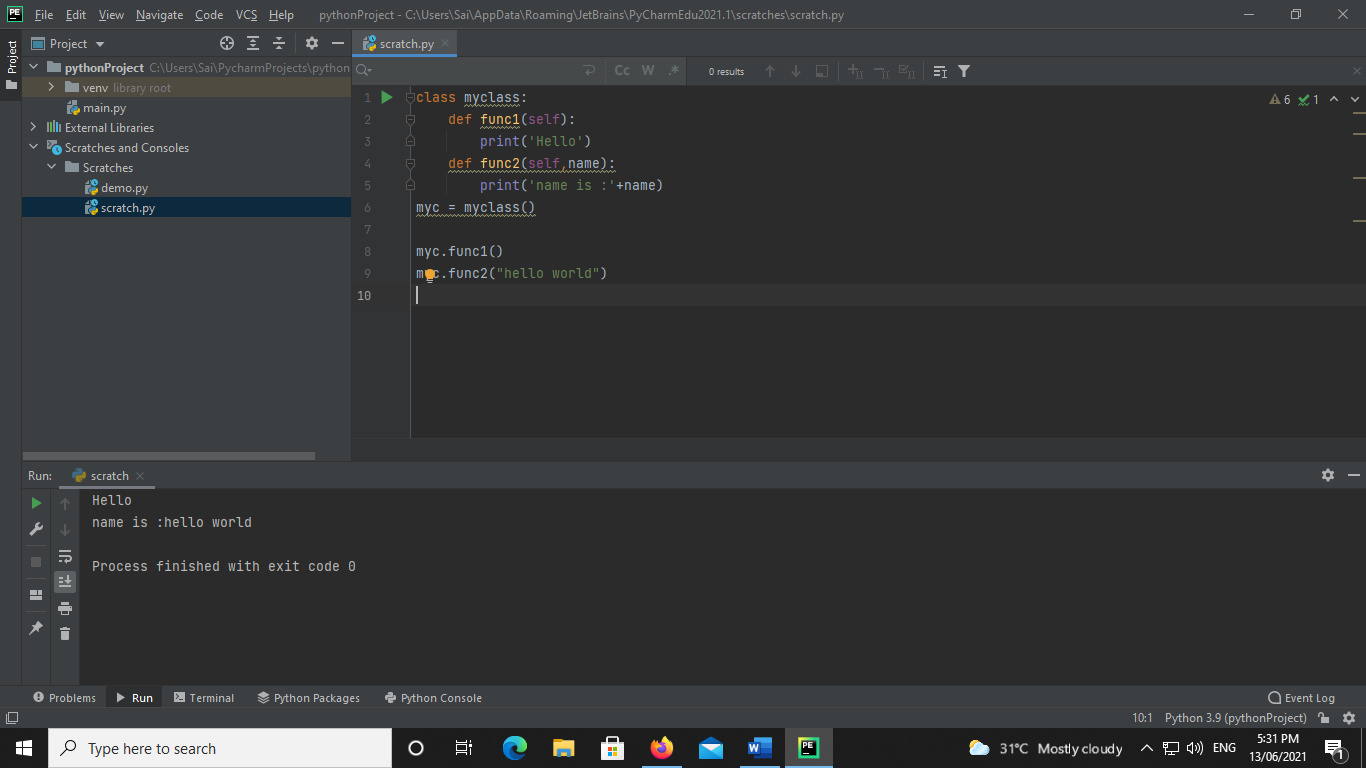
**Class:**

* A class is blueprint for the object.
* **Syntax:**

Class Myclass:

class myclass:  
 def func1(self):  
 print('Hello')  
 def func2(self,name):  
 print('name is :'+name)  
myc = myclass()  
  
myc.func1()  
myc.func2("hello world")

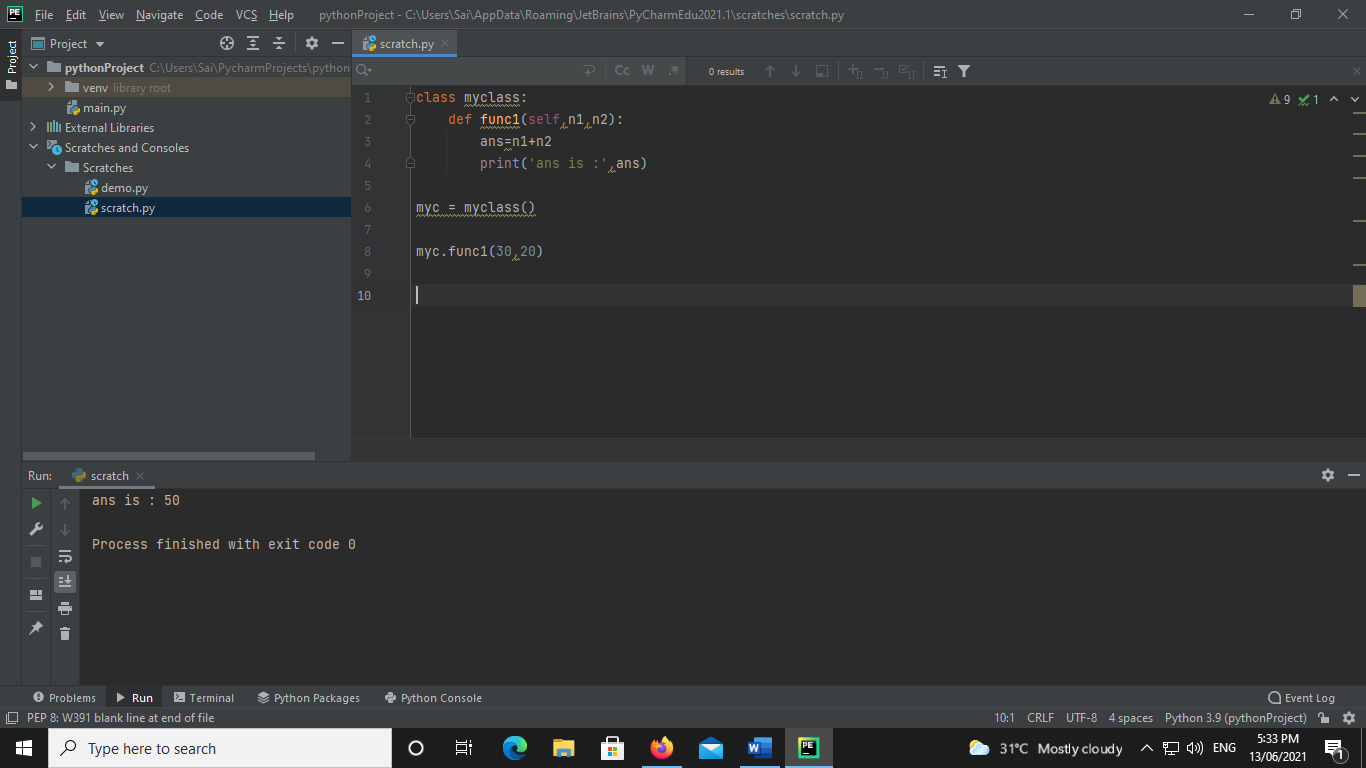
**Output:**

****

**Sum of 2 number using class:-**

**Code:**

class myclass:  
 def func1(self,n1,n2):  
 ans=n1+n2  
 print('ans is :',ans)  
  
myc = myclass()  
  
myc.func1(30,20)

****

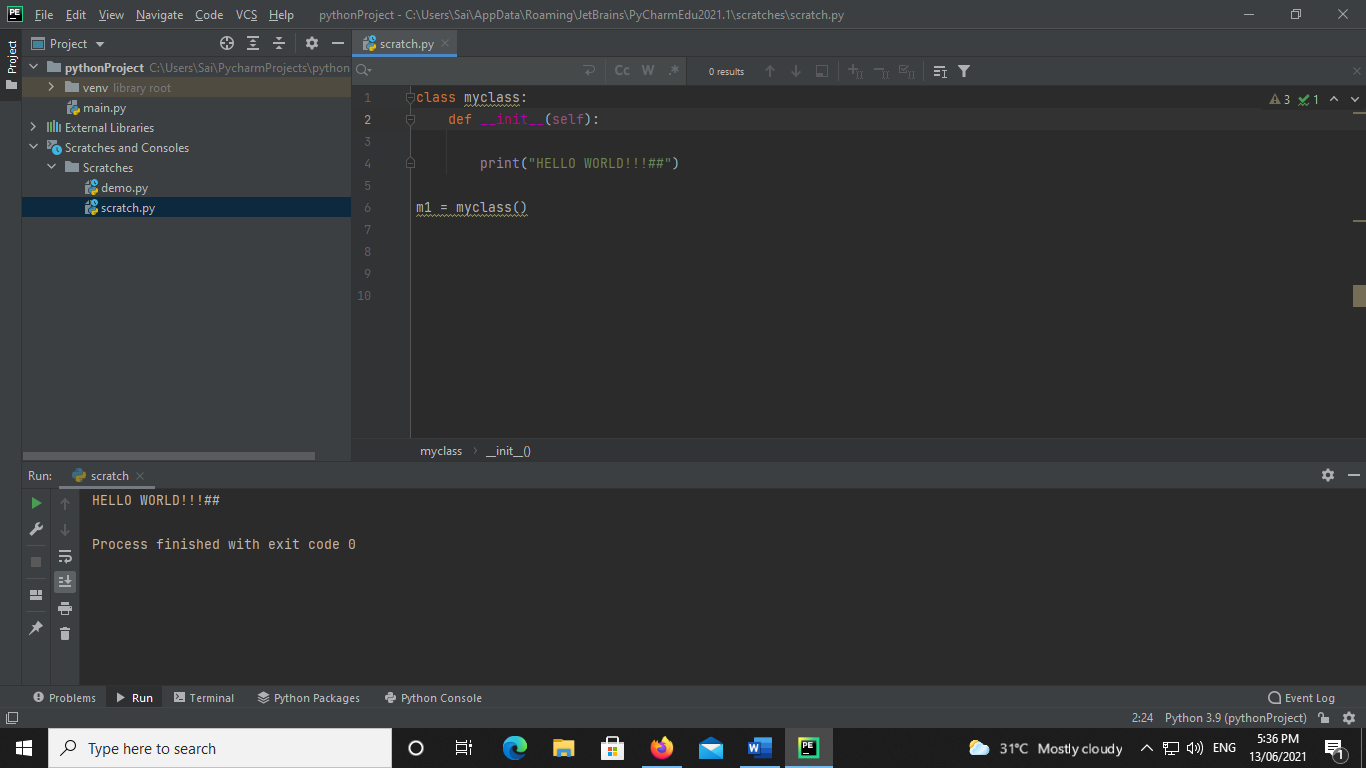
**Python Constructors**

* **Default constructor**

**Code:**

class myclass:  
 def \_\_init\_\_(self):  
  
 print("HELLO WORLD!!!##")  
  
m1 = myclass()

**Output:-**

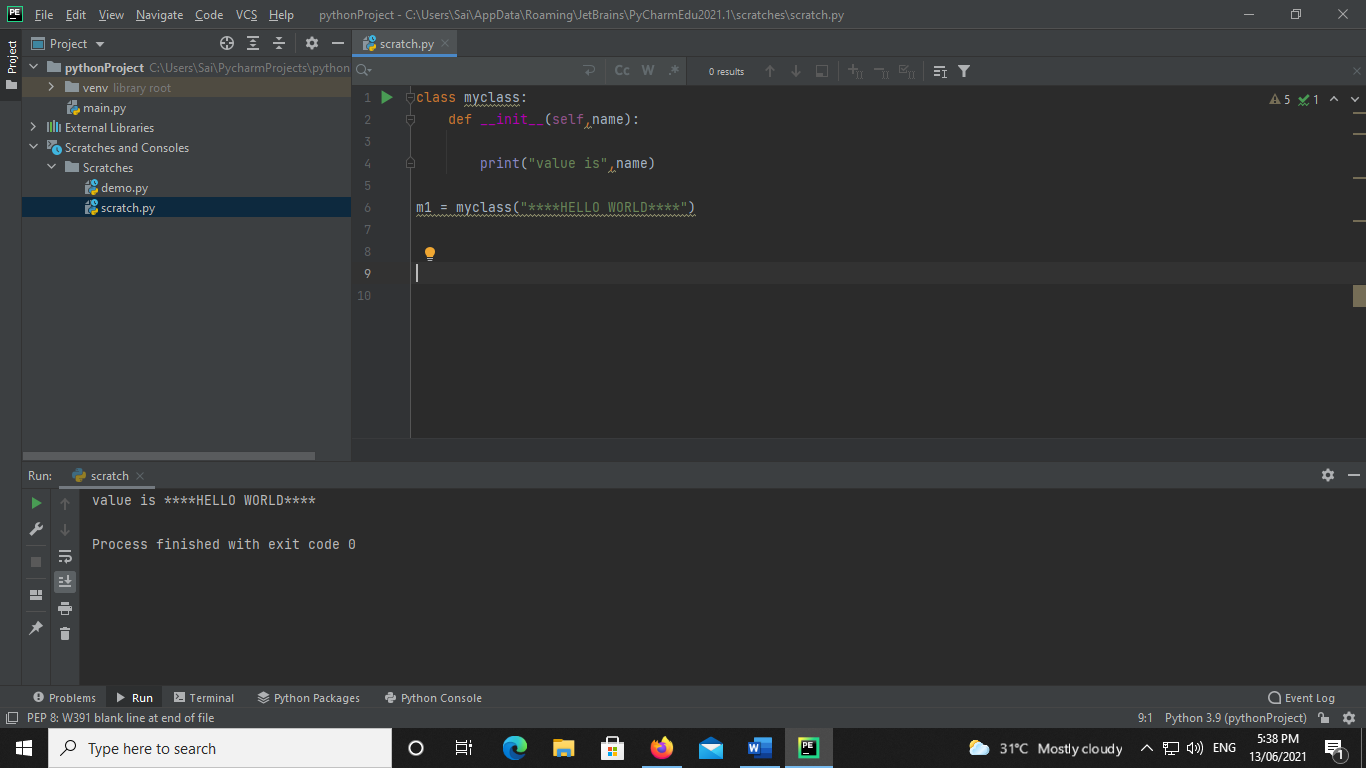
****

* **Parameterized constructor**

**Code:**

class myclass:  
 def \_\_init\_\_(self,name):  
  
 print("value is",name)  
  
m1 = myclass("\*\*\*\*HELLO WORLD\*\*\*\*")

**Output:-**

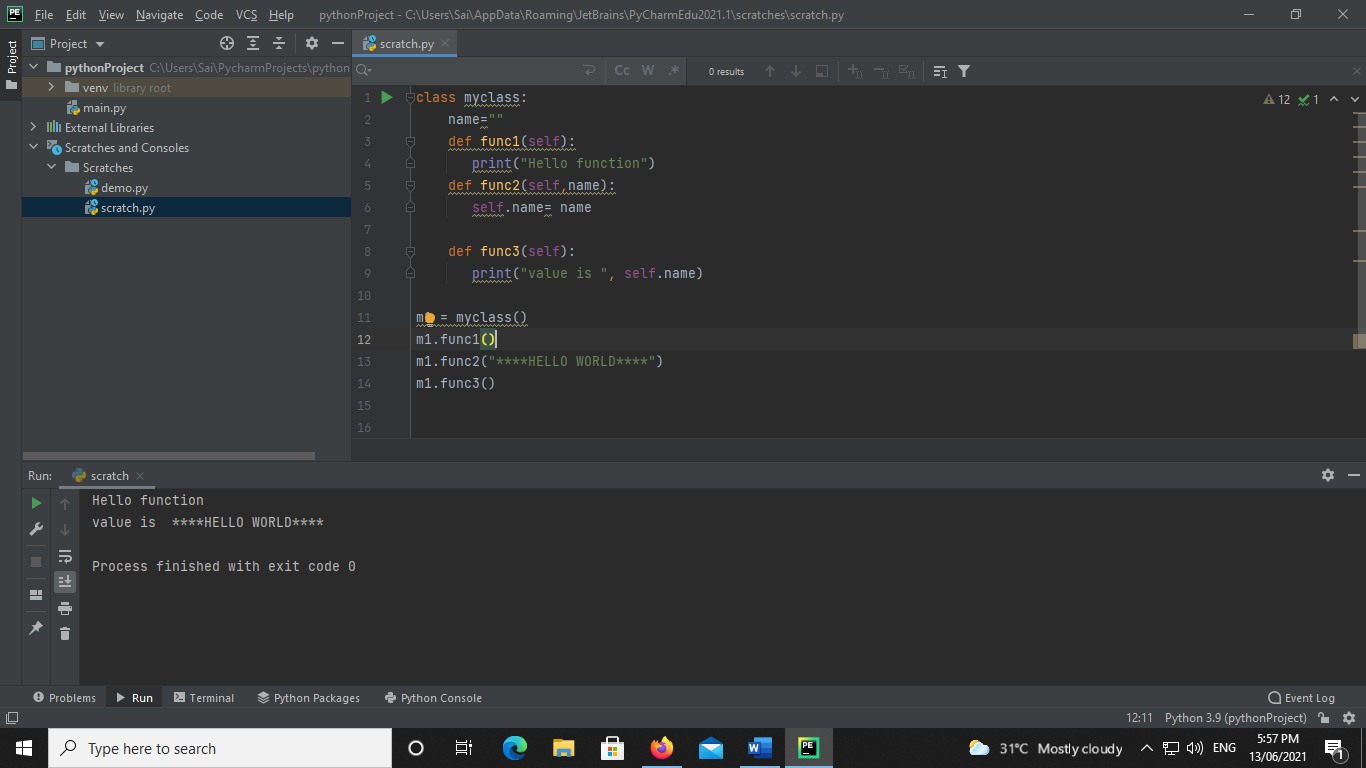


**Assign string value to class variable using Method**

**Code:**

class myclass:  
 name=""  
 def func1(self):  
 print("Hello function")  
 def func2(self,name):  
 self.name= name  
  
 def func3(self):  
 print("value is ", self.name)  
  
m1 = myclass()  
m1.func1()  
m1.func2("\*\*\*\*HELLO WORLD\*\*\*\*")  
m1.func3()

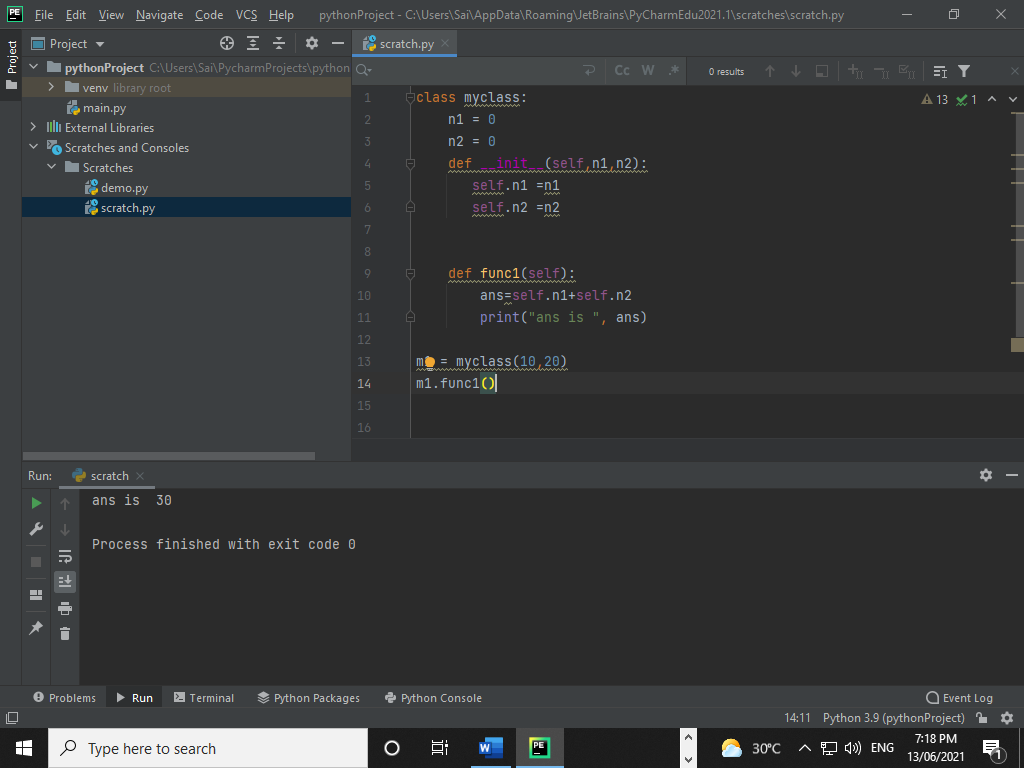
**Output:-**

****

**Assign string value to class variable using Constructor**

**Code:**

class myclass:  
 n1 = 0  
 n2 = 0  
 def \_\_init\_\_(self,n1,n2):  
 self.n1 =n1  
 self.n2 =n2  
  
  
 def func1(self):  
 ans=self.n1+self.n2  
 print("ans is ", ans)  
  
m1 = myclass(10,20)  
m1.func1()

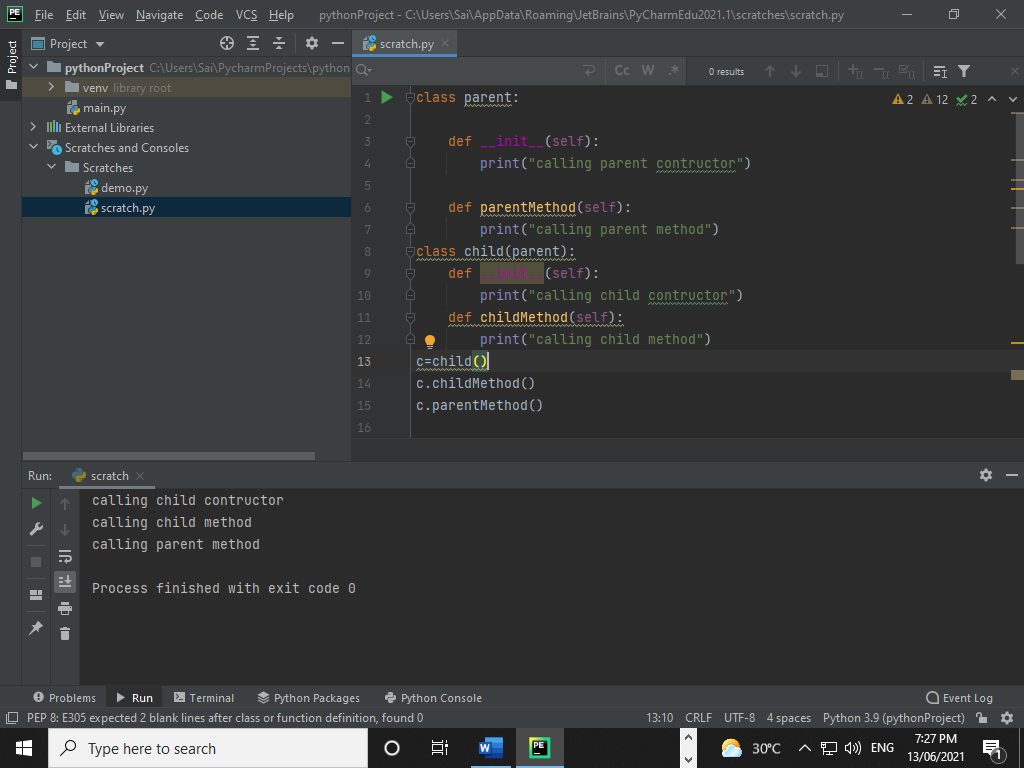
****

**Inheritance**

* Single-level Inheritance
* Multi-level Inheritance
* Multiple Inheritance
* Hierarchical Inheritance
* Hybrid Inheritance
* Single-level Inheritance

Code:

* class parent:  
    
   def \_\_init\_\_(self):  
   print("calling parent contructor")  
    
   def parentMethod(self):  
   print("calling parent method")  
  class child(parent):  
   def \_\_init\_\_(self):  
   print("calling child contructor")  
   def childMethod(self):  
   print("calling child method")  
  c=child()  
  c.childMethod()  
  c.parentMethod()

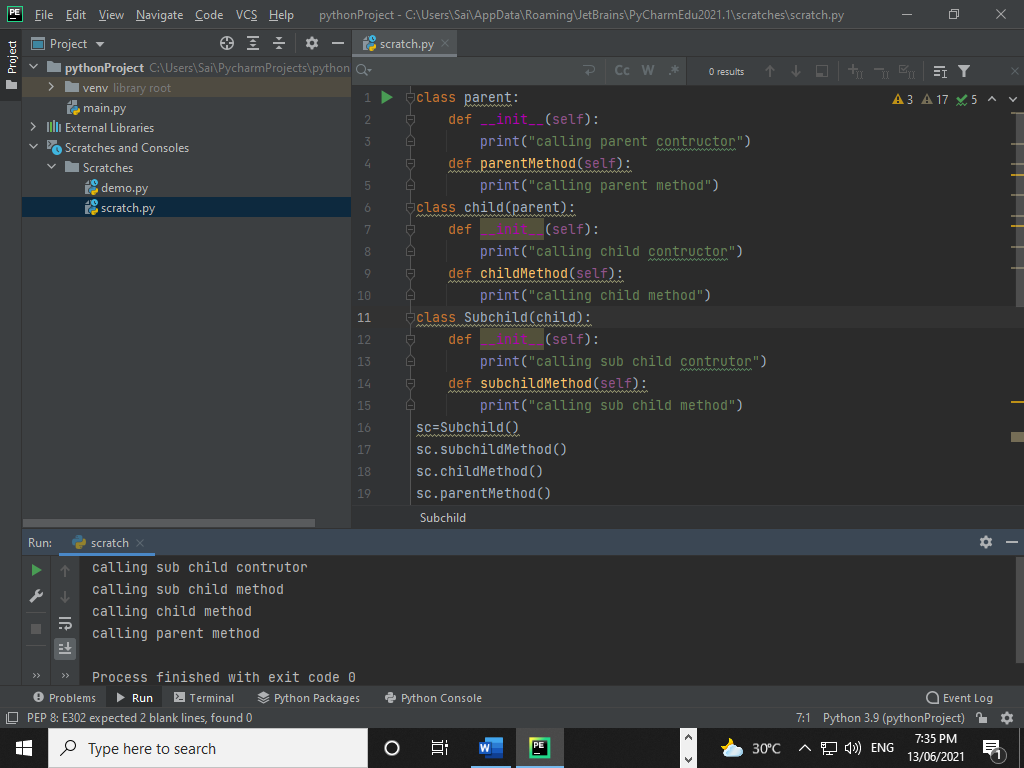


* Multi-level Inheritance

**Code:**

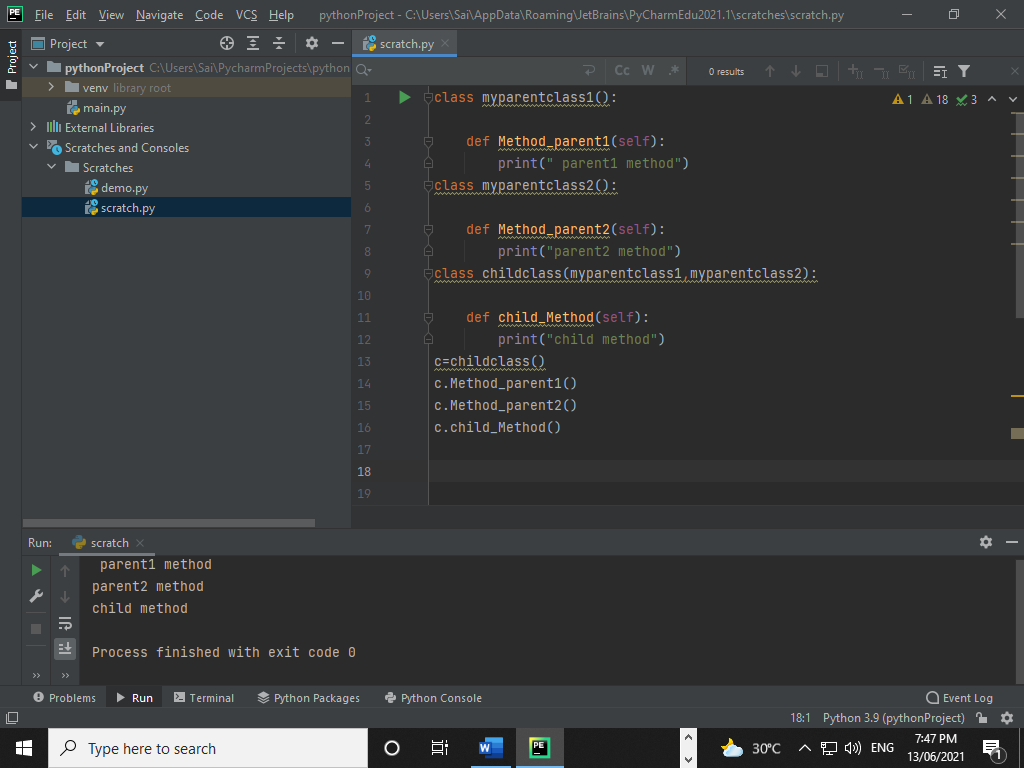
class parent:  
 def \_\_init\_\_(self):  
 print("calling parent contructor")  
 def parentMethod(self):  
 print("calling parent method")  
class child(parent):  
 def \_\_init\_\_(self):  
 print("calling child contructor")  
 def childMethod(self):  
 print("calling child method")  
class Subchild(child):  
 def \_\_init\_\_(self):  
 print("calling sub child contrutor")  
 def subchildMethod(self):  
 print("calling sub child method")  
sc=Subchild()  
sc.subchildMethod()  
sc.childMethod()  
sc.parentMethod()

**Output:-**

****

myparentclass1,myparentclass2):  
  
 def child\_Method(self):  
 print("child method")  
c=childclass()  
c.Method\_parent1()  
c.Method\_parent2()  
c.child\_Method()

**Output:**

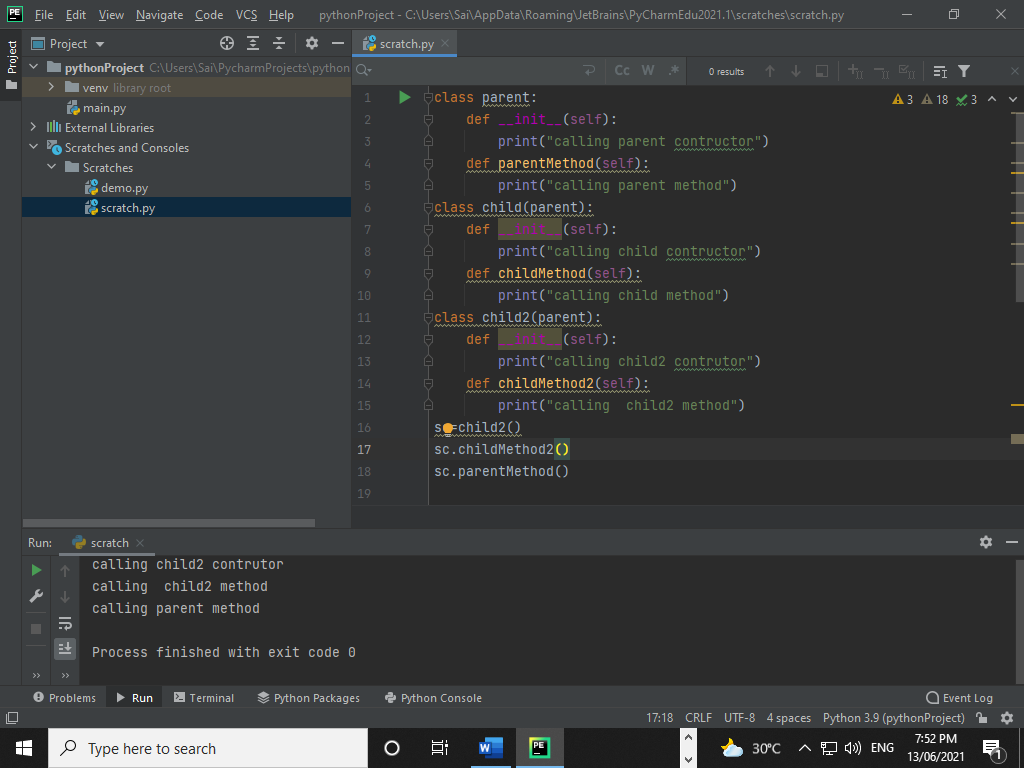
****

* Hierarchical Inheritance

**Code:**

class parent:  
 def \_\_init\_\_(self):  
 print("calling parent contructor")  
 def parentMethod(self):  
 print("calling parent method")  
class child(parent):  
 def \_\_init\_\_(self):  
 print("calling child contructor")  
 def childMethod(self):  
 print("calling child method")  
class child2(parent):  
 def \_\_init\_\_(self):  
 print("calling child2 contrutor")  
 def childMethod2(self):  
 print("calling child2 method")  
sc=child2()  
sc.childMethod2()  
sc.parentMethod()

**Output:-**

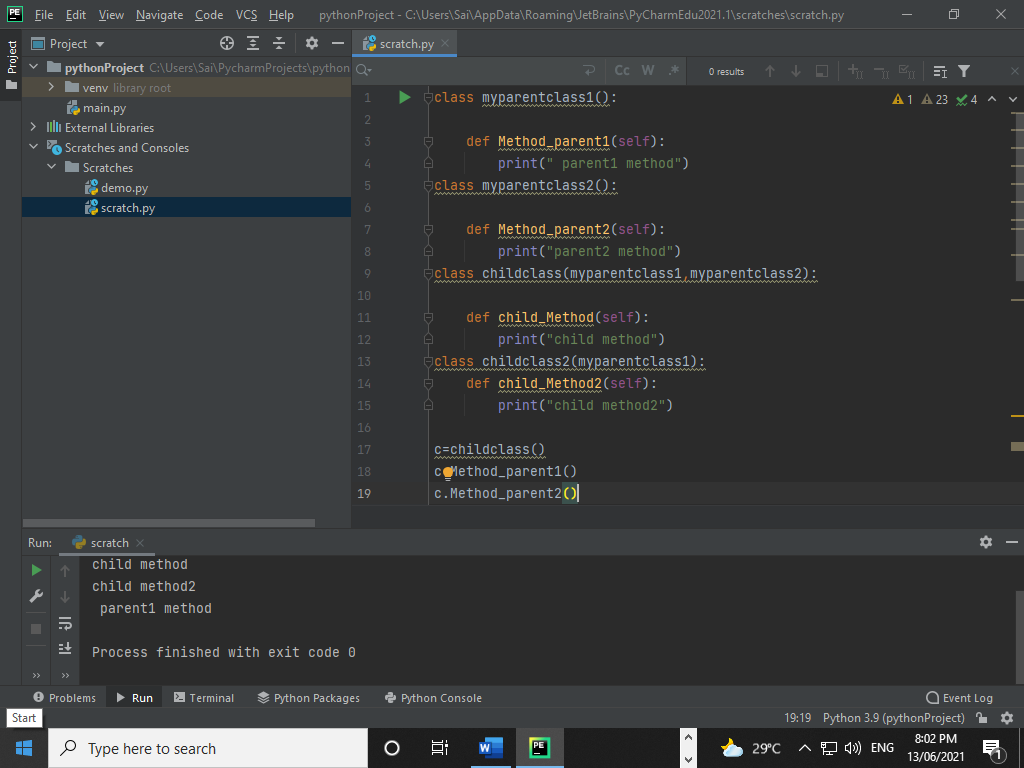
****

* Hybrid Inheritance

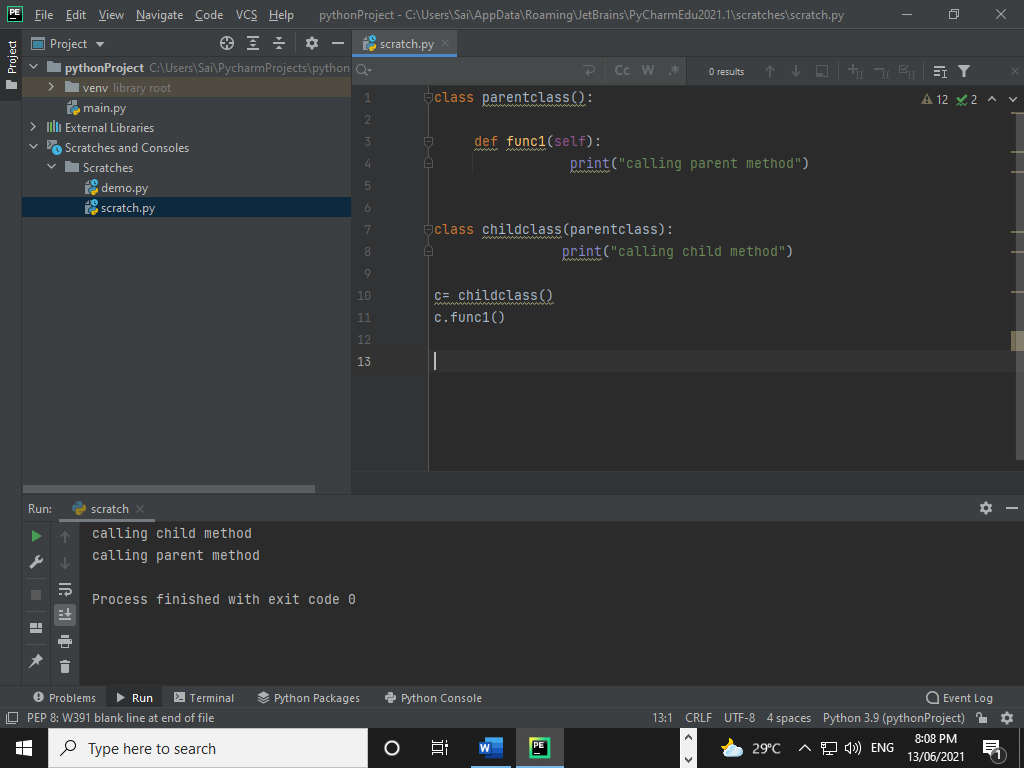
**Code:**

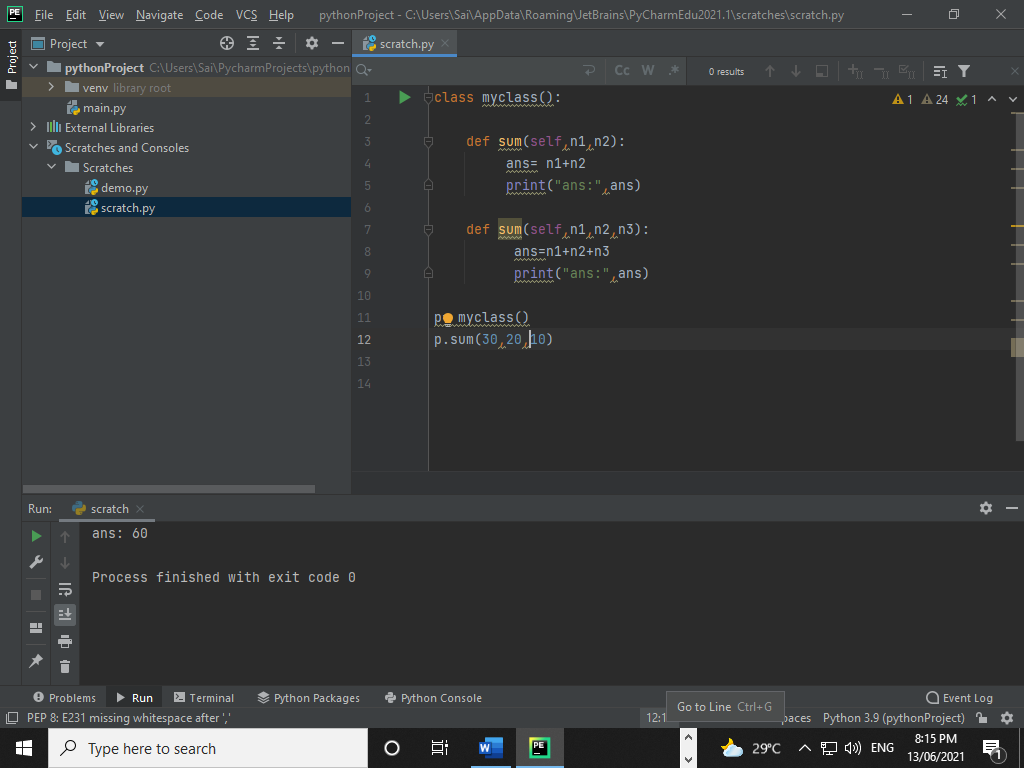
class myparentclass1():  
  
 def Method\_parent1(self):  
 print(" parent1 method")  
class myparentclass2():  
  
 def Method\_parent2(self):  
 print("parent2 method")  
class childclass(myparentclass1,myparentclass2):  
  
 def child\_Method(self):  
 print("child method")  
class childclass2(myparentclass1):  
 def child\_Method2(self):  
 print("child method2")  
  
c=childclass()  
c.Method\_parent1()  
c.Method\_parent2()  
c.child\_Method()  
  
c=childclass2()  
c.child\_Method2()  
c.Method\_parent1()

**Output:-**

****

Polymorphism

* Type of Polymorphism:-
* Overloading Method
* Overriding Method
* Overloading Method
* **** Overriding Method

****