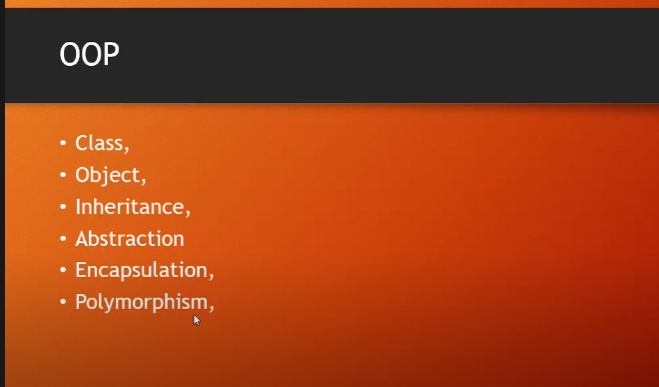
**Python OOP**



Function is a plate where is a user can reduce code and save time for coding

Function হচ্ছে ২ প্রকার

১.User Defiend Function

2.Library Function

Function Syntax:

**Def function\_name(perameter):**

**Logical terms…**

**Function\_name(perameter value pass)**

সব শেষে function এর বাহিরে আউটপুট এর জন্য function এর নাম ধরে কল দিতে হবে

Example:

def add(x,y):

    sum=x+y

    print(sum)

add(10,20)

add(10,20)

add(10,20)

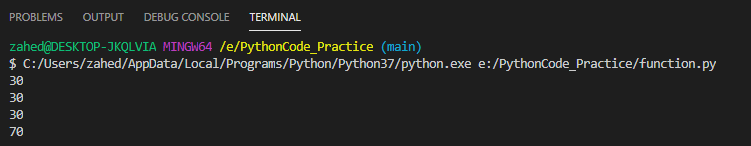
def summation(x,y,z):

    sum=x+y+z

    print(sum)

summation(20,10,40)

Output:



Returning Value form Function

এখানে return বলতে function এর under print statement এর পরিবর্তে return use করতে হয়|

Example:

#Sum

def add (a,b):

    sum = a+b

    return sum

result=add(10,30)

print(result)

#find the largest value using function

def largest(a,b):

    if a>b:

        return a

    elif b>a:

        return b

print(largest(100,1000))

print(largest(100,50))

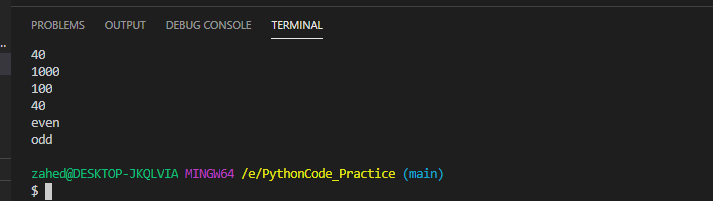
print(largest(30,40))

# result=largest(20,30)

# result=largest(20,100)

# print(result)

output:



Return function use করে even and odd number করার জন্য একটি program লিখা হল:

#even odd

def even (a):

    if a%2==0:

        return "even"

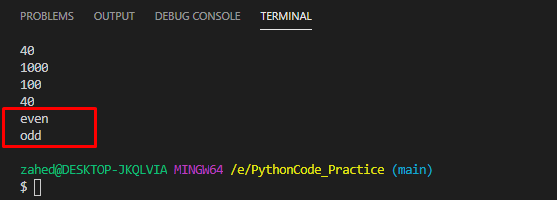
    else:

        return "odd"

print(even(4))

print(even(13))

Output:



**Xargs and Xxargs function in python**

Xargs এর কাজ হচ্ছে একটি perameter এর under একাধিক ডাটা পাস করানো যার জন্য function এর under এ perameter এর আগে একটি \* sign দিয়ে দিতে হবে

#xargs mane hocche function perameter same rekhe akadik value pass

koorano

def st\_info (\*details):

    print(details)

    print(details[2])

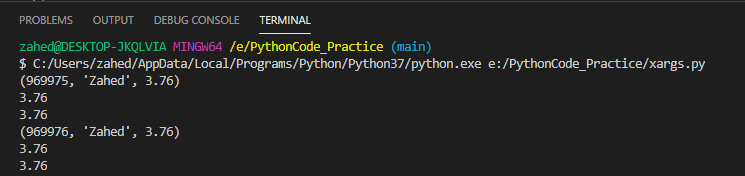
    print(details[2])

st\_info(969975,"Zahed",3.76)

st\_info(969976,"Zahed",3.76)

print(st\_info)

output:



Example-2

def add (\*summation):

    sum=0

    for num in summation:

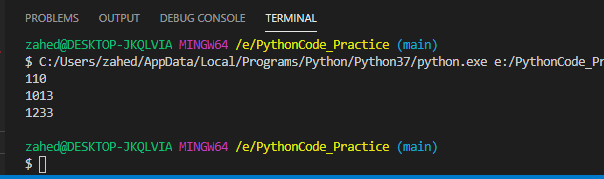
        sum=sum+num

    print(sum)

add(10,100)

add(10,1003)

add(10,1003,220)

Output: 

**Xxargs** হচ্ছে key ধরে একাধিক value পাস করানো যার জন্য \*\* use করতে হবে|

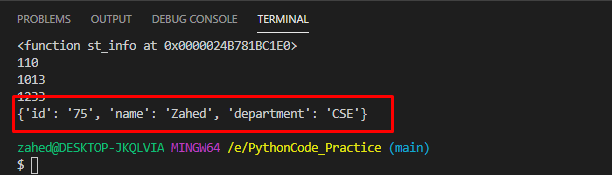
যাকে কীওয়ার্ড আর্গুমেন্ট আকারে ডাটা পাস হবে|

Example:

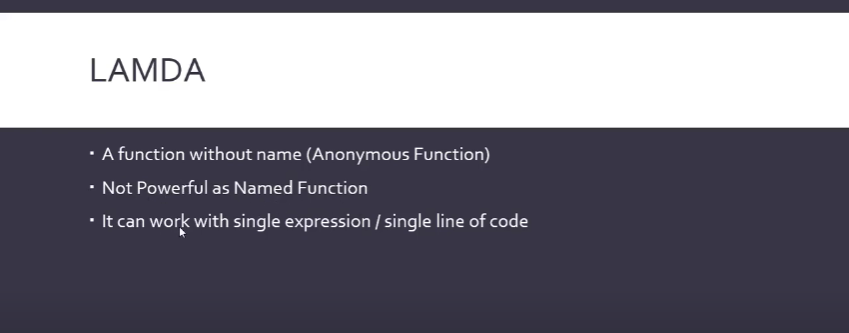
def student\_info(\*\*information):

    print(information)

student\_info(id="75",name="Zahed",department="CSE")



LAMBDA FUNCTION



**LAMBDA Syantax:**

**Lambda perameter:expression**

**Example:**

#named function

def calculate(a,b):

   return a\*a+2\*a\*b+b\*b

print(calculate(10,20))

print(calculate(10,200))

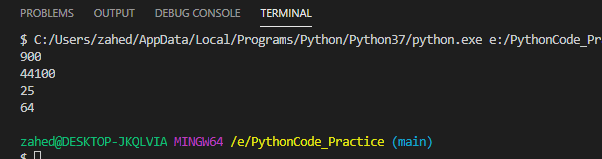
#LAMBDA FUNCTION WORk

print((lambda  a,b:a\*a+2\*a\*b+b\*b)(2,3))

p=(lambda  a,b:a\*a+2\*a\*b+b\*b)(2,6)

print(p)

**Output:**



**Map Function:**

Map function হচ্ছে powerful function যার কাজ হচ্ছে লিস্ট এর মধ্যে ডাটা পাস করা

Map এর সিনট্যাক্স:

map(function, itera1)

print(...)

map হচ্চে একটি function যার মধ্যে ডাটা iterable হতে পারে নিজে থেকে

Square measure:

def square(x):

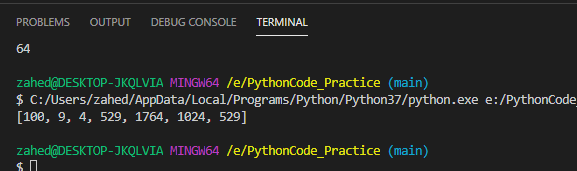
    return x\*x

num=[10,3,2,23,42,32,23]

result=list(map(square,num))

print(result)

Output:



\*Even & odd findout and Reserve in a list\*

def even(num):

    print(num)

    if num%2==0:

        return num

    else:

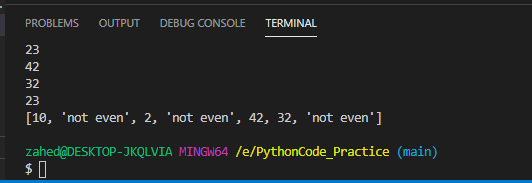
        return "not even"

num=[10,3,2,23,42,32,23]

result=list(map(even,num))

print(result)

Output:



Filter Function:

Filter function এর কাজ হচ্ছে sepcific element লিস্ট থেকে বাদ দেয়া|

Example:

Check Even & odd ignore the odd number:

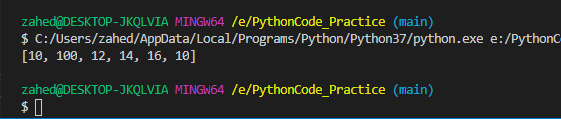
#check even in filter

num=[10,100,12,14,15,16,17,10,29,41]

result=list(filter(lambda x:x%2==0,num))

print(result)

output:



Zip function এর কাজ ই হচ্ছে দুইটা লিস্ট কে একসাথে জোড়া লাগানো

কোড:

#zip function er kaj hocche 2ta list ke akta list combine kora

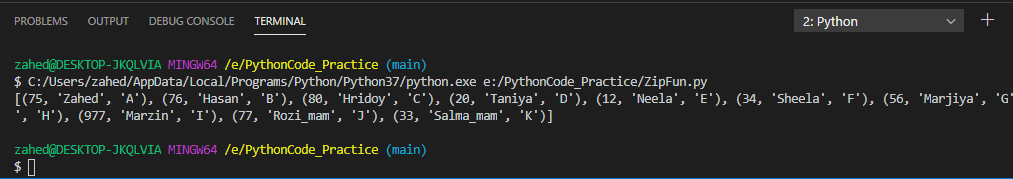
id=[75,76,80,20,12,34,56,678,977,77,33,12]

name=["Zahed","Hasan","Hridoy","Taniya","Neela","Sheela","Marjiya","Maruf","Marzin","Rozi\_mam","Salma\_mam"]

info=list(zip(id,name,"ABCDEFGHIJK"))

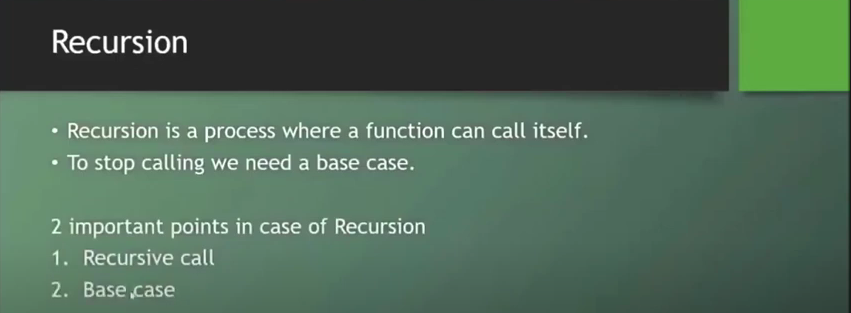
print(info)

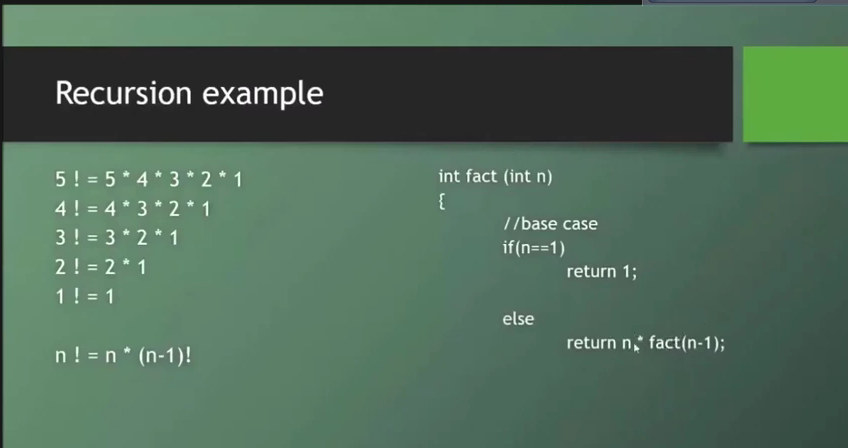
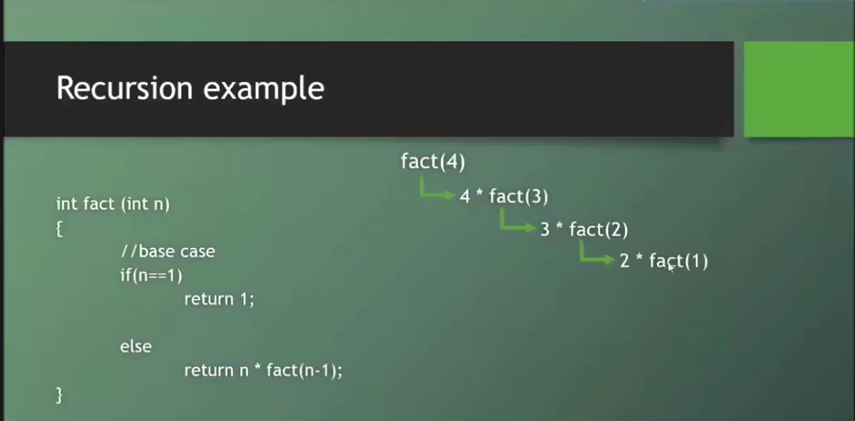
output:



Recursion

Recursion হচ্ছে একটি function যেখানে funciton নিজেই নিজেকে কল দিতে পারে যাকে কন্ট্রোল করার জন্য একটি base case উপর কাজ করতে হয় এবং এই function কে থামানোর জন্য base case কে কল করতে হয়|





#factorial er man ber kora

def fact(n):

    if n==1:

        return 1

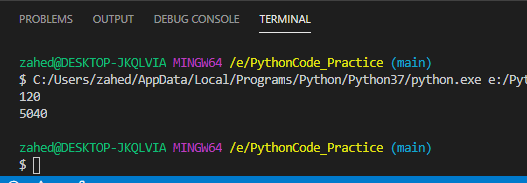
    else:

        return n\*fact(n-1)

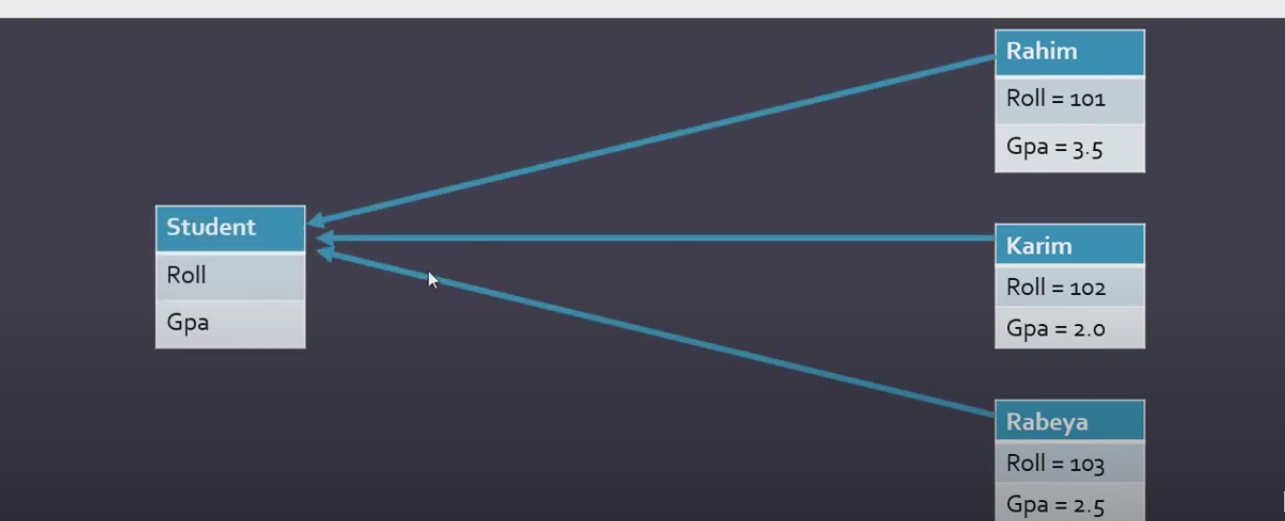
print(fact(5))

print(fact(7))

output:



Class and Object



class is a user-defined blueprint or prototype from which objects are created.

**Some points on Python class:**

* Classes are created by keyword class.
* Attributes are the variables that belong to a class.
* Attributes are always public and can be accessed using the dot (.) operator. Eg.: Myclass.Myattribute

Class Syntax:

class ClassName():

# Statement-1

.

.

.

# Statement-N

class student():

    name=""

    gpa=""

Zara=student() #এখানে obj create করা হয়েছে|

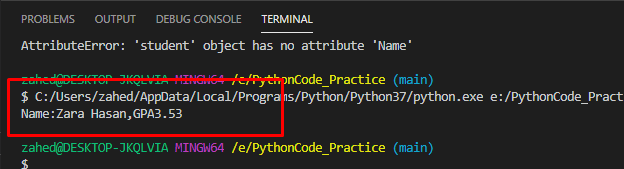
Print(isinstance(Zara,student))# ata diye zara cls er obj kina ta check korar jonno.

Zara.name="Zara Hasan"

Zara.gpa=3.53

print(f"Name:{Zara.name},GPA{Zara.gpa}")

Output:



Method:

**SELF Method: সেলফ মেথড functionটি ক্লাস এর under ডাটা নিজে নিজে কল করতে পারে**

Method হচ্ছে কোন ক্লাস এর under function ডিক্লেয়ার করা যার কাজ হচ্ছে কোড কে বার বার reuse করা|

কোড:

class student():

    name=""

    roll=""

#akhane self akta convention keyword jar kaj hocche elemnet value nije nije call dibe

    def show(self):

        print(f"Name:{self.name},Roll:{self.roll}")

Zahed=student()

Zahed.name="Zahed Hasan"

Zahed.roll=244

Zahed.show()

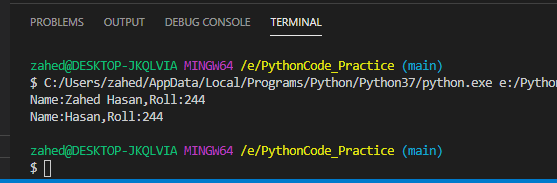
Hasan=student()

Hasan.name="Hasan"

Hasan.roll=244

Hasan.show()

Output:



Another way :

class student():

    name=""

    roll=""

    def value\_pass(self,name,roll):#akhne value pass korarnor akti function use kora hoyeche

        self.name=name

        self.roll=roll

#akhane self akta convention keyword jar kaj hocche elemnet value nije nije call dibe

    def show(self):

        print(f"Name:{self.name},Roll:{self.roll}")

Zahed=student()

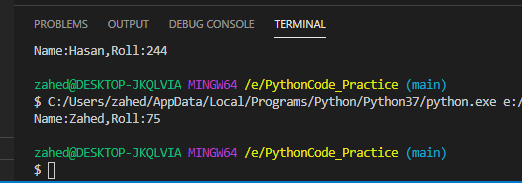
# Zahed.name="Zahed Hasan"

# Zahed.roll=244

Zahed.value\_pass("Zahed",75)#function ke call kora hoyeche

Zahed.show()

Output:



**Constructor Method:** constructor method কে init method বলা হয় এই method variable এর মান আলাদাভাবে ডিক্লেয়ার করতে হয় না function ডিক্লেয়ার করে দেয়া যায় এবং এই method কে আলাদাভাবে কল করার প্রয়োজন পড়ে না|

#Construcutor

class family():

    name=""

    blood\_group=""

    def \_\_init\_\_(self, name, blood\_group):

       self.name=name

       self.blood\_group=blood\_group

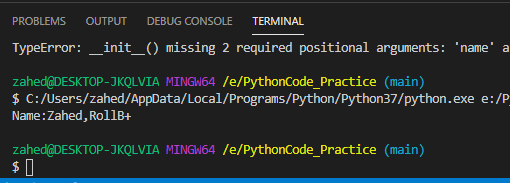
    def show(self):

        print(f"Name:{self.name},Roll{self.blood\_group}")

Zayan=family("Zahed","B+")#এখানে আলাদা করে ডাটা পাস এর জন্য value ইনপুট দিতে হহ্চ্ছে না জাস্ট obj এর মাধ্যমেই value পাস করা যাচ্ছে যা function কলিং ছাড়াই

Zayan.show()

Output:



**INHERITENCE: inheritence method() হচ্ছে এমন একটি method যা একটি ক্লাস এর বৈশিষ্ট অন্য একটি ক্লাস এর under নিয়ে আসা হয় যাকে পৈত্রিক সূত্রে বলা ও চলে|**

class phone:

    def call(self):

        print("you can call")

    def message(self):

        print("you can message")

class realme(phone):

    # def call(self):

    #     print("you can call")

    # def message(self):

    #     print("you can message")

    def battery(self):

        print("Joldi battery palta")

p=phone()

p.call()

p.message()

print()

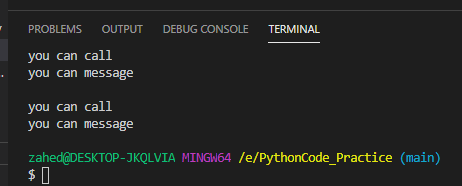
#///

R=realme()

R.call()

R.message()

Output:



MethodOverriding:method overriding এর কাজ হচ্ছে কোন একটি method কে overwrite করে তার বৈশিষ্ট পরিবর্তন করে ফেলা|

এখানে কোড base ক্লাস এর বৈশিষ্ট কে ইনহেরিট করে নিয়ে আশা হয়েছে|

class phone:

    def \_\_init\_\_(self):

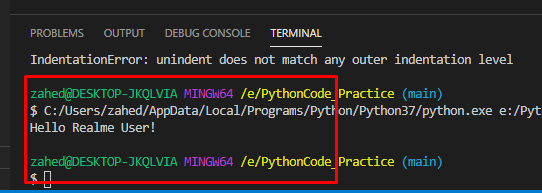
        print("Hello Realme User!")

class iphone(phone):

     pass

r=iphone()

output:



now go to override:

class phone:

    def \_\_init\_\_(self):

        print("Hello Realme User!")

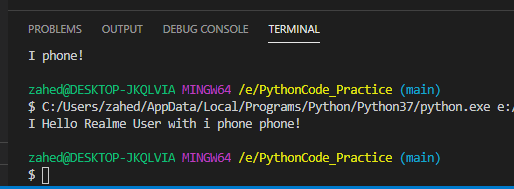
class iphone(phone):

     def \_\_init\_\_(self):

         print('I Hello Realme User with i phone phone!')

r=iphone()

Output:



Now Super class inherit করা উপরের কোড এর মধ্যে

যদি base ক্লাস এর ডাটা কে overridng class এর মধ্যে আনতে হয় তাহলে সুপার ক্লাস use করতে হবে example দেয়া হল:

class phone:

    def \_\_init\_\_(self):

        print("Hello Realme User!")

class iphone(phone):

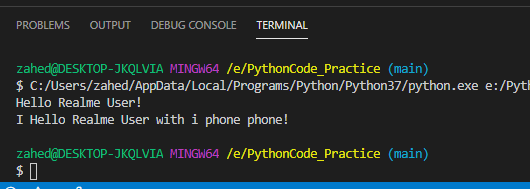
     def \_\_init\_\_(self):

         super().\_\_init\_\_()

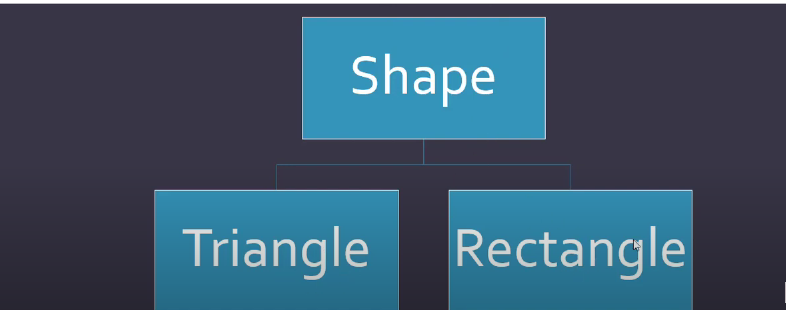
         print('I Hello Realme User with i phone phone!')

r=iphone()

output:



Practical example of inheritence:



এখানে shape নামের একটি ক্লাস থাকবে সেটি traingle এবং rectangle ক্লাস shape ক্লাস কে ইনহেরিট করবে|

class shape:

    def \_\_init\_\_(self, base,height):

        self.base=base

        self.height=height

    def area(self):

        print("im on shape")

class triangle(shape):

    def area(self):

        area=0.5\*self.base\*self.height

        print("The Triangle Area is",area)

class rectangle(shape):

    def area(self):

        area=self.base\*self.height

        print("The Triangle Area is",area)

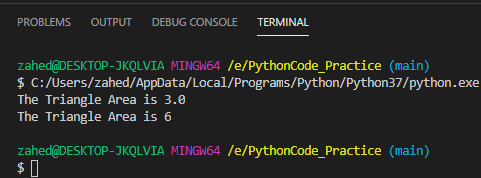
t1=triangle(3,2)

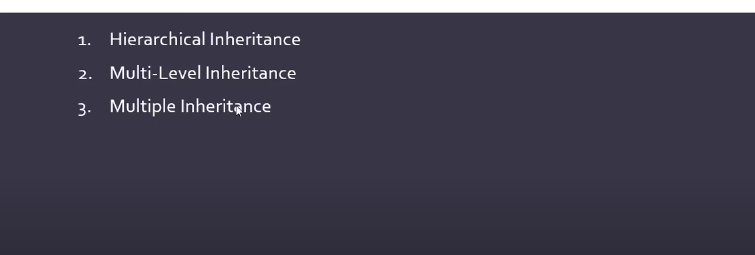
t1.area()

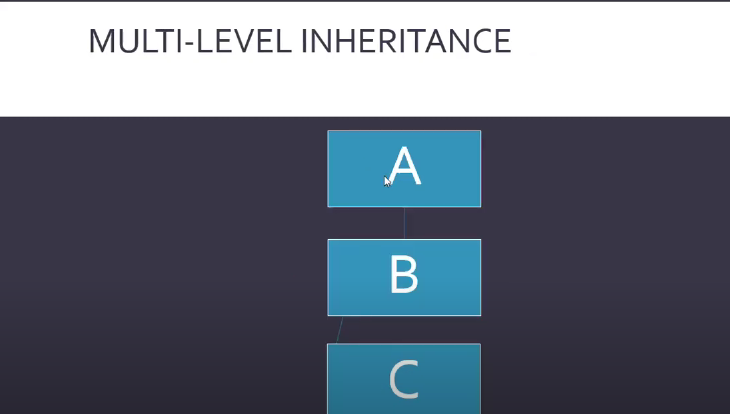
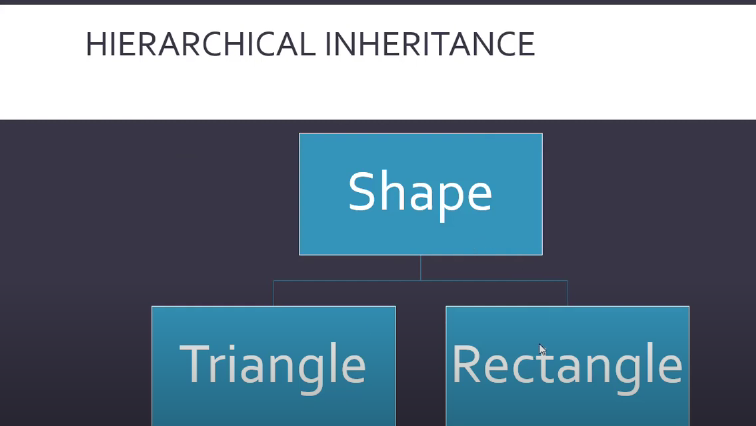
t1=rectangle(3,2)

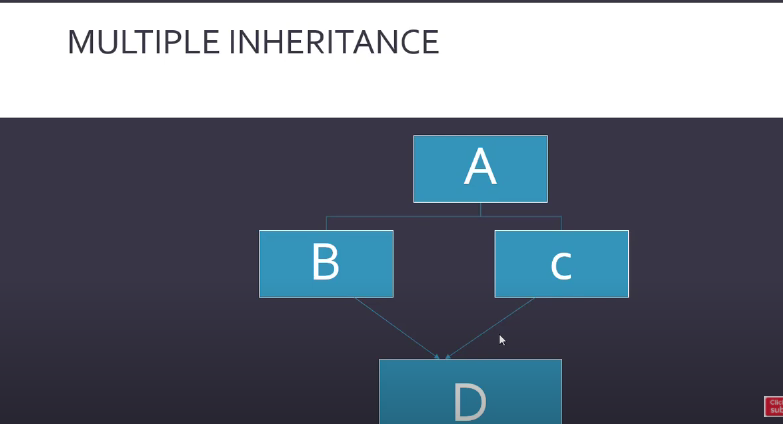
t1.area()

Output:



Types of inheritence:





MultiLevel Inheritence:

class A ():

    def display(self):

        print("Im on A Class")

class B (A):

    #inherit A

    def display1(self):

        print("Im on B Class")

class C (B):

    #inherit B

    def display2(self):

        print("Im on C Class")

obj1=C()

obj1.display()

obj1.display1()

obj1.display2()

Ouput:

