কোন কিছু প্রিন্ট করার জন্য print ব্যবহার করা হয়.

যদিও কোন স্ট্রিং লেখা হয় তাহলে স্ট্রিং এর উভয়পাশে ডবল কোটেশন বা সিঙ্গেল কোটেশন দিতে হবে|

Ex:

print('Hallo world')

**Comments:** সিঙ্গেল লাইনে কমান্ড লেখার জন্য #Comment ব্যবহার করা হয়.

ডবল লাইনে কমান্ড লেখার জন্য ‘’’Comment’’’ ব্যবহার করা হয়.

Ex:

# My Bio data

'''

This is

my

bio data

'''

* কোন লেখাকে তার পরের লাইনে নেওয়ার জন্য \n ব্যবহার করা হয়.

EX:

print('Antu \n 0183248716')

**Variable and Data type:** ভেরিয়েবল হলো কোন কিছু মান পরিবর্তন করা যায় .

* যদি ভেরিয়েবল হিসেবে স্ট্রিং অ্যাড করতে হয় তাহলে (+) ব্যবহার করা হয়
* যদি ভেরিয়েবল হিসেবে ইন্টিজার অ্যাড করতে হয় তাহলে (,) ব্যবহার করা হয়

EX:

name = "DD"

age = 55

gpa = 3.88

'''

    Change valude

'''

print("Antu"+" das")

print("Our student name is "+name)  # Other ous  = (,)  String = (+)

print(name+" lived in Dhaka")

print(name+" is age is ",age)

print("At age is ",age," year old He start phython")

print("Secondery Exim gpa ",gpa)

**Numerical\_Operations:**

EX:

print(4\*\*2)  #power

print(4+2)

print(4-2)

print(4\*2)

print(4/2)  #ভাগফল ভাগশেষ ২ টাই দেখাযাবে

print(4%2)  #ভাগশেষ

print(4//2) #only ভাগফল

**User Input:**

**EX:**

n = input("")

a = input("")

g = input("")

print("Name: "+n)

print("Age: "+a)

print("gpa: "+g)

q=int(input('Enter Any Number '))

print(q)

**Function:**

**Math related Library function:** নিম্নোক্ত সবগুলো from math import\* লাইব্রেরি ফাংশনে রয়েছে

* max = সব থেকে বড় সংখ্যা খুজতে

EX:

print(max(22,77.5))

* min = সব থেকে ছোট সংখ্যা খুজতে

EX:

print(min(22,77.5))

* abs = (-) মান কে (+) এ পরিনত করতে

EX:

print(abs(-22))

* pow = কোন কিছুর ওপর পাওয়ার বসাতে

EX:

print(pow(2,3))

* sqrt = রুট এর জন্য

EX:

print(sqrt(25))

* round = কাছাকাছি পূর্ন সংখ্যা (প্রায় মান)

EX:

print(round(25.50))

output: 26

* floor = নিচের দিকে পূর্ন সংখ্যা

EX:

print(floor(8.7))

* ceil = ওপরের দিকে পূর্ন সংখ্যা

EX:

print(ceil(9.2))

Type casting: type এর সাহায্যে ডাটা টাইপ চেক করে

EX:

n=10  
print(type(n))

output: <class 'int'>

**Formated string:** যদি আমরা কোন লেখা সহ আউটপুট পেতে চাই তাহলে ফরমেটিং স্ট্রিং ব্যবহার করা হয়|

EX:

num1 = 10  
num2 = 20  
print(f'Our sum value is {num1} + {num2} = {num1+num2}')

output: Our sum value is 10 + 20 = 30

**Use of end key word:**

দুইটি লাইন এক end=” ” লাইনে প্রিন্ট করার জন্য ব্যবহার করা হয়

EX:

print('Priyantu das Antu',end=' ')  
print('01832487161')

**Relational Operator and Boolean Data type:**

Boolean Value: বুলিয়ান ভ্যালু মধ্যে দুটি মান আসতে পারে

* True
* False

এর আউটপুট যদি সত্যি হয় True যদি মিথ্যা হয় False

EX:

print(30>20)# > = 20 এর চেয়ে বড় 30

print(30<20)# < = 30 এর চেয়ে ছোট 20

print(30>=20) # >= = বড় অথবা সমান

print(30<=20) # <= = ছোট অথবা সমান

print(30==20) # == = সমান-সমান

print(30!=20)# != = সমান নয়

in and not in:

in: মধ্যে রয়েছে

not in: মধ্যে নেই

**CONTROL STATEMENT:** দুই ধরনের কন্ট্রোল স্টেটমেন্ট রয়েছে

* Conditional control statement (if,elif,else)
* Loop control statement (for,while)

**Conditional control statement (if,elif,else):**

* যদি একটি স্টেটমেন্টের নিয়ন্ত্রণ করতে হয় তাহলে if ব্যবহার করা উত্তম
* যদি দুইটি স্টেটমেন্ট এর নিয়ন্ত্রণ করতে হয় তাহলে প্রথমটি if ও পরেরটি else ব্যবহার করা উত্তম
* তিনটি স্টেটমেন্টের নিয়ন্ত্রণ করার জন্য প্রথমটি if ও দ্বিতীয়টি elif ও তৃতীয়টি else ব্যবহার করা উত্তম
* যদি এগুলো চেয়েও বেশি স্টেটমেন্টের নিয়ন্ত্রণের দরকার হয় তাহলে প্রথমটি if শেষের টি else বাকি স্টেটমেন্ট গুলো elif

[যদি সবগুলো if হয় তাহলে যদি প্রথম if সত্যি হয় তাহলে সবগুলোর output দিবে]

If:

EX:

mark = int(input('Enter your number: '))  
if mark >=33:  
 print('Pass')  
if mark<33:  
 print('Fall')

elif:

EX:

mark = float(input('Enter Your Mark: '))  
if mark >= 80:  
 print('A+')  
elif mark >= 70:  
 print('A')  
elif mark >= 60:  
 print('A-')  
elif mark >= 50:  
 print('B')  
elif mark >= 33:  
 print('C')  
else:  
 print("F")

else:

EX:

mark = float(input('Enter your number: '))  
if mark >=33:  
 print('Pass')  
else:  
 print('Fall')

Inner if Statement / Nested if: স্টেটমেন্ট এর ভিতর আরেকটি if স্টেটমেন্ট চালনা করাকে বলে

EX:

#Compier 3 number in largest value

num1  =-200

num2  = 500

num3  = 40

if num1 > num2:

    if num1 > num3:

        print(num1)

    else:

        print(num3)

if num2 > num1:

     if num2 > num3:

        print(num2)

     else:

         print(num3)

Ternery Operator:

EX:

num1=20

num2=10

"""

if num1 > num2:

    print(num1)

else:

    print(num2)

    """

print (num1 if num1 > num2 else num2)

**Logical operator:** তিন ধরনের লজিক্যাল অপারেটর রয়েছে

* And
* Or
* Not

**And:** দুটি স্টেটমেন্ট সত্য হলে এর ভেতরের আউটপুট দিবে . দুইটা স্টেটমেন্ট সত্য না হলে আউটপুট দিবে না.

EX:

num1 = 20  
num2 = 10  
num3 = 5  
  
if num1 > num2 and num1 > num3:  
 print(num1)  
elif num2 > num1 and num2 > num3:  
 print(num2)  
else:  
 print(num3)

**Or:**যেকোনো একটি কন্ডিশন সত্য হলে আউটপুট দিবে

EX:

ch = 'A'  
  
if ch =='a' or ch =='e' or ch =='i' or ch=='o' or ch =='u' or ch =='A' or ch =='E' or ch =='I' or ch=='O' or ch =='U':  
 print('Vowel')  
  
else:  
 print('Consonant')

**Loop:** একই কাজ বারবার করতে হলে Loop ব্যবহার করা হয়.

* For loop
* While loop

**For loop:**

STEPES:

1. কোন ভেরিয়েবল স্টোর হবে in কোন ভেরিয়েবল এর মান

EX:

num = [10,50,90,70,80]  
print(num)  
for x in num:  
 print(x)

**While loop:**

STEPES:

1. কত থেকে শুরু হবে
2. #কন্ডিশন
3. #মান বেড়ে যাওয়া বা কমে যাওয়া বা অপরিবর্তিত থাকা

EX:

i = 1 #কত থেকে শুরু হবে  
while i<=10: #কন্ডিশন  
 print(i) #কাজ  
 i=i+1 #মান বেড়ে যাওয়া বা কমে যাওয়া বা অপরিবর্তিত থাকা  
print('End') #লুপের বাইরের অংশ

**Break and continue keyword:** এগুলো অবশ্যই লুপের মধ্যে চালাতে হবে না হলে প্রোগ্রাম অ্যারো দিবে.

**Break:** একটি লুপের ভেতর if কন্ডিশন এপ্লাই করে দিতে হয় যদি কন্ডিশন সত্য হয় তাহলে break কিওয়ার্ড টি কাজ করবে. Break মূলত ব্যবহার করা হয় প্রোগ্রাম থেকে বের হয়ে যাওয়ার জন্য

EX:EEX

i=1  
while i<=100:  
 if i == 20:  
 break  
 print(i)  
 i=i+1  
print("Hello")

**Continue:** একটি লুপের ভেতর if কন্ডিশন এপ্লাই করে দিতে হয় যদি কন্ডিশন সত্য হয় তাহলে continue কিওয়ার্ড টি কাজ করবে. continue মূলত ব্যবহার করা হয় প্রোগ্রাম নিচের দিকে না গিয়ে পুনরায় লুপে ফিরে যাওয়ার জন্য.

EX:

i=1  
while i<=100:  
 if i == 20:  
 continue  
 print(i)  
 i=i+1  
print("Hello")

List: অনেকগুলো আইটেম স্টোর করার জন্য অবজেক্ট

লিস্ট যেভাবে তৈরি করে:

Subject=["C","c++","Java","Python","Android","OS","TOC"]

* [0] তম ইনডেক্স এর মান দেখার জন্য

Ex:

print(subject[0]) # [0] তম ইনডেক্স এর মান দেখার জন্য

* [2:] ইনডেক্সের পর বাকি সবগুলো প্রিন্ট করার জন্য

Ex:

print(subject[2:]) # [2] তম ইন ডেক্স এর পর এর মান দেখার জন্য

* উল্টো থেকে মান দেখার জন্য

Ex:

print(subject[-1]) # উল্টা দেখার জন্য

* কোন আইটেম লিস্টে আছে কিনা তা চেক করার জন্য

Ex:

print("Python" in subject)#আইটেম আছে কি নাই তা দেখার জন্য

* আইটেমটি নেই তা প্রোগ্রামকে বলার জন্য

Ex:

print("Aco" not in subject)#কোন কিছু নাই তা বলে দেওয়ার জন্য

* প্রোগ্রামে কোন কিছু এড করার জন্য

EX:

print(subject + ["Aco",21])#কোন কিছু add করার জন্য

* লিস্ট বারবার প্রিন্ট করার জন্য

EX:

print(subject \* 3)#বার বার প্রিন্ট করার জন্য

List function:

subject = ["C","C++","Java","Python","BASIC","Rubi"]

print(len(subject)) # len ফাংশনের লেন্ত বরে করার জন্য।

subject.append("TOC")

print(subject) # append এর মধ্যে থাকা ওয়াড লিস্ট এর শেষে যু্ক্ত হবে।

subject.insert(2,"MAC OS")

print(subject) # insert যে কোন তম ইনডেক্স এ কোন কিছু যুক্ত করার জন্য।

subject.remove("C++")

print(subject) # remove কোন কিছু বাদ দেওয়ার জন্য ব্যাবহার করা হয়।

mon = [30,10,20,50,80,70,90]

mon.sort()

print(mon) # sort সিরিয়ালি ছোট থেকে বড় আকারে সাজিয়ে লেখার জন্য।

mon.reverse()

print(mon) # sort সিরিয়ালি বড় থেকে ছোট আকারে সাজিয়ে লেখার জন্য।

mon.pop()

print(mon) # ১ বার pop ফাংশন ব্যাবহার এর ফলে লিস্ট এর শেষের ১ টি বাদ হয়ে যাবে। ২ বার pop ফাংশন ব্যাবহার এর ফলে লিস্ট এর শেষের ২ টি বাদ হয়ে যাবে।

sob = [10,20,20,30,40,5,60]

po = sob.index(20)

print(po) # index কোন সংখ্যা কত তম ইনডেক্স এ আছে তা দেখতে

po = sob.count(20)

print(po) # count লিস্টে কয় বার আছে তা দেখার জন্য।#24

**range function:** একটি নির্দিষ্ট রেঞ্জের ফাংশন জেনারেট করার জন্য range ফাংশন

EX:

num = list(range(10))

print(num)

n=list(range(2,101,2)) #(কত থেকে চলা শুরু করবে,কত পর্যন্ত চলবে,ব্যবধান কত)

print(n)

# Python problem = Pynative

**User input in list:**

Split**:** লিস্টের প্রত্যেকটি ভ্যালু কে আলাদা করার জন্য

List: কোন স্ট্রিং কা লিস্টে পরিণত করার জন্য

EX:

n= input('Enter a text of number: ')

list = n.split()

sum = 0

for num in list:

    sum = sum + int (num)

print(sum)

upper keyword: ছোট হাতের অক্ষর কে বড় হতে পরিণত করার জন্য

lower keyword: বড় হাতের অক্ষর কে ছোট হাতের অক্ষরে পরিণত করার জন্য

**Matrix:** ম্যাট্রিক্স হলো টু ডাইমেনশনাল অ্যারে বা লিস্ট. এক কথায় লিস্টের মধ্যে লিস্ট.

EX:

matrix = [

    [1,2,3],

    [4,5,6],

]

print(matrix[1][2])

**Dictionary**: ডিকশনারিতে কি এর সাহায্য ডেটা স্টোর করা হয়

Ex:

studentid = { #Dictionarie name

    #all Key

    '101' : "antu",

    '102' : 'SS',

    '103' : 'Sakib',

}

print(studentid.get('101','Not a vlid value'))#('কি য়ের নাম',যখন কি টি থাকবে না তখন কি প্রিন্ট করবে)

**Tuples:** ট্রাফল অনেকটা লিস্টের মত . কিন্তু ট্রাবল এর মধ্যে রাখা মান পরিবর্তন করা যায়. ট্রাফল লিস্টের থেকে দ্রুত. স্লাইসিং সম্ভব

EX:

studentid = (

    ('Priyantu das antu',20,3.00), #Tuples

    'Shuvo Shil',

    'Hafiz Sakib',

)

print(studentid[0])

**Set:** সেট আইটেমগুলো অন অর্ডার থাকবে. তাই আইটেমগুলো কে ইনটেক্স নাম্বারে সাহায্যে অ্যাক্সেস করা যাবে. একই মান দুইবার লিখলে তা প্রিন্ট হবে না.

EX:

num = {1,2,3,4,5}

n = set([4,5,6]) #convert list to set

print(n)

**Stack:** Stack হলো একটি ডাটা স্ট্রাকচার. এটি কাজ করে অনেকটা তাকের মত.

Ex:

book = []

book.append("C")

book.append("C++")

book.append("Java")

print(book)

book.pop()

print('Top book is ',book [- 1])

book.pop()

print('Top book is ',book [- 1])

book.pop()

if not book:

    print('No books left')

**Queue:** Queueহলো একটি ডাটা স্ট্রাকচার . এটি অনেকটা লাইনের মতো কাজ করে.

EX:

from collections import deque

bank = deque(['Anis','Korim','Bijoy'])

print(bank)

bank.popleft()

print(bank)

bank.popleft()

bank.popleft()

if not bank:   #it works Stack was full empty

    print("No person left")

**Function:** দুই ধরনের ফাংশন রয়েছে. লক্ষ্য হলো কোড রিইউজ করা

* Library function
* User define Function

Library function: যে গুলো প্রোগ্রামে আগে থেকে থাকে

User define function: যে ফাংশনগুলো আমরা আমাদের প্রয়োজনে তৈরি করে এগুলোকে বলে

Ex of function:

def add(a,b): #add=name of function

    sum = a + b

    print(sum)

add(10,20)

add(30,20)

Return Function: ফাংশনটা যেখান থেকে এসেছে সেখানে পুনরায় return করা.

Ex:

def add(a,b):

    Sum = a + b

    return Sum

result = add(20,30)   # result এ return রাখা হয়েছে

print(result)

**xargs:** প্যারামিটারের মান পরিবর্তন না করে ডাটা এড করার জন্য ব্যবহার করা হয়

EX:

#xargs

def add(\*numbers):

    print(numbers)

add(10,10)

add(10,20,30)

**xxargs:** এটি ডিকশনারির মতো কাজ করে. এটি কি ভ্যালু পিয়ারে কাজ করে. কিওয়ার্ড আর্গুমেন্টস পাস করা হয়.

Ex:

#xxargs

def students(\*\*details):

    print(details["name"])

students(id=101, name='Anis')

**Debugging:** কোন প্রোগ্রামে ভুল হলে সে ভুল খুঁজে ঠিক করার কাজকে বলে.

Ex: video (40)

def add(\*numbers):

    sum = 0

    for num in numbers:

       sum = sum + num

       return sum

print(add(10,20))

lambda function: যে ফাংশনের কোন নাম থাকবে না .

তৈরি করার নিয়ম:

lambda parameter : expression #expretion = আমরা যে কাজটা করতে যাচ্ছি

EX:

'''

normal code

def calculate(a,b):

    return a\*a + 2\*a\*b + b\*b

print(calculate(2,3))

'''

#lambda function

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

map and filiter function: এটি মূলত ফাংশন ও Iterable(পুনরাবৃত্তিযোগ্য) কোন কিছু নিয়ে কাজ করে. Iterable যেমন লিস্ট.

Map:

def square(x):

    return x\*x

num = [1,2,3,4,5]

result = list(map(square,num))

print(result)

filter: ফিল্টারের ভেতরের কন্ডিশনের সাথে যদি লিস্টে মিল না হয় তাহলে রিমুভ করে দিবে.

Ex:

num = [1,2,3,4,5]

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

print(result)

list comprehension:

লেখার নিয়ম:

[expression for item in list]

Ex:

num = [1,2,3,4,5]

'''

Normal

result = list(map(lambda x: x\*x,num))

print(result)

'''

# comprehensive

result = [x\*x for x in num]

print(result)

zip function: একাধিক লিস্ট কে taple মত করে প্রিন্ট করার জন্য ব্যবহার করা হয়.

EX:

roll = [101, 102, 103, 104, 105, 106]

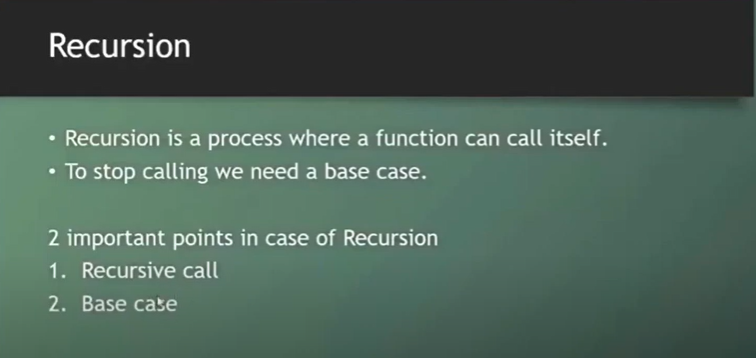
name = ["Antu", "SS", "Skib", "Yeasen", "Abir", "Soboj"]

print(list(zip(roll,name)))

output: [(101, 'Antu'), (102, 'SS'), (103, 'Skib'), (104, 'Yeasen'), (105, 'Abir'), (106, 'Soboj')]

Factorial: সেই সংখ্যা থেকে এক পর্যন্ত শংকর গুণফলকে ফ্যাক্টরিয়াল বলে

N ! =n \* (n-1)!

Recusion: একটি ফাংশন যখন নিজেকে নিজে কল দেবে তখন তাকে বলে Recursion. 

EX:

def fact(n):

    if n== 1:

        return 1

    else:

        return n \* fact(n-1)

print(fact(5))

Reading a file: কোন ফাইল এড করার জন্য ব্যবহার করা হয়

Read = ‘r’

Wright = ‘w’

Read + Wright = “r+w”

EX:

file = open("student.txt","r")

print(file.readable())

file.close

Ex 2:

file = open("student.txt","r")

#print(file.readable())

text = file.read()#ফাইলে থাকা লেখা পড়ার জন্য

print(text)

file.close()

Ex 3:

file = open("student.txt","r")

#print(file.readable())

text = file.read()#ফাইলে থাকা লেখা পড়ার জন্য

print(text)

size = len(text)#সাইজ দেখার জন্য

print(size)

file.close()

writing in a file:

Ex 1:

file = open("student.txt","a") #append

file.write('\nSadi - Lecturer of physics')#কোন লেখা যোগ করার জন্য

file.close()

Ex 2:

file = open("student.txt","w") #overwrite

file.write('\nSadi - Lecturer of Physics')#শুধুমাত্র এই লেখাটি থাকবে w

file.close()

Ex 3:

file = open("student1.txt","w") #overwrite

#যদি সেই ফাইলটি না থাকে তাহলে একটি নতুন ফাইল তৈরি করে ফেলবে সেই নামে

file.write('\nSadi - Lecturer of Physics')#শুধুমাত্র এই লেখাটি থাকবে w

file.close()

Ex 4:

file = open("student1.html","w") #overwrite

#অন্য যেকোনো কোডের ফাইল তৈরি করা যায়

file.write('<h1>Hallo world<\h1>')#শুধুমাত্র এই লেখাটি থাকবে (w)

file.close()

Exception handling: ইউজারের ভুল ইনপুটের কারণে কোন এরর আসে. এক্সেপশন এর পরের লাইন থেকে আর কাজ করবে না.

* যেখানে সমস্যা হতে পারে তা try ব্লকের মধ্যে রাখবো
* এক্সেপশন গুলো Exception এর মধ্যে রাখবো
* কোড থাকুক বা না থাকুক finally মধ্যে থাকা কোড অবশ্যই কাজ করবে

Ex:

try:

  list = [20,0,30]

  result = list[0] / list[1]

  print(result)

  print("Done")

except ZeroDivisionError:

  print("Divided by Zeroo is not possable")

finally:

  print("Successful")

Multiple Exception:

একের অধিক এক্সেপশন এক লাইনও দেওয়া যায়

Ex:

try:

  num1 = int(input("Enter 1st number: "))

  num2 = int(input("Enter 2nd number: "))

  result = num1 / num2

  print(result)

except (ZeroDivisionError,ValueError):

  print("You have entered incorrect input")

finally:

  print("Thanks !!")

Swaping: একটির মান আরেকটিতে বসা. অর্থাৎ মান অদলবদল করে দেওয়া.

Ex:

a = 20

b = 10

a, b = b, a

print('A= ',a)

print('B= ',b)

**Class:** ক্লাস হল একটি প্রোগ্রামের কমন প্রপার্টি

**Object:** কোন ক্লাসের অবজেক্ট হল সেই ক্লাসের সকল প্রপার্টি ধারন করা একটি ভ্যারিয়েবল।

EX:

class Student: # create a cllass

    roll = ""

    gpa = ""

rohim = Student() #creat a object

print(isinstance(rohim,Student)) #chak creat object or not

rohim.roll = 101

rohim.gpa = 3.00

print(f"Rohim Roll: {rohim.roll}, GPA: {rohim.gpa}")

korim = Student()

korim.gpa = 3.00

korim.roll = 1120

print(f"Korim Roll: {korim.roll}, Korim gpa: {korim.gpa}")

**Mathod:** উপরের প্রোগ্রামটিতে roll এবং gpa প্রিন্ট করার জন্য বারবার print ব্যবহার করা হয়. এইরকম একই কাজ বারবার না করার method ব্যবহার করা হয়.

Ex:

class Student: # create a cllass

    roll = ""

    gpa = ""

    def display(self):

        print(f"Roll: {self.roll}, GPA: {self.gpa}")

rohim = Student() #creat a object

rohim.roll = 101

rohim.gpa = 3.00

rohim.display()

korim = Student()

korim.gpa = 3.20

korim.roll = 1120

korim.display()

সাহায্যে সেট করতে পারি roll, gpa এর মান

class Student: # create a cllass

    roll = ""

    gpa = ""

    def set\_value(self,roll,gpa):

        self.roll = roll

        self.gpa = gpa

    def display(self):

        print(f"Roll: {self.roll}, GPA: {self.gpa}")

rohim = Student() #creat a object

rohim.set\_value(110,2.40)

rohim.display()

korim = Student()

korim.set\_value(120,2.75)

korim.display()

**Constructor:** অবজেক্ট এ মান সেট করে দেওয়া যায়. কনস্ট্রাক্টর কে আলাদাভাবে কল দেওয়ার প্রয়োজন হয় না.

Ex:

class Student: # create a cllass

    roll = ""

    gpa = ""

    def \_\_init\_\_(self,roll,gpa):

        self.roll = roll

        self.gpa = gpa

    def display(self):

        print(f"Roll: {self.roll}, GPA: {self.gpa}")

rohim = Student(110,3.75) #creat a object

rohim.display()

korim = Student(120,3.80)

korim.display()

Mathod overriding: একই ধরনের ক্লাসে দুই ধরনের ডাটা স্টোর করার জন্য ব্যবহার করা হয়.

Ex:

class Phone:

     def \_\_init\_\_(self):

          print("I am in Phone class")

class Samsang(Phone):

     '''

     def \_\_init\_\_(self):

          print("I am in Phone class")

     '''

     def \_\_init\_\_(self):

          super().\_\_init\_\_()

          print("I am in Samsang class")

S = Samsang()

Inheritance: একটি ক্লাসের বৈশিষ্ট্য অন্য ক্লাসে নিয়ে যাওয়ার জন্য

#Sub class

class Phone:

    def call(self):

        print("You can call")

    def massage(self):

        print("You can massage")

#Super class

class Samsang(Phone):#Inheritence

    #call

    #massage

    def photo(self):

        print("You can take photo")

s = Samsang()

s.massage()

s.call()

s.photo()

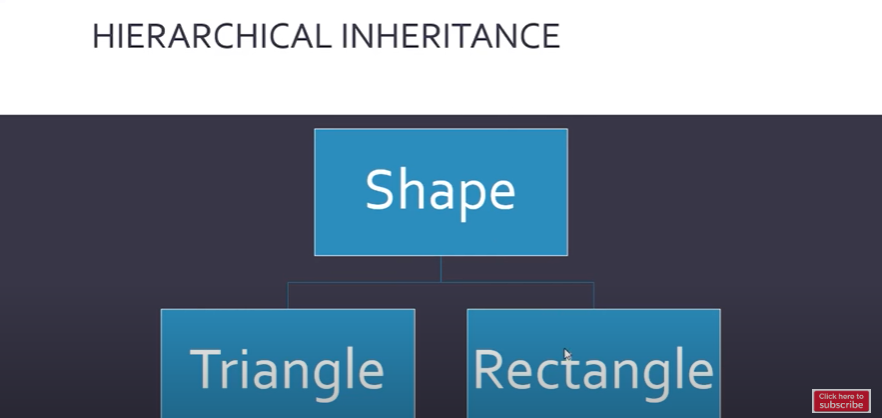
#print(issubclass(Samsang,Phone)) #সাব ক্লাস কিনা চেক করার জন্য

Type of Inheritance:

1. Hierarchical Inheritance.
2. Multi-Level Inheritance.
3. Multiple Inheritance.

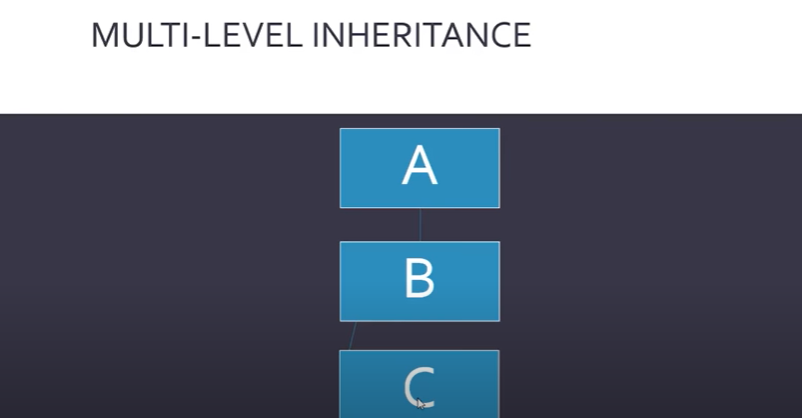
Hierarchical Inheritance: একটি ক্লাসকে অনেক ক্লাস ইনহেরিট করতে পারবে

Ex:



Multi-Level Inheritance: একটি ক্লাসকে অন্য একটি ক্লাস ইনহেরিট করতে পারবে| আবার ইনহেরিট কৃত ক্লাস টিকে অন্য আরেকটি ক্লাস ইনহেরিট করতে পারবে|

Ex:



Code:

#Multi-Level Inheritance

class A:

    def display1(self):

        print("I am inside A class")

class B(A):

    def display2(self):

        print("I am inside B class")

class C(B):

    def display3(self):

        print("I am inside C class")

ob1 = C()

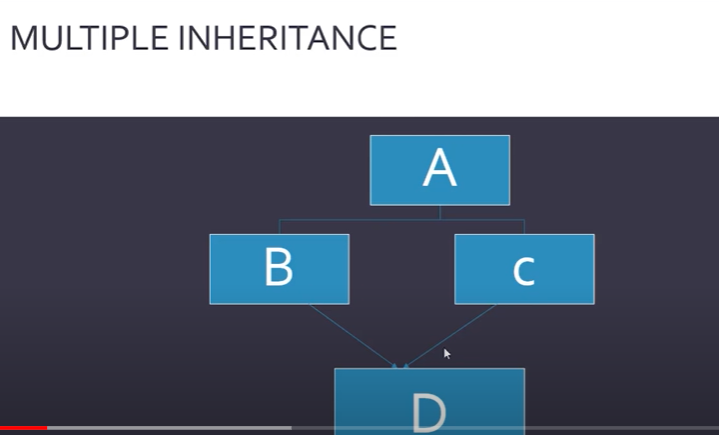
ob1.display1()

ob1.display2()

ob1.display3()

Multiple Inheritance: একটি ক্লাসকে অন্য কতগুলো ক্লাস ইনহেরিট করবে| এবং ওই ইনহেরিট কৃত ক্লাস গুলোকে আবার ইনহেরিট করবে অন্য আরেকটি ক্লাস|

Ex:



Code:

#Multiple Inheritance

class A:

    def display1(self):

        print("I am inside A class")

class B:

    def display2(self):

        print("I am inside B class")

class C(A,B):

    def display(self):

        print("I am inside C class")

ob1 = C()

ob1.display()

**Abstraction:**

**Ex:**

from abc import ABC,abstractmethod

class Shape(ABC):

    def \_\_init\_\_(self,dim1,dim2):

        self.dim1 = dim1

        self.dim2 = dim2

    @abstractmethod

    def area(self):

        pass

class Triangle(Shape):

    def area(self):

        area = 0.5 \* self.dim1 \* self.dim2

        print("Area of Triangle : ",area)

class Rectangle(Shape):

    def area(self):

        area = self.dim1 \* self.dim2

        print("Area of Rectangle : ",area)

t1 = Triangle(20,30)

t1.area()

t2 = Rectangle(20,30)

t2.area()

**Polymorphism**: যখন কোন কিছুর বহু রূপ হয়

Ex:

print(len("Priyantu Das"))

print(len([10,20,30]))

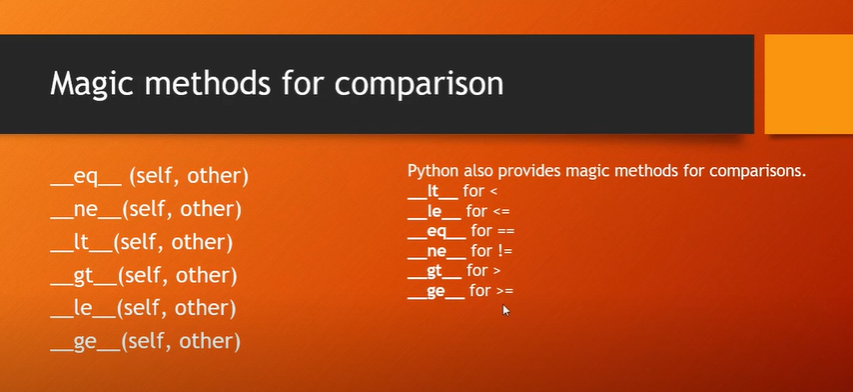
def add(x, y, z=0):

    return x + y + z

print(add(2, 3))

print(add(2, 3, 4))

**[pragties: GeeksforGeeks]**

**Megic Methods:** 

**Code:**

class Bike:

    def \_\_init\_\_(self, name, color):

        self.name = name

        self.color = color

    def \_\_eq\_\_(self, other): #অবজেক্ট এর ভ্যালু কমপেয়ার করার জন্য

        return self.name == other.name and self.color == other.color

    def \_\_str\_\_(self): #অবজেক্ট এর মান প্রিন্ট করার জন্য

        return(f"Name = {self.name}, Color= {self.color}")

    def display(self):

        print(f"Name = {self.name}, Color= {self.color}")

bike1 = Bike("Yamaha R15","Blue")

bike2 = Bike("Yamaha R15","Blue")

print(bike1==bike2)

**Moduels**: মডিউল হল একটি পাইথন ফাইল যেখানে পাইথন কোড থাকবে| মডিউলে আমরা সাধারণত ফাংশন ক্লাস ও ভেরিয়েবল ডিক্লেয়ার করে থাকি যেগুলো আমরা পরবর্তীতে অ্যাক্সেস করে থাকি|

Ex:

def tringle\_area(b,h):

    print(f"Area off tringle is : {0.5\*b\*h}")

def rectangle\_area(a,b):

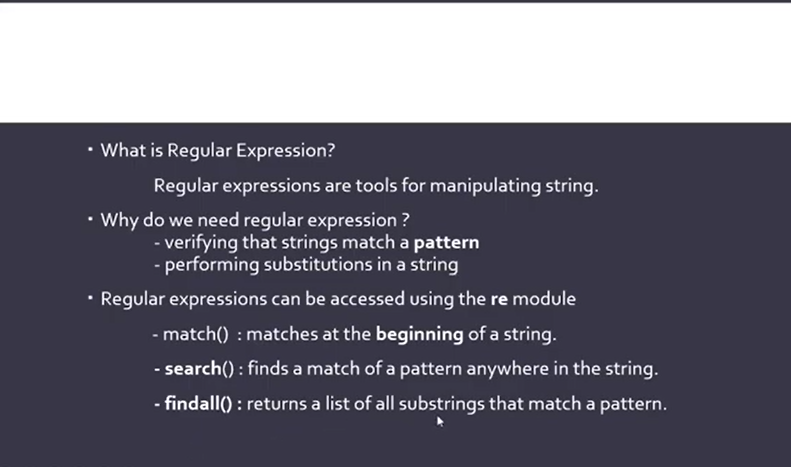
    print(f"Area of Rectangle is : {a\*b}")

from Modules import tringle\_area,rectangle\_area #press (ctrl+space)

tringle\_area(10,20)

rectangle\_area(10,20)

**Regular Expreassion**:



কোন কিছু মিলে কি না দেখার জন্য

import re

pattern = r"Colour"

if re.match(pattern,"Colour is a colour, I love read colour"):

    print("Match")

else:

    print("Not Matched")

প্যাটার্নে কোন কিছু আছে কিনা দেখার জন্য

import re

pattern = r"colour"

if re.search(pattern,"Read is a colour, I love read colour"):

    print("Match")

else:

    print("Not Matched")

প্যাটার্ন খুঁজে বের করার

import re

pattern = r"c"

print (re.findall(pattern,"Read is a colour, I love read colour"))

প্যাটার্ন খুঁজে নিয়ে রিপ্লেস করার জন্য

import re

pattern = r"Red"

text1 = "My favourite colour is Red.I love blue"

text2 = re.sub(pattern,"Green",text1)

print(text2)

output:

My favourite colour is Green.I love blue

কয়বার রিপ্লেস করবে তাও বলে দিতে পারি

Count

import re

pattern = r"colour"

text1 = "My favourite colour is Red.I love blue colour"

text2 = re.sub(pattern,"color",text1,count=1)

print(text2)

output: My favourite color is Red.I love blue colour

**Meta Chreactor:**

**EX 1:**

import re

pattern = r"colo.r"#যতটি ডট ততটি ক্যারেক্টারের পর ডটের পরের আলফাবেট টি রয়েছে

if re.match(pattern,"colour"):

    print("Matched")

**EX 2:**

import re

pattern = r"^colo..r$"#^পরে অবশ্যই ক্যারেক্টারগুলো থাকতে হবে..$এটির আগের ক্যারেক্টার টি শেষ হতেই হবে

if re.match(pattern,"colomar"):

    print("Matched")

**EX 3:**

import re

pattern = r"a\*"#\*এর আগে থাকা ক্যারেক্টারটা প্যাটার্নের থাকতেও পারে নাও থাকতে পারে(0 or more)

if re.match(pattern,"colomar"):

    print("Matched")

**Ex 4:**

import re

pattern = r"a+"#+এর আগে থাকা ক্যারেক্টারটা প্যাটার্নে একবার বা তার থেকে বেশি থাকতে হবে(1 or more)

if re.match(pattern,"colomar"):

    print("Matched")

**EX 5:**

import re

pattern = r"a+b"#+এর আগের টির পর অবশ্যই এর পরেরটি থাকতে হবে

if re.match(pattern,"abcolomar"):

    print("Matched")

**EX 6:**

import re

pattern = r"ice(-)?crem"#?এর আগে থাকা চিহ্নটি একবার বা না থাকলেও প্রোগ্রাম কাজ করবে(0or1)

if re.match(pattern,"icecrem"):

    print("Matched")

**EX 7:**

import re

pattern = r"a{1,3}$"#সর্বোচ্চ কয়টি থেকে কয়টি পর্যন্ত রাখতে পারবে তা দেখানোর জন্য

if re.match(pattern,"aaa"):

    print("Matched")

**Character class:**

**Ex1:**

import re

pattern = r"[aeiou]"#কোন একটি সেটের কিছু স্ট্রিং আছে কিনা তা চেক করার জন্য

if re.match(pattern,"anhfeou"):

    print("Matched")

**EX2:**

import re

pattern = r"[0-9][a-z][A-Z]"#কোন সেটের রেঞ্জ এর ভেতর কিছু আছে কিনা তাও চেক করা যায়

#মাল্টিপল সেট ও ব্যবহার করা যায়

if re.match(pattern,"0aEanhfeou"):

    print("Matched")