## PYTHON

* Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

**What can Python do?**

* Python can be used on a server to create web applications.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* AI and ML
* Python is used to perform Data Analytics

**Why Python?**

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written.
* Python can be treated in a procedural way, an object-oriented way or a functional way.

**Hierarchy of programming :**

* Project
  + Package
    - Subpackage
      * Module
        + OOP (Class , Method , Constructor )

Functions (Method / variable / FBP)

**Dynamic inference :**

* Python is a dynamic inference language -(Which identifies its own variable datatype )
* Python is a strongly typed programming language

#the below prog shows that it automatically finds a as int

a=12

strings="Hi BYe"

print(strings + a);

|  |  |
| --- | --- |
| Static inference | Dynamic inference |
| name="Akshaya Baalaji " | name :str="Akshaya Baalaji "  #this is for the definition that name is a variable will be of type string Even if you enter the integer in it it can identify it because of the dynamic inference in python  Ex:  number:str=9  print(type(number)) // type int |

**Interpreter vs Compiler :**

INTERPRETER :

* It evaluates line by line
* if there is a error in a function
  + it won’t be found out till the function is called

COMPILER :

* It evaluates all the lines at once
* if there is a error in a function
  + it still shows it even the function is not called

**Who use Python today :**

1. Google
2. Facebook
3. Instagram
4. Spotify
5. Quora
6. Netflix
7. Dropbox
8. Reddit

**OPERATORS :**

Types of operators in python

* Arithmetic operators
* Assignment operators
* Comparison operators
* Logical operators
* Identity operators
* Membership operators
* Bitwise operators
* Ternary operator

1. Arithmetic

|  |  |  |
| --- | --- | --- |
| + | Addition | x + y |
| - | Subtraction | x - y |
| \* | Multiplication | x \* y |
| / | Division | x / y |
| % | Modulus | x % y |
| \*\* | Exponentiation | x \*\* y |
| // | Floor division | x // y |

1. Assignment

* used to assign the value to variable
* usually = (or) operator = and the operator can be (+ , - , \* , / , \*\* , ^)

|  |  |  |
| --- | --- | --- |
| = | x = 5 | x = 5 |
| += | x += 3 | x = x + 3 |
| -= | x -= 3 | x = x - 3 |
| \*= | x \*= 3 | x = x \* 3 |
| /= | x /= 3 | x = x / 3 |
| %= | x %= 3 | x = x % 3 |
| //= | x //= 3 | x = x // 3 |
| \*\*= | x \*\*= 3 | x = x \*\* 3 |
| &= | x &= 3 | x = x & 3 |
| |= | x |= 3 | x = x | 3 |
| ^= | x ^= 3 | x = x ^ 3 |
| >>= | x >>= 3 | x = x >> 3 |
| <<= | x <<= 3 | x = x << 3 |

1. Comparison / Relational

* (== , >= , <= , != , > , < ) return the boolean result

|  |  |  |
| --- | --- | --- |
| = = | Equal | x == y |
| != | Not equal | x != y |
| > | Greater than | x > y |
| < | Less than | x < y |
| >= | Greater than or equal to | x >= y |
| <= | Less than or equal to | x <= y |

1. Bitwise

* same like logical operator
* costly to use because it does the comparision bit by bit returning the true or false
* & , | , ^ , ~ same as logical operator

|  |  |  |  |
| --- | --- | --- | --- |
| & | AND | Sets each bit to 1 if both bits are 1 | x & y |
| | | OR | Sets each bit to 1 if one of two bits is 1 | x | y |
| ^ | XOR | Sets each bit to 1 if only one of two bits is 1 | x ^ y |
| ~ | NOT | Inverts all the bits | ~x |
| << | Zero fill left shift | Shift left by pushing zeros in from the right and let the leftmost bits fall off | x << 2 |
| >> | Signed right shift | Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off | x >> 2 |

1. Logical

* Return only Boolean value (and , or , not)

|  |  |  |
| --- | --- | --- |
| and | Returns True if both statements are true | x < 5 and x < 10 |
| Or | Returns True if one of the statements is true | x < 5 or x < 4 |
| Not | Reverse the result, returns False if the result is true | not(x < 5 and x < 10) |

1. Identity

|  |  |  |
| --- | --- | --- |
| is | Returns True if both variables are the same object | x is y |
| is not | Returns True if both variables are not the same object | x is not y |

1. Membership

|  |  |  |
| --- | --- | --- |
| in | Returns True if a sequence with the specified value is present in the object | x in y |
| not in | Returns True if a sequence with the specified value is not present in the object | x not in y |

1. Ternary operator

Example :

n=input("Enter the input ")

add=10

summ=add+n if isinstance (n, int) else int(n)+add

print("Your addition is " , summ)

**Variable Names**

* A name or identifier to refer the memory location of a object or variable not start with special character except \_
* **Rules for naming a variable :**
* A variable name must start with a letter or the underscore character
* A variable name cannot start with a number
* Var name cant be more than 79 characters
* A variable name can only contain **alpha-numeric characters** and underscores (A-z, 0-9, and \_ )
* Variable names are **case-sensitive** (age, Age and AGE are three different variables)
* A variable name cannot be any of the Python keywords.
* Multi Words Variable Names - Variable names with more than one word can be difficult to read There are several techniques you can use to make them more readable

1. Camel Case - Each word, except the first, starts with a capital letter: myVariableName = "John"
2. Pascal Case - Each word starts with a capital letter: MyVariableName = "John"
3. Snake Case - Each word is separated by an underscore character:

my\_variable\_name = "John"

* Multi assignment : Multiple variable assigned on single line

Example :

a,b,c=1,2,3

print(a)

print(b)

print(c)

**Python Start of code :**

#!/usr/bin/python3 : #! This is Shebang that tells system the following path is interpreter and /usr/bin/python3 here is where the interpreter resides

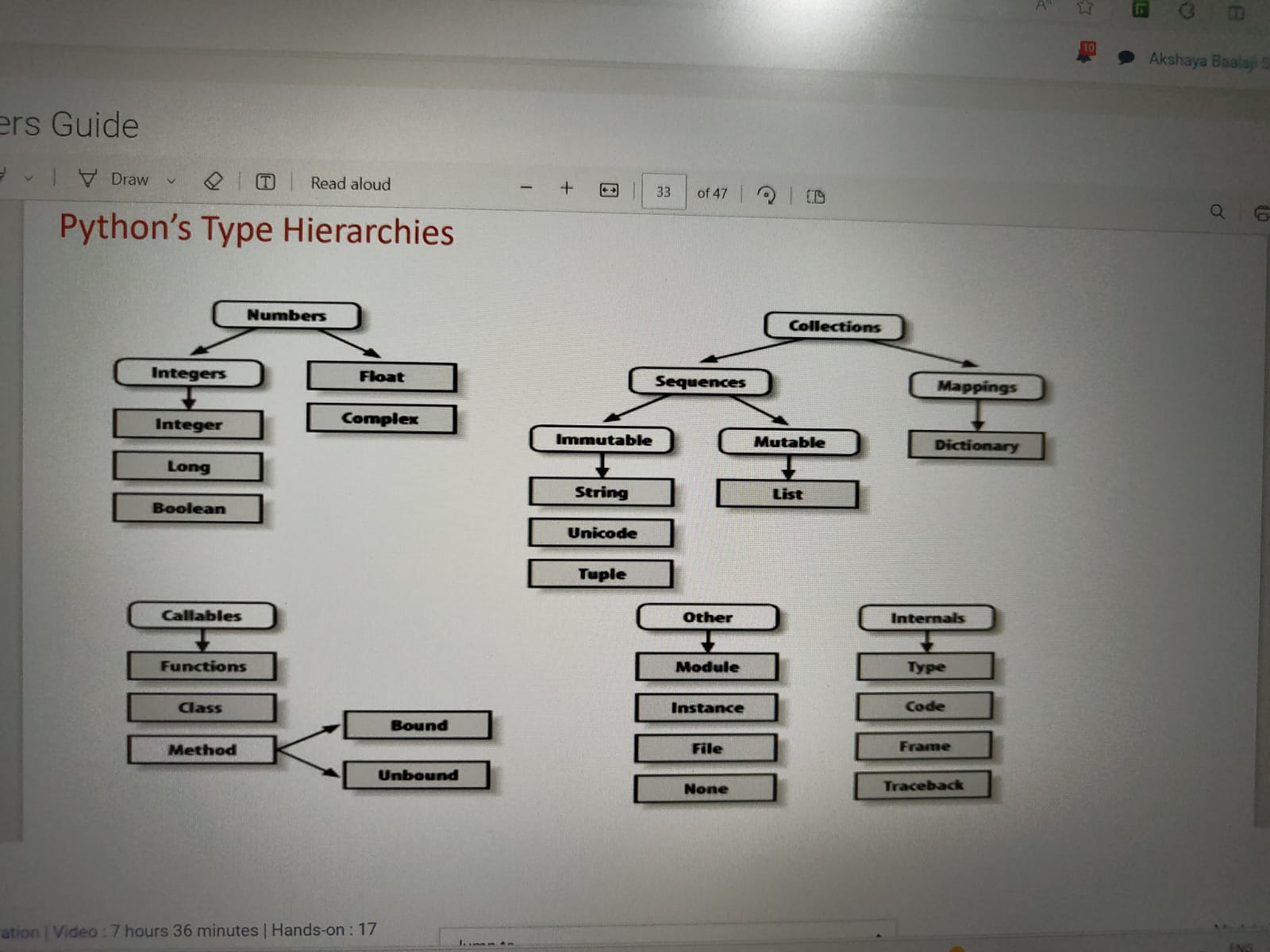
**Precedence :**

1. Parenthesis ()
2. Exponent \*\*
3. Mul \* , Div / , Remainder %
4. Addition + and Subtraction –

**String :**

* #string is a iterable type and
* #it is a collection of charecter in sequence
* If a variable value can be changed it is mutable and if not immutable

|  |  |  |
| --- | --- | --- |
| Concatenation  Repetition  Indexing  Negative Indexing  Slicing  Size  Search  Mutable | print("Hello" + "World")  print("Hello" , "World")  print("Hi"\*3)  a="Akshaya Baalaji "  print("Indexing a[0]", a[0])  print("Negative Indexing a[-1]", a[-1])  print("Slicing a[0:5]" , a[0:5])  print("Size : LEngth " , len(a))  print("Searching a present in string " , "a" in a )  list =[1,2,"Akshaya"]  list[2] = "Baalaji"  #the above can be done because list , set , dictionary are mutable  strings="Akshaya Baalaji"  strings[0]='A'  #this is not possible because string is immutable | HelloWorld  Hello World  HiHiHi  Indexing a[0] A  Negative Indexing a[-1]  Slicing a[0:5] Aksha  Size : LEngth 16  Searching a present in string True  #error |



**Control Structure / conditional structure :**

If elif else and while , for , break , continue

|  |  |
| --- | --- |
| If elif else  Def : Checks the condition whether the stmt is true or not if true enter into the loop | a=int(input("Enter a number "))  b=int(input("Enter another number "))  if(a>b) :  print("A is greater " )  elif(a==b) :  print("You have enter the same number ")  else:  print("B is greater") |
| For | a=int(input("Enter a number "))  b=int(input("Enter another number "))  #To loop through a set of code a specified number of times, we can use the range() function  #range (1,100,2) : 1-start , 100-end , 2-skipby  for i in range(a,b):  print(i)  for i in range(a,b):  print(i , end =" " )  print("Star program 1")  aa=int(input("Enter number of lines "))  for i in range (0,aa+1) :  for j in range (0,i) :  print("\*" , end=" ")  print("")  print("Star program 2")  for i in range (0,aa+1):  for j in range (aa,i,-1):  if(j>i) :  print("\*" , end=" ")  print("")  print("Star progrma 3 ")  for i in range (aa+1,0,-1):  for j in range (0,aa+1,1):  if(j>=i):  print("\*" , end=" ")  else:  print(" " , end=" ")  print("")  print(“Star program 4 ”)  list1="Akshay"  for i in range (0,len(list1)+1,1):  for j in range (0,i,1):  print(list1[j], end=" ")  print() |
| While | a=int(input("Enter a unmber "))  i=1  while i<=a:  print(i)  i=i+1 |

**if – control structure understanding :**

1. if :

there can be only one if without else - true

there can be only one else and elif without if - false

possible conditional structure in if :

1.if

2.if else

3. if elif else

Ex :

inputt=int(input("Enter a number "))

if(inputt==0):

print("Zero")

if(inputt==1):

print("One")

if(inputt==2):

print("Two")

if(inputt==3):

print("Three")

using elif :

inputt=int(input("Enter a number "))

if(inputt==0):

print("Zero")

elif(inputt==1):

print("One")

elif(inputt==2):

print("Two")

elif(inputt==3):

print("Three")

else :

print("SOme other numbers ")

2. pass : #used for passing / skipping the current iteration

Ex :

if(a==1):

print("1")

else :

pass #skip the iteration

**Practical Irfan exercise for greatest of three numbers :**

1. #using list built in functions

make\_list=[a,b,c] # making the list from the variables

print(max(make\_list));

#we can take the greatest by sorting and taking the last element in list

make\_list.sort(); #sort will sort the list

print("Taking the last elemrent from the list ",make\_list[-1])

1. #using simple conditional structure

a=10

c=10

b=30

if a==b and a==c :

print("All are same ")

elif a>=b and a>=c :

print(a, " is the greatest ")

elif b>=a and b>=c :

print(b, " is the greatest ")

else :

print(c, " is the greatest ")

1. #using nested conditional structure

a=10

b=10

c=30

if a==b and b!=c:

if a>=b and a>=c :

print(a , " is the greatest ")

elif b>=a and b>=c :

print(b," is the greatest ")

else :

print(c , " is the greatest ")

else :

print("All are same ")

Some deep :

|  |
| --- |
| Prg : for understanding the for constrain  #control structure if elif else  """  a=int(input("Enter a number "))  if(a%2==0):  print("Even")  elif(a%2!=0):  print("ODD");  else:  print("Enter the number correctly ")  """  #control structure for  n=int(input("ENetr a number of start for making the star pattern "))  """  for i in range (1,n+1):  #print(i) 1 2 3 4 5  for j in range (0,i): # j = 0 1 2 3 4 when i =5  print("\*" , end =" ")  print()  \*  \* \*  \* \* \*  \* \* \* \*  \* \* \* \* \*  """  """  for i in range (n,0,-1):  for j in range (0,i):  print("\*" , end =" ")  print()  \* \* \* \* \*  \* \* \* \*  \* \* \*  \* \*  \*  """  """  for i in range (1,n+1):  for j in range (0,i):  print(" " , end =" ")  for k in range (n+1, i ,-1):  print("\*" , end =" ")  print()  \* \* \* \* \*  \* \* \* \*  \* \* \*  \* \*  \*  """  """  for i in range (1,n+1):  for j in range (0,i):  print(" " , end =" ")  for k in range (n+1, i ,-1):  print("\*" , end =" ")  for l in range(n+1 , i+1,-1):  print("\*" , end=" ")  print()  \* \* \* \* \* \* \* \* \*  \* \* \* \* \* \* \*  \* \* \* \* \*  \* \* \*  \*  """  #control structure while  while n<20:  print(n, end= " ")  n=n+1  print(n) |

**Python Indentation**

* Indentation refers to the spaces at the beginning of a code line.
* Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.
* Python uses indentation to indicate a block of code.
* Python will give you an error if you skip the indentation
* The number of spaces is up to you as a programmer, the most common use is 4, but it has to be at least 1.

**Python Comments**

* Comments can be used to make the code more readable.
* Comments starts with #, and Python will ignore them
* Python does not really have a syntax for multiline comments . To add a multiline comment you could insert # for each line:
* Python will ignore string literals (String inside quotes) that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:

Ex:

#This is the comment

"""

This is the long line comment

"""

**Python Variables :**

* Variables are containers for storing data values.
* In Python, variables are created when you assign a value to it
* Python has no command for declaring a variable.

Example : a=12 , b= “string” , c=34.657

* If you want to specify the data type of a variable, this can be done with casting.

Example :

x = str(3)    # x will be '3'  
y = int(3)    # y will be 3  
z = float(3)  # z will be 3.0

* You can get the data type of a variable with the type() function.

Example : a=10 type(a)#int

* String variables can be declared either by using single or double quotes

Example :

x = "John"  
# is the same as  
x = 'John'

* Variable names are case sensitive

Example :

a = 4  
A = "Sally"  
#A will not overwrite a

* Multi line value in python

Example :

#multi line value :

strings=""" HI

Hello

Goodbye """

print(strings)

|  |
| --- |
| Prg:  var1=3  var2 =34.56  var3 = "Akshaya Baalaji "  var4='c'  var5=12345678965432456789076543  var6=True  def identifytype():#creating a function or defining a function  global var1  global var2  global var3  global var4  global var5  global var6  print("The type of var1" , type(var1))  print("The type of var2" , type(var2))  print("The type of var3" , type(var3))  print("The type of var4" , type(var4))  print("The type of var5" , type(var5))  print("The type of var5" , type(var6))    identifytype();#function call |

**STRINGS**

* Strings in python are surrounded by either single quotation marks, or double quotation marks.
* 'hello' is the same as "hello"
* You can assign a multiline string to a variable by using three quotes:
* String index starts with 0
* R or r represent the string is in raw format

Ex:

|  |  |
| --- | --- |
| a = """Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua."""  print(a) | a = '''Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'''  print(a) |

* Strings in Python are arrays of bytes representing unicode characters

Ex : a = "Hello, World!"

print(a[1]) #e

* Looping in Strings can also be done .

Ex: for x in "banana":

print(x)

* Slicing can also done on String in python

Ex : a = "Hello, World!"

Print(a[0:7]) #Hello W

* To get the length of a string, use the len() function.

Ex: a = "Hello, World!"

print(len(a))

* To check if a certain phrase or character is present in a string, we can use the keyword **in**

Ex: str='Hi hello goodbye'

if 'hello' in str :

print("Super ")

* To check if a certain phrase or character is NOT present in a string, we can use the keyword **not in**

Ex: str='Hi Hello goodbye'

if 'hello' not in str :

print("Super ")

* To find the length of the string

Ex: b=len(a)

|  |
| --- |
| String comparison :  #string comparision  string1="Akshaya Baalaji"  string2='Akshaya Baalaji Senthilraj'  if(string1 == string2):  print("Both are same compared by == assignment operator ")  if(string1 != string2):  print("Both are Different compared by != assignment operator ")  if(string1 is string2):#here the identity operator compare the both string1 and string2 whether all charecters are same in string1 and string2  print("Both are same compared by is identity operator ")  if(string1 is not string2):  print("Both are Different compared by is not identity operator ")  if(string1 in string2): # here the membership operator compare the the first char of string1 and second char of string 2 till the end of string1  print("Both are same compared by in membership operator")  if(string1 not in string2):  print("Both are Different compared by not in membership operator")  //Additional information  Membership operator compares the first string to second string ex  A =”Akshaya Baalaji”  B= “Akshaya Baalaji ”  If (A in B ) : //true  If(A is B ) : // false  Identity operator compares both the string with each other |

**Diff between raw string and normal string :**

|  |  |
| --- | --- |
| Raw String | Normal String |
| regular\_string = "C:\new\folder" # This may lead to issues because \n is interpreted as a newline character | raw\_string = r"C:\new\folder" # The 'r' prefix indicates a raw string, so \n is treated as literal backslashes |

**String Slicing**

* You can return a range of characters by using the slice syntax

Ex: b = "Hello, World!"

print(b[2:5])

Ex : b = "Hello, World!"

print(b[:5]) (0 – 4 )

Ex : b = "Hello, World!"

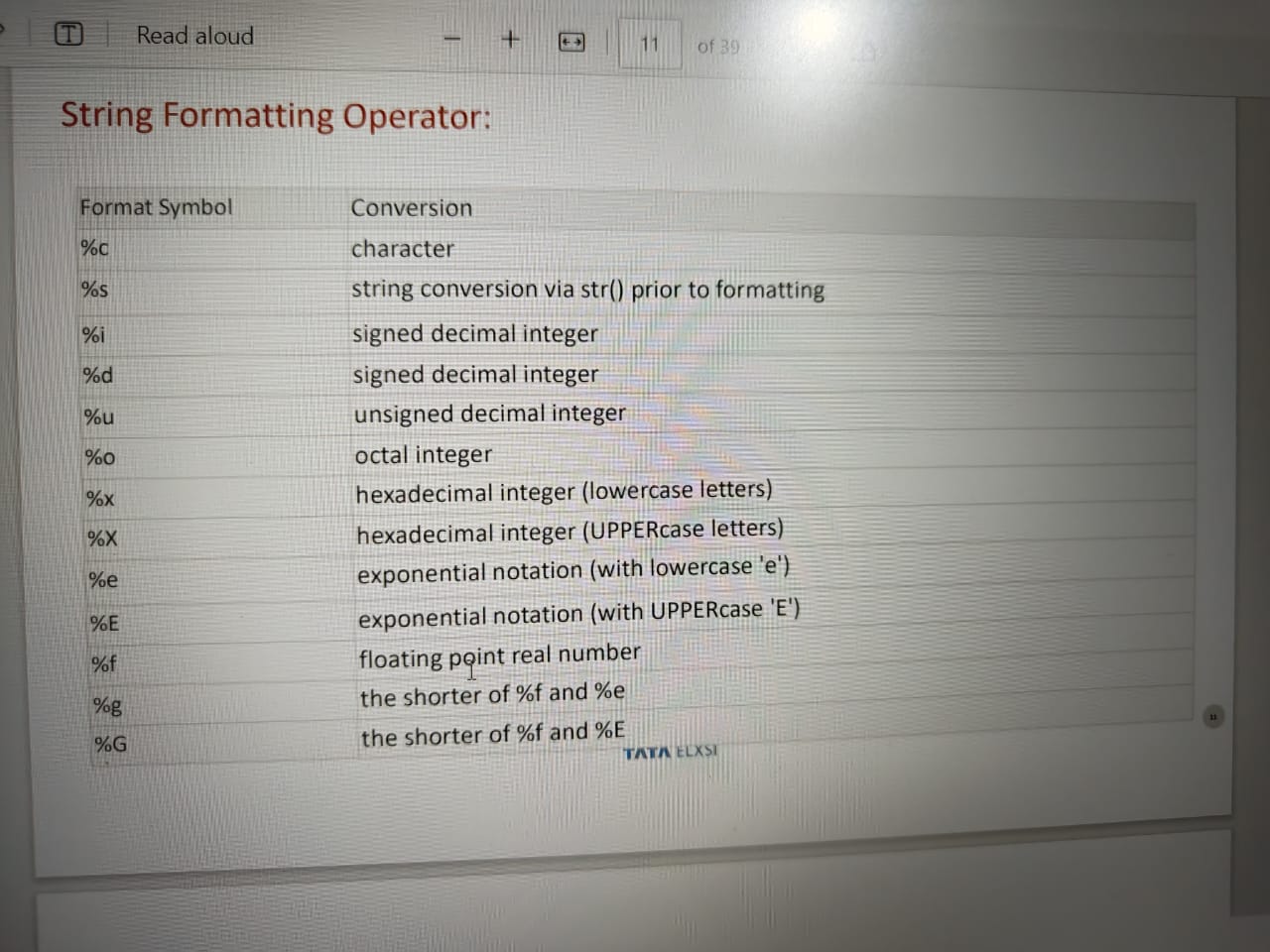
print(b[-5:-2]) (Counts – from the last )

|  |  |
| --- | --- |
| str='Hi Hello bye'  print(str[:9])  print(str[0:])  print(str[3:5])  print(str[-7:-2])  print(str[:10]) | Hi Hello  Hi Hello bye  He  llo b  Hi Hello b |

**String Formatting : Using format() function**

Ex :

|  |  |
| --- | --- |
| a="Hello"  b = "Akshay"  c="baalaji "  age=21  print("Hi my name is %s and my lst name is %s and my age is %d %s YOu are logged in " %(b,c,age,a))  print ("OR")  d="Hi my name is {} and my lst name is {} and my age is {} {} YOu are logged in"  print(d.format(b,c,age,a))  print(“AND”)  e ="Hi He is {1} and {1} told {2} at {0}"  print(e.format(age,b,a)) | Hi my name is Akshay and my lst name is baalaji and my age is 21 Hello YOu are logged in  OR  Hi my name is Akshay and my lst name is baalaji and my age is 21 Hello YOu are logged in  AND  Hi He is Akshay and Akshay told Hello at 21 |



**String Methods** :

|  |  |
| --- | --- |
| a=" Hi Hello \' Akshaya Baalaji \' Goodbye "  print(str)  print("Strip" , a.strip())  print("Upper" , a.upper())  print("Lower" , a.lower())  print("Replace" , a.replace('H' , 'I'))  print("Split" , a.split(" "));  print(a.startswith("Hi")) | Strip Hi Hello ' Akshaya Baalaji ' Goodbye  Upper HI HELLO ' AKSHAYA BAALAJI ' GOODBYE  Lower hi hello ' akshaya baalaji ' goodbye  Replace Ii Iello ' Akshaya Baalaji ' Goodbye  Split ['', 'Hi', 'Hello', "'", 'Akshaya', 'Baalaji', "'", 'Goodbye', '']  False |

|  |
| --- |
| **Prg: String**  """  string1="C:hello\hi\newfolder\file"  string2=r"C:hello\hi\newfolder\file" # the string is in raw format  print(string1)  print(string2)  """  #string formatting  """  c="Senthilraj"  a="Baalaji"  b="Akshaya"  print("Hi everyone my name is %s and my middle name is %s and my father name is %s " %(b,a,c))  string = "Hi I am {1} and my middle name is {0} and my father name is {2} "  print(string.format(a,b,c))  """  #string methods  #strip removes the extra whiteapaces from the staret and end  string=" HI I am \" Akshaya Baalaji Senthilraj \" I am working as an Engineer in Tata Elxsi "  print("Stripped string ",string.strip())  print("Normal String ",string)  #len  print("Length of string " ,len(string))  #string to upper case  print("The string converted to upper case ", string.upper())  #string to lower case  print("The string converted to lower case " ,string.lower())  #Replace  print("Replace I with II " ,string.replace('I' , 'II'))  #Split  print("Spliting done " , string.split())  #starts with  print("Starts with HII " ,string.startswith("HII"))  #ends with  print("Ends with Elxsi " ,string.endswith("Elxsi"))#false because of space inbetween |

**Python Collection types:**

**Before types understand Below definitions :**

* Iterable (looping )
* mutable (changable)
* updatable (modifyable )
* resizable (add/delete element )
* accessable (using index , position ,value , size )

**Different types :** Arranged based on the priority

1.List []

2.Dict {}

3.Tuple ()

4.Set {}

**Tuples :**

* Store multiple item on Single variable
* Can be of same or Different datatype
* A tuple is a collection which is ordered and unchangeable.
* Tuples are written with round brackets.
* tuples are indexed, they can have items with the same value multiple times

**Tuple Operation :**

tuple1 = (1 , 1 , 2 , 2 , 3 , 3 , 4 , 5 , "Akshaya Baalaji" , "Senthil raj" , True ,False)

tuple2 = (23,"Baalaji")

print("Tuple 1 : " , tuple1)

print("Tuple 2 : " , tuple2)

#resizable : rather we can create a new tuple and add the element

#tuple1.\_\_add\_\_(45,56) #cant alter the size of the tuple

tuple3 = tuple1.\_\_add\_\_(tuple2)

#or

tuple4 = tuple1 + tuple2

print("Tuple 3 = Tuple1.\_\_add\_\_(Tuple2) : ",tuple3)

print("Tuple 4 = Tuple 1 + Tuple 2 : ",tuple4)

#changable : we cant change the tuple

#tuple1[0] = 45

#delete : we cant delete the element of the tuple

#tuple1.\_\_remove\_\_(45)

element\_to\_remove = 2

#as the tuple is not mutable we must create a new tuple and reomve the element

tuple5 = tuple(item for item in tuple3 if item != element\_to\_remove)

print("Tuple 5 = Remove 2 from the tuple 3 : ",tuple5)

Ex :

|  |  |
| --- | --- |
| a=("hi", " Hello " , "Good Bye ")  print(a)  print(type(a))  b=(1,True ,"Akshay", 6.765)  print(b[2])  #b[2]=”Aksa” error arises here  print("Tuple with all datatypes " , b )  print(len(b))# cal tuple length  c=("Hi",)  print("Tuple with one item " , c ) | ('hi', ' Hello ', 'Good Bye ')  <class 'tuple'>  Akshay  Tuple with all datatypes (1, True, 'Akshay', 6.765)  4  Tuple with one item ('Hi',) |

**LIST :**

* Lists are used to store multiple items in a single variable.
* list is changeable, meaning that we can change, add, and remove items in a list after it has been created
* List is similar to Tuple but it is changeable
* There can be list inside of list

Ex :

|  |  |
| --- | --- |
| a=["hi", " Hello " , "Good Bye "]  print(a)  print(type(a))  c=["Hi",]  print("List with one item " , c )  b=[1,True ,"Akshay", 6.765]  b[2]="Baalaji"  print("List with all datatypes " , b )  print(len(b))# cal tuple length  b[0:2]=[2,False]  print("Altered ",b)  b[0:2]=[]  print("Altered ",b)  d=[1,True,False , [1,"Akshay","Baalaji",False]]  print(d[3]) | ['hi', ' Hello ', 'Good Bye ']  <class 'list'>  List with one item ['Hi']  List with all datatypes [1, True, 'Baalaji', 6.765]  4  Altered [2, False, 'Baalaji', 6.765]  Altered ['Baalaji', 6.765]  [1, 'Akshay', 'Baalaji', False] |

**List operations :**

a=[1,"Akshay", True , False , "Baalaji " , 12]

b=[1,34,32,56,32,43,54,56,43,43]

1. Append – Add element at last

Ex : a.append(55)

1. Remove – This removes the specific term

Ex : A=[‘A’,’b’,’A’,’A’]

A.remove(‘A’) # This removes the first occurence of term

1. Pop – Delete the element based on index we specify

Ex : print("Pop " , a.pop(1))

1. Del – Del can also delete a specific index if we specify with index or else it delete the list

Ex: del a[0] / del a

1. clear – Clears the list content but the list remain same

Ex: a.clear()

1. Insert – Insert element based on index we specify a.insert(index,Element)

Ex: print("Insert" , a.insert(3, "Viji"))

1. Count – Return the number of times text appear in list

Ex: a.count(“Akshay”)

1. Reverse – reverse a list

Ex: a.reverse()

1. Sort – Sort a list

Ex : b.sort() and in reverse a.sort(reverse = True)

1. Extend – This help to append a list to another

Ex : a.extend(b)

1. + - Concatenate 2 List

Ex : c=a+b

1. Indexing – We can specifically call a element from list by indexing

Ex : a=[12,True,”akshay”,”Baalaji” , False,[12,324,34,324]]

a[5] [3] # 324

1. copy – Copy the List to another List

Ex : c=a.copy() or c=list(a)

1. Len – Calculate the length of String

Ex : print(len(a))

1. index – Give the index of entered the term

Ex : a.index(“Akshay”)

|  |
| --- |
| Prg:  listt=[1,True,"Akshaya Baalaji" , False , 45.3435, 'g']  print(listt)  #slicing and indexing  for index , elements in enumerate(listt):  print(f"Index {index} : {elements}");  """  enumerate :  Enumerate is Python built in function  return a tuple containg index and element in list  f - is the formateed string in latest pattern  """  listt[1:2] = [] #this removes the index 1  print(listt);  print();  #list functions  print("Length of list " , len(listt)); #Len return the lenght of list  print("Append 2 " , listt.append(2) );#append will add a element at last  print("List after function performed " , listt);  print("Remove 1 " , listt.remove(1)); #This removes the last element from the list  print("List after function performed " , listt);  print("Pop : " , listt.pop());#This deleted the last element from the list  print("List after function performed " , listt);  print("Insert " , listt.insert(1,True))  print("List after function performed " , listt);  print("Count \"Akshaya Baalaji\" " ,listt.count("Akshaya Baalaji"))  print("Reverse ",listt.reverse())  print("List after function performed " , listt);  print("Extend [Viji , True , 0 , False ] ",listt.extend(["Viji" , True , 0 , False ]))  print("List after function performed " , listt);  #Shared list  slist=listt  print("The slist is the shared list of list " ,slist)  #pop an element from listt to see it reflect on slist  listt.pop()  print("Original list after pop" , listt)  print("Shared list after pop " , slist)  #pop an element from slist to see it reflect on listt  slist.pop()  print("Original list after pop" , listt)  print("Shared list after pop " , slist)  print("Clear ", listt.clear()); #This clears all content but the list remains  print("List after function performed " , listt); |
| O/P:  [1, True, 'Akshaya Baalaji', False, 45.3435, 'g']  Index 0 : 1  Index 1 : True  Index 2 : Akshaya Baalaji  Index 3 : False  Index 4 : 45.3435  Index 5 : g  [1, 'Akshaya Baalaji', False, 45.3435, 'g']  Length of list 5  Append 2 None  List after function performed [1, 'Akshaya Baalaji', False, 45.3435, 'g', 2]  Remove 1 None  List after function performed ['Akshaya Baalaji', False, 45.3435, 'g', 2]  Pop : 2  List after function performed ['Akshaya Baalaji', False, 45.3435, 'g']  Insert None  List after function performed ['Akshaya Baalaji', True, False, 45.3435, 'g']  Count "Akshaya Baalaji" 1  Reverse None  List after function performed ['g', 45.3435, False, True, 'Akshaya Baalaji']  Extend [Viji , True , 0 , False ] None  List after function performed ['g', 45.3435, False, True, 'Akshaya Baalaji', 'Viji', True, 0, False]  The slist is the shared list of list ['g', 45.3435, False, True, 'Akshaya Baalaji', 'Viji', True, 0, False]  Original list after pop ['g', 45.3435, False, True, 'Akshaya Baalaji', 'Viji', True, 0]  Shared list after pop ['g', 45.3435, False, True, 'Akshaya Baalaji', 'Viji', True, 0]  Original list after pop ['g', 45.3435, False, True, 'Akshaya Baalaji', 'Viji', True]  Shared list after pop ['g', 45.3435, False, True, 'Akshaya Baalaji', 'Viji', True]  Clear None  List after function performed [] |

**List as Queue :**

Cmd:

from collections import deque

# Creating a deque

my\_deque = deque([1, 2, 3, 4, 5])

# Accessing elements in a deque

print(my\_deque[0]) # Output: 1

# Modifying elements in a deque

my\_deque[1] = 10

print(my\_deque) # Output: deque([1, 10, 3, 4, 5])

# Adding elements to a deque

my\_deque.append(6)

print(my\_deque) # Output: deque([1, 10, 3, 4, 5, 6])

# Adding elements to the left end of a deque

my\_deque.appendleft(0)

print(my\_deque) # Output: deque([0, 1, 10, 3, 4, 5, 6])

# Removing elements from a deque

my\_deque.popleft()

print(my\_deque) # Output: deque([1, 10, 3, 4, 5 ,6])

|  |  |
| --- | --- |
| a=[1,"Akshay", True , False , "Baalaji " , 12]  print("List operations starts ")  print("List At firest " , a )  print("Append " , a.append("HIBYE"))  print("List after append " , a)  print("Pop " , a.pop(1))  print("List after pop" , a )  print("Insert" , a.insert(3, "Viji"))  print("List after insert" , a )  b=[1,34,32,56,32,43,54,56,43,43]  print("Reversed List is " , a.reverse())  print("List after reverse" , a )  print("Integer list for sorting " , b )  b.sort()  print("Sorted array is " , b)  b.extend(["Akshay",False,True,"A"])  print("Extended B is ",b) | List operations starts  List At firest [1, 'Akshay', True, False, 'Baalaji ', 12]  Append None  List after append [1, 'Akshay', True, False, 'Baalaji ', 12, 'HIBYE']  Pop Akshay  List after pop [1, True, False, 'Baalaji ', 12, 'HIBYE']  Insert None  List after insert [1, True, False, 'Viji', 'Baalaji ', 12, 'HIBYE']  Reversed List is None  List after reverse ['HIBYE', 12, 'Baalaji ', 'Viji', False, True, 1]  Integer list for sorting [1, 34, 32, 56, 32, 43, 54, 56, 43, 43]  Sorted array is [1, 32, 32, 34, 43, 43, 43, 54, 56, 56]  Extended B is [1, 32, 32, 34, 43, 43, 43, 54, 56, 56, 'Akshay', False, True, 'A'] |

**Shared list :**

We can also create a share list in python

a=[1,3,2,3]

b=a

a.append(5)

print(a)#[1, 3, 2, 3, 5]

print(b)#[1, 3, 2, 3, 5]

**Sets :**

* When you print the set, it may not preserve the order you expect. Sets in Python are unordered collections, and the order of elements may vary across different executions of the program
* A set is a collection which is unordered, unchangeable\*, and unindexed.

Ex : thisset = {"apple", "banana", "cherry"}  
 print(thisset)

* unordered, unchangeable, and **do not allow duplicate values**.
* For length of String Ex : print(len(thisset))

Ex:

a=set("akshayabaalaji")

print(a) # {'a', 'j', 'l', 'h', 'i', 'b', 's', 'y', 'k'}

|  |
| --- |
| Prg :  # Define a set with a mix of different data types  sett = {1, True, "Akshaya Baalaji", False, 'a', 23.24, -234}  # Print the set, note that the order is unpredictable due to the nature of sets  print(sett)  # Convert the set to a list  listt = list(sett)  # Print the list  print("List is ", listt)  # Access the 4th element of the list (Python uses 0-based indexing)  print("The 4th element is ", listt[3])  # Slice the list from index 1 to 5 (end index is exclusive)  print("The elements from 1 to 6 are ", listt[1:6]) |
| O/P:  {False, 1, 'a', 'Akshaya Baalaji', -234, 23.24}  List is [False, 1, 'a', 'Akshaya Baalaji', -234, 23.24]  The 4th element is Akshaya Baalaji  The elements from 1 to 6 are [1, 'a', 'Akshaya Baalaji', -234, 23.24] |

**Set Operations :**

set1={2,3,4,5,6,True,False , 0 ,1 ,23 , 34 , 45 , 56 , 67 ,78 , 89 ,10 ,"Akshaya" , "Baalaji" }

print(set1)

#add : add an element in set

set1.add("Mukesh")

print("Set after adding element Mukesh : " , set1)

# remove a element from a set

set1.remove("Baalaji")

print("Set after removing the Baalaji elemennt : " , set1)

#union used to merge 2 set

set2={"A" , "K" , "s" , "H" , "a" , "Y" , "B",2,3,4}

print(set2)

set3=set1.union(set2)

print("The union of set1 andd set2 is set3 : " , set3)

#intersection common in set1 and set2 will be printed

set4=set1.intersection(set2)

print("This is the set after the intersection of set1 and set2 : ",set4)

print("The set1 after the intersection of the set2 " , set1)

#intresection update will update the set1 with the intersection

set5=set1.intersection\_update(set2)

print("This is the set after the intersection update of set1 and set2 : ",set5)

print("The set1 after the intersection\_update is " , set1)

#difference and difference update

set1={2,3,4,5,6,True,False , 0 ,1 ,23 , 34 , 45 , 56 , 67 ,78 , 89 ,10 ,"Akshaya" , "Baalaji" }

set2={"A" , "K" , "s" , "H" , "a" , "Y" , "B",2,3,4}

#difference Only the set without the element of set2 will be returned

print("Actual set is " , set1)

set3=set1.difference(set2)

print("The set differnce of set1 from set2 " , set3)

#difference update : Only the set without the element of set2 will be returned and updated to set1

set4=set1.difference\_update(set2)

print("The set difference update after set1 and set2 " , set4)

print("The set1 after difference update " , set1)

set1={2,3,4,5,6,True,False , 0 ,1 ,23 , 34 , 45 , 56 , 67 ,78 , 89 ,10 ,"Akshaya" , "Baalaji" }

set2={"A" , "K" , "s" , "H" , "a" , "Y" , "B",2,3,4}

#isdisjoint return whether same elements - false No common elements true

print("Ths isdisjoint of set1 and set2 " , set1.isdisjoint(set2) )

#issuperset : set1 contains all elements of set2

#issubset : set1 is completely contained within set2

print("the subset of set1 and set2 is ", set1.issubset(set2))

print("the superset of set1 and set2 is ", set1.issuperset(set2))

**Dictionary :**

* Dictionaries are used to store data values in key:value pairs.
* A dictionary is a collection which is ordered\*, changeable and do not allow duplicates
* Length of Dictionary Ex: print(len(thisdict))

Ex :

a={"akshay" : 1 , "baalaji" : 2 , "Chris" : 3 }

print(a) #{'akshay': 1, 'baalaji': 2, 'Chris': 3}

or

a = dict(name = "John", age = 36, country = "Norway")

print(a)

Methods:

1. clear() Removes all the elements from the dictionary
2. copy() Returns a copy of the dictionary
3. fromkeys() Returns a dictionary with the specified keys and value
4. get() Returns the value of the specified key
5. items() Returns a list containing a tuple for each key value pair
6. keys() Returns a list containing the dictionary's keys
7. pop() Removes the element with the specified key
8. popitem() Removes the last inserted key-value pair
9. setdefault() Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
10. update() Updates the dictionary with the specified key-value pairs
11. values() Returns a list of all the values in the dictionary

|  |
| --- |
| **Prg:**  dictt = {  "Name": "Akshaya Baalaji ",  "Age": 21,  "Phno ": 9361486996,  "Sex ": "Male",  "Address": "22 \"Gokulam nagar \" Second cross "  }  print(dictt)  #length of the dictionary  print("Length of the dict ", len(dictt))  #iterate  for keyy, value in dictt.items():  print("Key - ", keyy, end=" ")  print(" Value - ", value)  #Copy the dictionary  dictt2 = dictt.copy()  print("The copied dictionary is ", dictt2)  #Fromkeys function  keys = ["Dist", "State", "Pincode"]  values = ["TNJ", "TN", "612103"]  new\_dict = dictt.fromkeys(keys, values)  print("New dict created by from keys ", new\_dict)  #get values  print("The address is fetched by get function ", dictt.get('Address'))  #.items function  print("The dict print by dict.items() ", dictt.items())  #from keys make list  key\_list = dictt.keys()  print("The list from the keys of dict ", key\_list)  #from values make list  values\_list = dictt.values()  print("The list from the values of dict ", values\_list)  #pop with key deletes it  deleted\_ele = dictt2.pop("Address")  print("The deleted element is ", deleted\_ele)  print("The dictionary after pop ", dictt2)  #popitem deleted the last element  dictt2.popitem()  print("The dict after the popitem ", dictt2)  #update the value in dictt2  dictt2.update({'Phno': 9342083665})  print("The dict after the update ", dictt2)  #clear : deleted all data  dictt2.clear()  print("The dictt2 is cleared ", dictt2)  print("Only the data is cleared the dict remains ") |
| **O/P:**  {'Name': 'Akshaya Baalaji ', 'Age': 21, 'Phno ': 9361486996, 'Sex ': 'Male', 'Address': '22 "Gokulam nagar " Second cross '}  Length of the dict 5  Key - Name Value - Akshaya Baalaji  Key - Age Value - 21  Key - Phno Value - 9361486996  Key - Sex Value - Male  Key - Address Value - 22 "Gokulam nagar " Second cross  The copied dictionary is {'Name': 'Akshaya Baalaji ', 'Age': 21, 'Phno ': 9361486996, 'Sex ': 'Male', 'Address': '22 "Gokulam nagar " Second cross '}  New dict created by from keys {'Dist': ['TNJ', 'TN', '612103'], 'State': ['TNJ', 'TN', '612103'], 'Pincode': ['TNJ', 'TN', '612103']}  The address is fetched by get function 22 "Gokulam nagar " Second cross  The dict print by dict.items() dict\_items([('Name', 'Akshaya Baalaji '), ('Age', 21), ('Phno ', 9361486996), ('Sex ', 'Male'), ('Address', '22 "Gokulam nagar " Second cross ')])  The list from the keys of dict dict\_keys(['Name', 'Age', 'Phno ', 'Sex ', 'Address'])  The list from the values of dict dict\_values(['Akshaya Baalaji ', 21, 9361486996, 'Male', '22 "Gokulam nagar " Second cross '])  The deleted element is 22 "Gokulam nagar " Second cross  The dictionary after pop {'Name': 'Akshaya Baalaji ', 'Age': 21, 'Phno ': 9361486996, 'Sex ': 'Male'}  The dict after the popitem {'Name': 'Akshaya Baalaji ', 'Age': 21, 'Phno ': 9361486996}  The dict after the update {'Name': 'Akshaya Baalaji ', 'Age': 21, 'Phno ': 9361486996, 'Phno': 9342083665}  The dictt2 is cleared {} |

**There are four collection data types in the Python programming language:**

* **List** is a collection which is ordered and changeable. Allows duplicate members.
* **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
* **Set** is a collection which is unordered, unchangeable\*, and unindexed. No duplicate members.
* **Dictionary** is a collection which is ordered\*\* and changeable. No duplicate members.

**Built-in Functions :**

A=[9,3,6,1,4,8,6,9,9,6]

Min min(a)

Max max(a)

Bin bin(a) – binary conversion

Hex hex(a) – Hexadecimal conversion

Sort a.sort() – This sorts the entire var

Sorted sorted(a) – This only sorts and not store it

|  |  |
| --- | --- |
| a=[12,23,34,34,34,54,54,4335,532,45]  a.sort()  print(a)  b=[9,3,6,1,4,8,6,8,6]  print(sorted(b))  print(b) | [12, 23, 34, 34, 34, 45, 54, 54, 532, 4335]  [1, 3, 4, 6, 6, 6, 8, 8, 9]  [9, 3, 6, 1, 4, 8, 6, 8, 6] |

Slice – Here the slice value is assigned to obj

Ex: str="Pyhon is the best languagge and it is easy to learn "

a=slice(0,15,2)

b=slice(4)

print(str[b]) #Pyho

print(str[a]) #Phni h e

type – This gives the data types of variable

Ex : type(a) #<class 'slice'>

**Some other functions with code**

|  |
| --- |
| **Prg :**  1. split in string  Ex :  string="i am Akshaya Baalaji i am from thanjavur "    list\_of\_str=list(string)  print(list\_of\_str)  string\_spli=list(string.split(" "));  print(string\_spli)  string\_spli=list(string.split(" " ,2));# 2 is the index till we need to split  #on above the split will be done for the first 2 index and make the remaining content in 3third index    2. capitalize : make the first letter as capital  Ex:  print(string.capitalize())    3. upper , lower #make all charecter ass upper / lower  Ex :  print(string.upper())  print(string.lower())  4. title #make all start charecter as upper case  Ex :  print(string.title())  5.format (important )  Ex:  string\_new="I am {3} and I am from {2} and I am specialized in {1} and my age is {0}"  print(string\_new.format(21,"Bigdata","TNJ","Akshaya Baalaji"))    6. replace  Ex :  print(string.replace("Akshaya", "AAkshaya"));  #the above function replace all the occurence of the string  print(string.replace("A", "X",2));  #the above function represent the number of times we need to apply the replace on the string    7.count #count the number od occurence of the find  Ex :  print(string.count("A")); # do a full search on the string to find the number of occurences  print(string.count("A",0,9));  #the above count we are mentioning the index where to start and end the count of occurence  8.strip and rstrip  Ex :  print("Strip....................")  print(" a b c d ".strip()) # remove leading and trailing white spaces  print(" a b c d ".rstrip()) # remove the spaces at the end  9.\_ \_lt\_ \_ # return boolean and it is less than  Ex :  a=10  print(a.\_\_lt\_\_(11)) #true    10. We can also import funcitons from the modules  Ex :  import math # module from which we are going to use function  floatvar=100.34  print(floatvar.\_\_floor\_\_())  #usually the floor cant be applied on the float and will only be applied on the int  #so we are using the float funciton from the math module for solving it |

**Type casting –**

* Changing the datatype of the variable
* The originality of the variable wont be changed in type casting
* Functions to convert the variable type int , float , str , list , tuple , dict , set

|  |
| --- |
| Prg :  a=1;  aa=str(a);  string1="Hi BYe "  print(string1+aa) # there will be no error  “””  number=1000  string\_num = str(number)  float\_num=float(number)  #char\_number=char(number)  list\_number=list(string\_num)  tuple\_number=tuple(string\_num)  set\_number = set(string\_num)  #dictionary\_number=dict(number ) // need key value for this  print(string\_num)  print(float\_num)  print(list\_number)  print(tuple\_number)  print(set\_number)  #this will be string by default  number=input("Enter a String or NUmber ")  print(type(number))  #so type cast  number1=int(input("Enter the number because it is type casted at time of input itself "))  print(type(number1))  “”” |
| O/P:  Hi BYe 1  1000  1000.0  ['1', '0', '0', '0']  ('1', '0', '0', '0')  {'1', '0'}  Enter a String or NUmber 100  <class 'str'>  Enter the number because it is type casted at time of input itself 123  <class 'int'> |

**Constrains:**

**if constrains :** Checks the condition whether the stmt is true or not if true enter into the loop

**isinstance() :** It is a function that check whether the datatype of the variable is correct or not if same return true and if different enter false

**Example :**

a=1;

aa=str(a);

if isinstance (aa,str):

print("It is a string datype")

**Features of using Function / Methods :**

1. concurrency - calling the same function at same time from different programs / environment .
2. Parallelism / Multi threading : The function called at same time by different thread and act based on the different input from the used on different thread
3. Distributed : The function run in parell and provide the desired output as per the input passed across multiple work area
4. Functions are composible :

Ex :

```python

a="Akshaya Baalaji"

b= a.lower().index('b') # passing the output of the lower to index function

print(a)

print(b)

```

1. Function are time bound :

Ex :

#the below code is time oriented

a=1

if a>=1 :

a=a+1

print(a)

# when we put the above code in a function it become non time bounded

a=1

def function ():

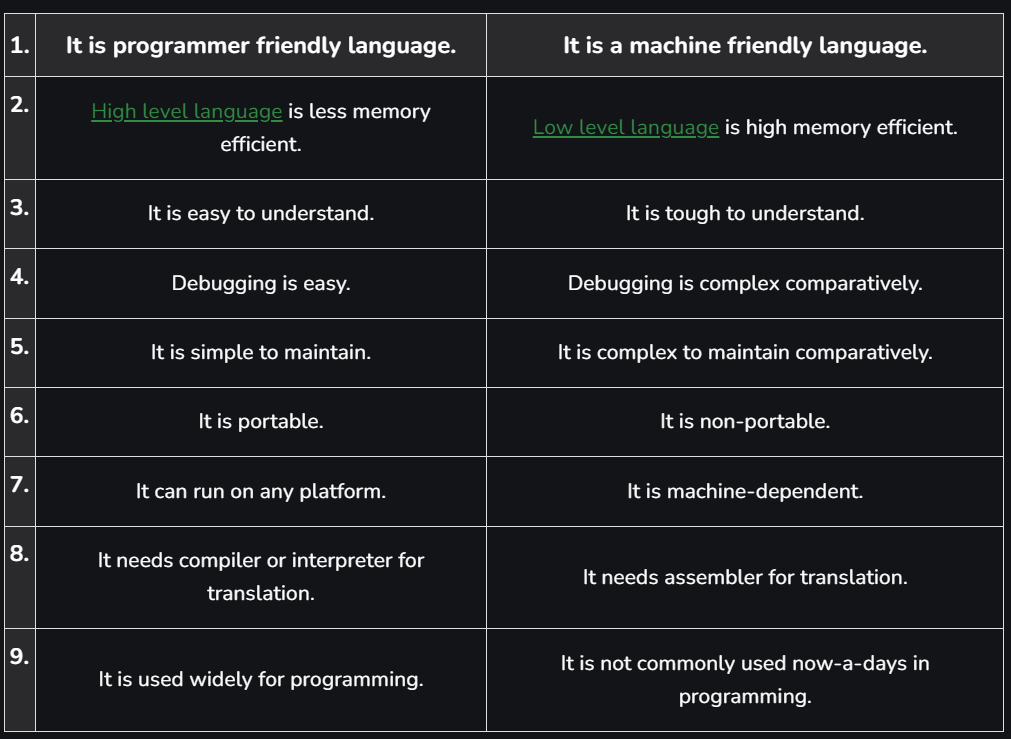
global a

if a>=1 :

a=a+1

print(a)

function()



**Function based programming :**

* Function/Method is a group of statement to perform certain task
* If we are giving the high priority for a function and use that function again and again rather than writing the function again and again this mainly reduces the complexity .
* Function based programmin help us convert the programmin language to high level language (10 line of code for calling n number of times) rather than the low level code (typing it for the n number of times (boiler plate coding ))
* Grouping the code to perform a specific task for reuse
* Parameter - during a function declaration , definition
* Argument - during a function call

**Function in Python :**

* Simple code to do a task
* Start with lambda - function
* Start with def - method
  + Name must be in lower case
  + followed by ( )
  + and a colon :
* A function is a block of code which only runs when it is called.
* You can pass data, known as parameters, into a function.
* A function can return data as a result.
* Functions are mainly used for the reusability.
* Function can be understood by hovering the function while creation

**Ex :1**

x=10 # **global variable** used outside of functions

y=20

def add(a,b): #parametrer

c=a+b # **local variable** used within functions

print(c)#30

add(x,y);#argument

def mul(a,b):

return a\*b

vari=mul(x,y)

print(vari)#200

**Ex: 2**

money=1000

def add():

global money # here we are referring to global variable

money=money+1

print(money)

add() # here we are calling function then global variable is incremented

print(money)

**Ex 3 :**

x=100

def fun():

x=20

print("Inside",x); #Inside 20

def fun1():

global x

x=40

print("Inside",x) #Inside 40

print("outside" , x) #outside 100

fun()

print("outside 1" , x) #outside 1 100

fun1()

print("outside 2" , x)#outside 2 40

**Ex 4:**

def add (a,b)://a and b are arguments

return a+b;// we are ending the function by the return

|  |
| --- |
| Prg:  i=10  #function for looping  def function(n):  for i in range (0,n,1) :  print(i , end=" ")  function(i)  #function for binary conversion :  binary=0;  hexadecimal=0;  decimal=12312  def dec\_binary\_func(decimal):  global binary  binary=bin(decimal)  dec\_binary\_func(decimal)  print()  print("Decimal converted to binary :" , binary)  #hexadecimal conversion  def dec\_hexa\_func(decimal):  global hexadecimal  hexadecimal=hex(decimal)  dec\_hexa\_func(decimal)  print("Decimal conversion to Hexadecimal ", hexadecimal)  #Function for list operation :  listt1=[["Akshaya" , "Baalaji "] , "Senthilraj " , 22 , "Gokulam nagar" , 612103, True]  print("List starts ")  def list\_function(list1):  print("The original list is " , list1)  print("Index the first element " , list1[0])  #print("The max of the list is " , max(list1)) compare on string with int is not possible  #print("The min of the list is " , min(list1))  #exterd  list1.extend(["Maarimuthu" , "Govindhaswamy"])  print("In function the list is " , list1)  #extend on the list in function make the original list affected  #pop  list1.pop()  print("The list after pop is " , list1)  list1.reverse()  print("The list after reverse" , list1)    list\_function(listt1)  print("After the functions " ,listt1)  list2=["Akshay","Chris","Baalaji"] # adding annumber here makes the list min , max error  print(min(list2))  print(max(list2)) |

|  |
| --- |
| Ex 2 :  def add(a,b):  print(f"The addition of {a} and {b} is : " , a+b)  def sub(a,b):  print(f"The substraction of {a} and {b} is : " , a-b)  def mul(a,b):  print(f"The multiplication of {a} and {b} is : " , a\*b)  def div(a,b):  print(f"The division of {a} and {b} is : " , a/b)  def calculator(a,b):  print("1 - Addition \n2 - Substraction \n3 - Multiplicaton \n4 - Division")  n = int(input("Enter your choice "))  if n==1 :  add(a,b)  elif n==2 :  sub(a,b)  elif n==3 :  mul(a,b)  elif n==4 :  div(a,b)  else :  print("Enter the choice perfectly ")  func\_call\_from\_different\_file.py  from function\_of\_function import calculator  a=int(input("Enter number 1 "))  b=int(input("Enter number 2 "))  calculator(a,b) |

|  |
| --- |
| """  Swiggy usecase using function :  swiggy\_mart = {"max\_discount":500, "min\_purchase" : 5000 , "discount\_percentage" : 10}  def calculation\_function(swiggy\_mart,cost:int):  variable = int(swiggy\_mart["min\_purchase"])  if cost == variable :  print(f"The cost you have purchased is {cost} and the dicount is {swiggy\_mart["discount\_percentage"]} and amount reduced is {swiggy\_mart["max\_discount"] } ")  variable = int(swiggy\_mart["max\_discount"])  payable = cost - variable  print(f"The amount you have to pay is {payable} ")  calculation\_function(swiggy\_mart , 5000)  """ |

**Argument types available in Python :**

We can call a method with different argument types

1. Positional
2. Default
3. Named
4. Arbitary
5. Arbitary and Keyword

**Programs for understanding :**

|  |
| --- |
| """  Simple mail id generation function in arbitary argument types  """  #default is the function without having any arguments passed in it  a="baalaji12"  b="akshay"  def normal ():  global a  global b  print(a + "." + b + "@gmail.com")  normal()  #positional argument and Named argument    def positional(fname , lname ):  print(fname + '.' + lname + '@gmail.com')  positional("baalaji" , "12akshay") #Here positional argument is used  positional(lname="12akshay",fname="baalaji") # here the named argument is used  #the below can generate some error when  # arbitary\_argument("s","baalaji","akshay")  # in a unexpected argument this creates error because we need baalajiakshays@gmail.com  # but we will get sbaalajiakshay@gmail.com  # \*a is the tuple that hold the argument  def arbitary\_argument(\*a):  print(type(a))  print(a[0] + '.' + a[1] + '.' + a[2] + '@gmail.com')  arbitary\_argument("baalaji","12" , "akshay")  #\*\*a is the dictionary that hold the key and value of argument  def keyword\_argumet(\*\*a):  print(type(a))  print(a["fname"] + "." + a["lname"] + "." + a["father\_name"] + "@gmail.com")  keyword\_argumet(father\_name="s" , fname="baalaji" , lname="akshay") |

**Function Based programming :**

**Special Methods** : Anonymous funciton / Lambda function , Higher Order Function , Recursive Function , Hierarchical function , Closures funciton

**Object based Programming**

**Special Methods** : Instance , class , static methods

**Higher order function :**

* A method taking another method as input (input argument )
* Taking Function as an argument this make the code reusability
* There are 2 types of Higher Order Function

1. Type 1
2. Type 2

**Higher Order Function Type 1 :** if a method takes another argument as a input it is a HOF type 1

|  |
| --- |
| #Example 1  print("Complex programming / Higher order Method")  a=5  b=6  def add():  global a  global b  return a+b  def complex\_funciton(value , a): # it is a Higher Order method of type 1  print("The diff between a and b is " , value()-a)  complex\_funciton(add , 5)  #Example 2  print("Higher Order Funciton Type 1 ")  def mul(a,b):  print(f"The multiplication of {a} and {b} is : " , a\*b)  def div(a,b):  print(f"The division of {a} and {b} is : " , a/b)  def calculator(a,b , variable):  if variable =='add' :  def add(a,b):  print(f"The addition of {a} and {b} is : " , a+b)  return add(a,b)  elif variable =='sub' :  def sub(a,b):  return a-b  return sub(a,b)  elif variable =='mul' :  def mul(a,b):  print(f"The multiplication of {a} and {b} is : " , a\*b)  return mul(a,b);  elif variable =='div' :  def div(a,b):  print(f"The division of {a} and {b} is : " , a/b)  return div(a,b)  else :  print("Enter the choice perfectly ")  calculator(12,23 , "div")  output= calculator(12,23,"sub") # here I have tried return stmt  print(f"The substraction of 12 and 23 is : " , output) |

**Higher order Function Type 2** : If a method return another method as a return type the it is HOF type 2 this is the combination of Hierarchical + Nested / Partial / currying function

|  |
| --- |
| **Prg 1 :**  print("Higher Order Funciton Type 2")  def calculator(a,b , variable):  if variable =='add' :  def add(a,b):  print(f"The addition of {a} and {b} is : " , a+b)  return add(a,b)  elif variable =='sub' :  def sub(a,b):  print(f"The substraction of {a} and {b} is : " , a-b)  return sub(a,b)  elif variable =='mul' :  def mul(a,b):  print(f"The multiplication of {a} and {b} is : " , a\*b)  return mul(a,b);  elif variable =='div' :  def div(a,b):  print(f"The division of {a} and {b} is : " , a/b)  return div(a,b)  else :  print("Enter the choice perfectly ")  calculator(12,23 , "div") |

|  |
| --- |
| **Prg 2 : AC remote use-case**  on\_off = input("Enter ON to turn on the AC and OFF to turn off the AC ")  temparature = 15  def ac\_remote (args ):  if args == "ON":  def on\_function (on\_off):  if (on\_off == 'ON'):  temparature = int(input("Enter the temparature of the AC to be Set "))  wing = input("Enter the wing mode ")  return temparature,wing  print(on\_function(args))  else :  def off\_function (on\_off):  if (on\_off == 'OFF'):  return "Turn on Fan and sleep "  print(off\_function(args))  ac\_remote(on\_off)  print("The temparature outside the function is : ",temparature) |

|  |
| --- |
| **Prg 3:**  def add\_all(a,b,c):  return a+b+c  def HOF\_type2 (a , variable):  b =10  def function\_to\_return ():  c =10  return variable(a,b,c)  return function\_to\_return()  a =10  output =HOF\_type2(a,add\_all)  print(output) |

**Recursion :**

Function calling itself is recursion

**Ex :**

|  |  |  |
| --- | --- | --- |
| a=5  fact=1  def facto(x):  if x > 1 :  y=x  global fact  fact=fact\*y  facto(y-1)  return fact  facto(a)  print(fact) | a=int(input("Enter the input "))  fact=1  def facto(x):  if x < 1 :  return 1  else:  return(x \* facto(x-1))  fact =facto(a)  print(fact) | aa=10  a=0  b=1  lim=0  print ("0 1" , end= " ")  def fib(a,b):  global aa , lim  if a >= 0 and b >= 0 and aa-2>lim:  c=a+b  print(c , end =" ")  a=b  b=c  lim=lim+1  fib(a,b)  fib(a,b) |
| Factorial :  # normal factorial  def factorial\_normal (arg):  fact =1  while arg > 0 :  fact = arg \* fact  arg =arg - 1  return fact  inputt = int ( input ("Enter a input for finding the factorial " ))  output = factorial\_normal(inputt)  print("The output is " , output)  #recursive factorial function  fact =1  def factorial\_recursive (arg):  global fact  if arg == 0 :  return fact  fact = arg \* fact  arg = arg -1  return factorial\_recursive(arg)  inputt = int ( input ("Enter a input for finding the factorial " ))  output = factorial\_recursive(inputt)  print("The output is " ,output)  print("The output is " , fact ) | | | |

|  |
| --- |
| Fibonocci series :  Recursive frunction - Fibonocci series  #normal fibonocci  def fibonocci(rangee):  a = 0  b = 1  print(a , b , end = " " )  for i in range (0,rangee - 2,1):  c= a+b  print(c , end = " ")  a=b  b=c  rangee=int(input("Enter the range for fibonocci : "))  fibonocci(rangee)  # recursive fibonocci  a = 0  b = 1  print(a ,b , end = " ")  def fibonocci\_recursive(a , b ,rangee ) :  if rangee >= 3:  c= a+b  print(c , end = " ")  a=b  b=c  rangee = rangee -1  fibonocci\_recursive(a,b,rangee) # made to back up the funciton call on the call stack  fibonocci\_recursive(a,b,10)  print() # for new line  print(a ,b , end = " ")  what\_it\_return = fibonocci\_recursive(a,b,10)  print() # for new line  print(what\_it\_return) |

**Closure function :**

If the variable outside the function affect the variable inside the function is known as closure function .

|  |
| --- |
| set\_value =23345  def funciton():  set\_value =23  print(" value Inside the funciton " , set\_value )    funciton()  print("The value outside the funciton is " , set\_value) |

**Enumerate :**

enumerate() function takes a collection (e.g. a tuple) and returns it as an enumerate object.

Syntax enumerate(iterable, start)

Ex :

a=["Akshay","baalaji","Chris"]

b=list(enumerate(a))

print(b) # [(0, 'Akshay'), (1, 'baalaji'), (2, 'Chris')]

c=list(enumerate(a,start = 1))

print(c); # [(1, 'Akshay'), (2, 'baalaji'), (3, 'Chris')]

|  |
| --- |
| **Prg:**  list1=["Akshay","Baalaji","Chris","Deepak","Elavarasan"]  list1.reverse()  list2=list(enumerate(list1,start=1))  print("This is the original list ",list1)  print("This is the enumerated list ",list2)  list2.sort() #this will sort based on index  print(list2) |
| **O/P:**  This is the original list ['Elavarasan', 'Deepak', 'Chris', 'Baalaji', 'Akshay']  This is the enumerated list [(1, 'Elavarasan'), (2, 'Deepak'), (3, 'Chris'), (4, 'Baalaji'), (5, 'Akshay')]  [(1, 'Elavarasan'), (2, 'Deepak'), (3, 'Chris'), (4, 'Baalaji'), (5, 'Akshay')] |

**\*\*kgwars :** (least bother) (keyword argument )

If you do not know how many keyword arguments that will be passed into your function, add two asterisk: \*\* before the parameter name in the function definition

**Ex**

def name(\*\*name):

print("Hi " , name["firstn"] ,"Bye " , name["lastn"])

#Hi Akshay Bye baalaji

name(firstn="Akshay" , lastn="baalaji")

**Lambda Function or Operator:**

* Start with lambda - function
* We can use a variable to use this anonymous function then and there without importing the code from other file
* Logic can’t be in multiple lines
* lambda function is a small anonymous function.
* A lambda function can take any number of arguments, but can only have one expression
* Syntax : lambda argument : expression
* Expression are faster than functions

**Simple Lambda function :**

|  |
| --- |
| a = 10  b = c = 10  output\_with\_lambda = lambda a,b,c : a+b+c  print(output\_with\_lambda(a,b,c)) |

**Difference between Lambda and Function :**

|  |
| --- |
| **Prg 1 :**  a=lambda b : b\*b  print("Printed by using lambda ", a(5)) # Printed by using lambda 25  #which is similar to below  a=5  def mull(a):  print("printed by using Function " , a \* a) #printed by using Function  **Prg 2 :**  #some ex of lambda  b=lambda c,d,e : c\*d+e  c=5  d=5  e=5  print("Printed by lambda " , b (c,d,e))#Printed by lambda 30  # if done by functions  def cal(c,d,e):  print("Printed by function " , c\*d+e)#Printed by function 30  cal(c,d,e)  **Prg 3 :**  """  Difference between lambda funciton and method  In the below code the complex logic cant be acheived by the lambda  def function(a,b,variable ):  if variable == 'min':  if a<b :  return a  else :  return b  else :  if a>b :  return a  else :  return b  print("The min is " , function(10,20,"min"))  #cant do in lambda and it is very comples to make the above logic in one line  a=100  b=20  lambda\_funciton = lambda a,b : a>b  min = lambda\_funciton(a,b)  if min is False :  print("The min value is " , a)  else :  print("The maximum value is " , a)  """ |

|  |
| --- |
| **Prg:**  #lambda operation (a way to create a small function)  #Use lambda when you need a function for a shorter period of time  """  i=10  result=lambda i : i\*i  print("The result is " , result) # this prints the lambda address to use the function do below  """  #which is similar to  #def function(i):  # return i\*i;  #var=function(i)  """  print("Square is " , result(i))  calc = lambda i: "Even number" if i % 2 == 0 else "Odd number"  print("Even or Odd by lambda ", calc(i))  addition=lambda a,b : a+b  print("Addition by lambda" , addition(20,40))  #Lambda on list  list1=["Akshaya" ,"Baalaji" , "Chris"]  list2=lambda list1: list1  print(list2(list1))  #squaring using lambda  list3=[1,2,3,4,5,6,7]  list4=lambda list3 : list3\*list3  print("The squared list is " , list(list4(list3)))  """  """  Original list of integers:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  Even numbers from the said list:  [2, 4, 6, 8, 10]  Odd numbers from the said list:  [1, 3, 5, 7, 9]  list\_org =[1,2,3,4,5,6,7,8,9,10]  lambdaa\_filter=list(filter(lambda list1 : list1%2==0 , list\_org ))  lambdaa\_map=list(map(lambda list1 : list1%2!=0 , list\_org ))  print(lambdaa\_filter)  print(lambdaa\_map)  """  #isdigit function  """  strings ="Akshay"  string\_int="1"  print(strings.isdigit())  print(string\_int.isdigit())  lambda\_work=lambda var : var.isdigit()  print("Using the lambda ")  print(lambda\_work(string\_int))  print(lambda\_work(strings))  print("NOw for the list ")  list\_of\_combo=["1" , "a" , "Akshay" , "Baalaji" ,"1.4563", "C"]  print(list\_of\_combo)  lambda\_work\_detailed=list(map(lambda list1 : list1.isdigit() , list\_of\_combo))  print(lambda\_work\_detailed)  print("Same as above without lambda ")  def function (list1):  if list1.isdigit()==1 :  return True;  else:  return False  without\_lambda=list(map(function , list\_of\_combo))  print(without\_lambda)  """  """  [1, 2, 3, 5, 7, 8, 9, 10]  Number of even numbers in the above array: 3  Number of odd numbers in the above array: 5  list\_of\_num=[1,2,3,4,5,6,7,8,9,10,11]  lambda\_even=list(map(lambda list1: list1%2==0 , list\_of\_num))  lambda\_odd=list(map(lambda list1: list1%2!=0 , list\_of\_num))  print(lambda\_even)  print(lambda\_odd)  result=0  def function(list1):  global result  for i in range (0,len(list1),1):  if(list1[i] == True):  result=result+1  function(lambda\_even)  print("Number of even ",result)  result=0  function(lambda\_odd)  print("Number of odd ",result)  """  list\_1=[1,2,3]  list\_2=[4,5,6]  lambda\_sum=list(map(lambda x,y : x+y , list\_1 , list\_2 ))  print(lambda\_sum) |

**Map functions :**

* map() function executes a specified function for each item in an iterable. The item is sent to the function as a parameter
* Syntax : map(function, iterables)

**Ex :**

a=["Apple","Oreange" , "Malgouva" , "narthangai "]

def funlen(b):

return len(b)

b=map(funlen , a )

print(list(b)) #[5, 7, 8, 11]

#using lambda

g=map(lambda x : len(x) , a)

print(list(g))

# another ex

b=["Akshay" , "Baalaji" , "Chris" , "Deepak "]

c=["-s" ,"-d", "-e" ,"-r"]

def joinn(d,e):

return d+e

e=map(joinn , b,c)

print(list(e)) #['Akshay-s', 'Baalaji-d', 'Chris-e', 'Deepak -r']

#using lambda

d=map(lambda x,y : x+y , b,c )

print(list(d))

Ex 2:

a=[1,4,29,4,23,3]

b=list((map(lambda x : x+100 , a)))

print(b) #[101, 104, 129, 104, 123, 103]

#or else

def funadd(d):

return d+100

c=list(map(funadd , a))

print(c)#[101, 104, 129, 104, 123, 103]

|  |
| --- |
| Prg:  # mpp function has a function as a parameter and execute function in iteratable form  import itertools  """  list1=["Akshay", "Baalaji"]  list2=["-s","-s"]  def function(list1, list2):  return list1+list2  list3=map(function, list1,list2)  print("Map function : ",list3)#adderss  print("Map Working : " , list(list3))  """  #exercise  """  1. Write a Python program to triple all numbers in a given list of integers. Use Python map.  list\_int=[1,2,3,4,5,6,7,8,9,10]  def function\_loop (list1):  return list1\*list1\*list1  print("The original list " , list\_int)  map\_triples=map(function\_loop , list\_int )  print("This is the list of triples " , list(map\_triples))  """  """  2. Write a Python program to add three given lists using Python map and lambda.  list1=[1,2,3,4,5]  list2=[2,3,4,5,6]  list3=[3,4,5,6,7]  list4=[4,5,6,7,8,9,10]  def function\_list\_add (list1,list2,list3,list4):  return list1+list2+list3+list4  map\_func=map(function\_list\_add , list1,list2 ,list3, list4)  print(list(map\_func))  """  """  3. Write a Python program to listify the list of given strings individually using Python map.  list1=["Akshaya" ,"Baalaji" , "Chris"]  def function\_print(list21):  return list21  list2=list(map(function\_print , list1))  print("List copied which is similar to list1.copy() : ", list2)  #list3=list2.copy()  #print(list3)  """  """  5. Write a Python program to square the elements of a list using the map() function.  list1=[1,2,3,4,5,6]  def sq\_func(list1):  return list1\*list1  list2=list(map(sq\_func , list1))  print("Squared list is " ,list2)  #list3=sq\_func(list1) cant be done  #print(list3) when we use map we can iterate using the map function  """  """  6. Write a Python program to convert all the characters into uppercase and lowercase and eliminate duplicate letters from a given sequence. Use the map() function.  list1=['a' ,'b','c','d','e']  list3=["Akshay" , "Baalaji" "Chris" , "Deepak" , "Elavarasan"]  def function\_convert(lista):  return lista.upper()  list2=list(map(function\_convert , list1))  list4=list(map(function\_convert , list3))  print(list2)  print(list4)  """  """  7. Write a Python program to add two given lists and find the difference between them. Use the map() function.  list1=[1,2,3,4,5,6,7,8,9,10]  list2=[10,9,8,7,6,5,4,3,2,1]  def function\_add(list1, list2):  return list1+list2  list3=list(map(function\_add,list1, list2))  print("The added list " , list3)  def function\_sub(list1,list2):  return list1-list2  list4=list(map(function\_sub, list1, list2))  print("The deifference of the list is " , list4)  """  """  8. Write a Python program to convert a given list of integers and a tuple of integers in a list of strings.  convert the list and tuple to list of string  list1=[1,2,3,4,5]  tuple1=(1,2,3,4,5)  string1=list(map(str, list1))  string2=list(map(str, tuple1))  print("List converted to string " , string1)  print("Tuple converted to string " , string2)  """  """  9. Write a Python program to create a new list taking specific elements from a tuple and convert a string value to an integer.  """  """  10. Write a Python program to compute the square of the first N Fibonacci numbers, using the map function and generate a list of the numbers.  n=50  a=0  b=1  c=0  def function\_fib(i):  global a,b,c  for i in range (0,i,1):  c=a+b  a=b  b=c  print(c , end=" ")  #list1=list(itertools.islice(function\_fib(n)))  function\_fib(n)  # I dont know how to do with map  """  """  11. Write a Python program to compute the sum of elements of an array of integers. Use the map() function.  list1=[1,2,3,4,5,6,7,8,9 , 1]  list2=list(map(lambda x: x+1 , list1)) #add one to every element  print(list2)  sum=0  def function\_sum\_of\_ele(lista):  global sum  for i in lista :  sum =sum+i  return sum  sum\_of\_ele=function\_sum\_of\_ele(list1)  print("Sum of all element is " , sum\_of\_ele)  """ |

**Additional :**

**str.isdigit() method:**

This method checks whether all characters in the string are digits. It will return True if the string consists only of digits, and False otherwise.

Copy code

def is\_integer(s):

return s.isdigit()

# Example usage:

print(is\_integer("123")) # True

print(is\_integer("123abc")) # False

**Filter functions:**

* The filter() function returns an iterator where the items are filtered through a function to test if the item is accepted or not.

Ex:

a=[1,2,3,4,5,6,7,8,9,11,12,13,1,4,43,53,23]

b=list(filter(lambda x: (x%2 == 0 ), a))

print(b)

print(sorted(b))

#or else by function

def myfunc(r):

if ( r % 2 == 0 ):

return r

c=list(filter(myfunc ,a))

|  |
| --- |
| **Prg:**  #filter function filters the record based on the condition  """  #this is the list filtered only the odd element  list1=[1,2,3,4,5,6,7,8,9,10]  list2= list(filter(lambda s : (s%2==0) , list1 ))  print("filtered List is ",list2)  #understanding the string  string1="Akshayabaalaji"  list\_string=string1.split(" "); #searches for space  list\_string\_conversion=list(string1)#directly convert to LIST fROm string  print(list\_string\_conversion)  print("The string converted to list\_string is " , list\_string)  def func\_vowel(list1):  if list1 in ['a' , 'e' , 'i' , 'o' , 'u'] :  return True  else :  return False  list3=list(map(func\_vowel, list\_string\_conversion)) #map is used for mapping to list  list4=list(filter(lambda list : list in ['a' , 'e' , 'i' , 'o' , 'u'] , list\_string\_conversion))  print(list3)  print(list4)  #Write a Python program that creates a  #list of names and uses the filter function to extract names that start with a vowel (A, E, I, O, U).  """  """  10. Write a Python program that implements a Python program  that filters out dates (in the format "YYYY-MM-DD") that are in the future using the filter function.  date="2024-12-12"  print("String is : " , date)  splitt=date.split("-")  print(splitt)  filter\_prg=list(filter(lambda s: s.split("-") , date ))  print(filter\_prg)  """ |

**Reduce Function :**

* This helps in reduce the list to last one item
* We have import a package to use this **from functools import reduce**

Ex:

from functools import reduce

a=[1,2,3,4,5,6,7,8,9,11,12,13,1,4,43,53,23]

b=reduce(lambda x,y: x+y, a)

print(b)

**Exceptions (error) and Exception Handler :**

**try**

* try is a place where we can expect exception happen in the code
* try must require either except , finally

**except**

* is used for the handling the exception
* What has to be done for the exception or error occurred

**else**

* it is a place if everything goes right the code will be redirected to else part after the try block
* if except occur else wont run

**finally**

* is a place at any cost it will be running even error occur or not

**Exception handling :**

* There are many errors may arise in python

1. Indentation error
2. Syntax error (compilime timer error)
3. Runtime error (known as Exceptions )

* Error which cant be avoided are handled by exceptions . The BaseException is where all exceptions arise it is the parent of hierarchy
* try : Block in which we expect exception
* If there are no exceptions arises the try only arises and we can also specify a specific else part to denote the stmt in else has no exceptions
* **Exception error :** It is the common exception handler it can handle any type of error

Ex:



Ex:

|  |  |
| --- | --- |
| while True :  try:  x=int(input("Enter a number "))  break  except ValueError :  print("You need to enter numbers only ") | Enter a number e  You need to enter numbers only  Enter a number te  You need to enter numbers only  Enter a number tr  You need to enter numbers only  Enter a number 7 |

Ex 1:

|  |  |
| --- | --- |
| while True :  try:  #x=10\*(1/0)  # y=4+3\*spam  z='2'+4  break  except ZeroDivisionError :  print("You are trying to devide by 0  break  except NameError :  print("You are using the un defined variable " , )  break  except TypeError:  print("you had entered something wrong ")  break  else:  print(x,y,z)  break  try :  a='2'+2  except Exception as e :  print("Error arises here ") | #1 You are trying to divide by 0  #2  You are using the un defined variable  #3  you had entered something wrong  #Here in else part there are no errors which can be even specified in try for clarity we specify in else part  #error arises here |

**Practical :**

#Zero exception error

try :

a=99/0

except Exception as e :

print("The error arraises is : " , e )

#Type error

try:

x = "hello" + 5 # This will raise a TypeError

except TypeError as e :

print("The error arraises is : " , e )

#index error

try:

fruits = ["apple", "banana", "cherry"]

index = 5

print("Fruit retrieved:", fruits[index])

except IndexError as e :

print("The error arraises is : " , e )

#HAndling the unexpected error using the exception

try:

prunt

except Exception as e:

print("An unexpected error occurred:", e)

#Keyerror

try:

my\_dict = {'a': 1, 'b': 2}

value = my\_dict['c'] # This will raise a KeyError

except KeyError as e :

print("The error arraises is : " , e )

#Attribute error

try:

x = 5

x.append(10) # This will raise an AttributeError

except AttributeError as e :

print("The error arraises is : " , e);

#keyboard interrupt error

try:

while True:

pass

except KeyboardInterrupt as e :

print("The error arraises is : " , e )

#custom exceptions #class named Kolaru, which inherits from the built-in Exception class.

class Kolaru(Exception):

def \_\_init\_\_(self, message="Some random exception occurred"):

self.message = message

super().\_\_init\_\_(self.message)

try:

option = int(input("Enter a number: "))

if option > 56:

raise Kolaru("Option cannot be greater than 56")

except Kolaru as e:

print("An unexpected error occurred:", e)

**Python Scripting Modules and Packages :**

* A module is a python object with arbitrarily named attribute that you can bind and refer . Simply a file containing a python code
* Library are created by concept called Package

Ex

One.py , Two.py, Three,py are some of the modules and we apecify a particular module and import the rest in that

Main.py

import one.py

import two.py

import three.py

* Now when we declare main.py everything inside it will be imported while using a particular module . If we don’t need the entire module to be imported we can specify explicitly here we are not importing entire module we are importing functions in it

Ex : from functools import reduce

* When the file size is huge we can use alias **as**

import Main as m

* \_\_init\_\_.py is a special file in a Python package directory. It signifies that the directory should be treated as a Python package. It can be an empty file, but it must be present for the interpreter to recognize the directory as a package. It is executed when the package is imported, allowing initialization code to be included

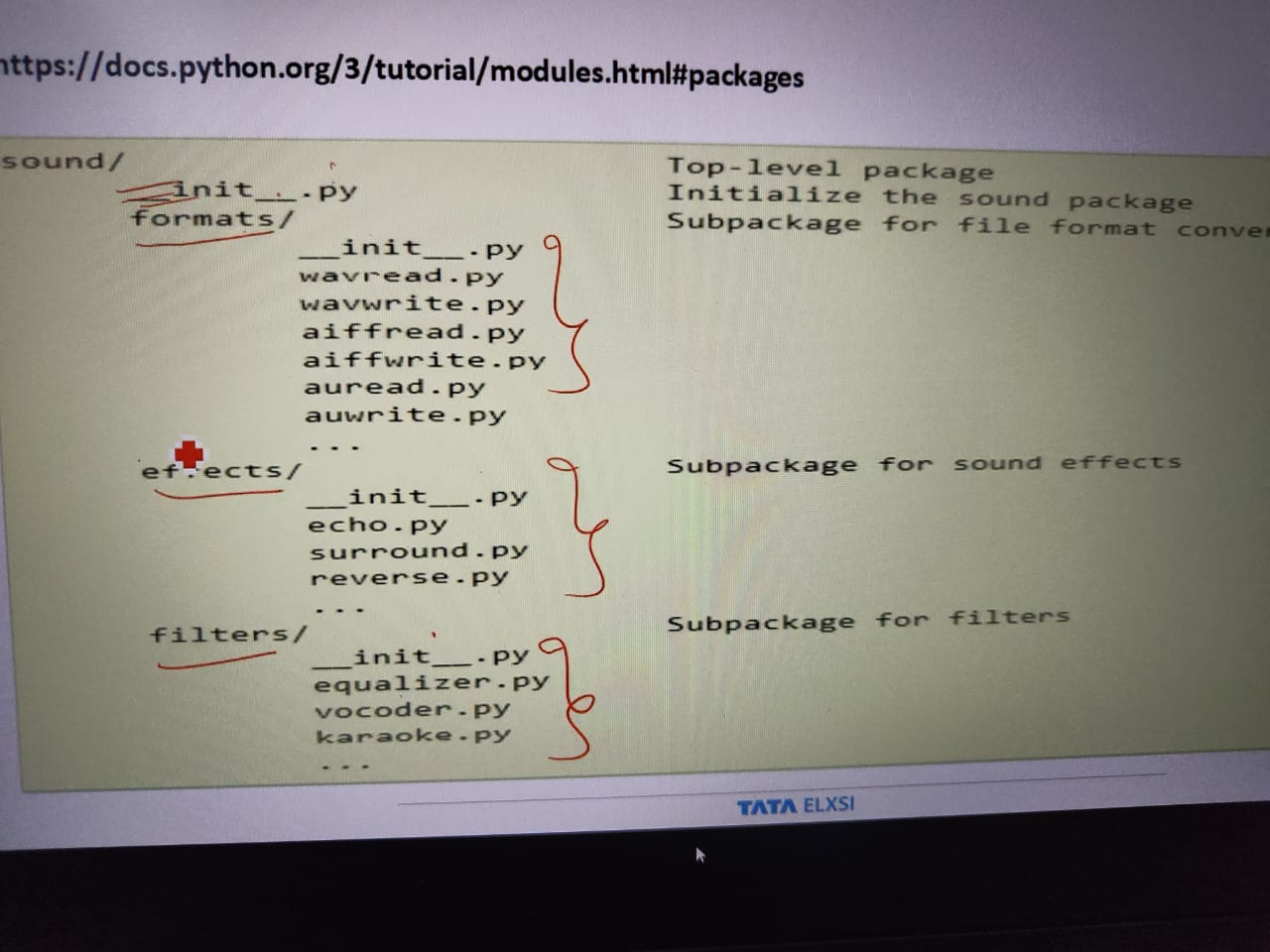
Ex:

|  |  |
| --- | --- |
| #fibo.py  def fibo(n):  a,b=0,1  d=0  while d < n-2 :  c=a+b  print(c , end=" ")  a=b  b=c  d=d+1 | from fibo import fibo  fibo(10)  “””  Reloaded modules: fibo  1 2 3 5 8 13 21 34  “”” |

Ex 2:

|  |  |
| --- | --- |
| import fibo  x=int(input("Enter a number "))  y=int(input("Enter a number "))  z=int(input("Enter 1 for add \nEnter 2 for sub \nEnter 3 for mul \nEnter 4 fro div \nEnter 5 fro rem "))  result =0  if z==1 :  result=fibo.add(x,y)  print("Added " , result)  elif z==2 :  result=fibo.sub(x,y)  print("Sub " , result)  elif z==3 :  result=fibo.mul(x,y)  print("Mul" , result)  elif z==4 :  result=fibo.div(x,y)  print("Div" , result)  elif z==5 :  result=fibo.rem(x,y)  print("Remainder " , result)  else :  print("Enter the number correctly ") | #fibo.py  def add(a,b):  return a+b  def sub(a,b):  return a-b  def mul(a,b):  return a\*b  def div(a,b):  return a/b  def rem(a,b):  return a%b |

**Ex 3 :**



User using this package can import individual modules from package

Ex : import sound.effects.echo.py (or ) import sound.filters.karaoke.py

**Random module :**

* Python has a built-in module that you can use to make random numbers.
* The random module has a set of methods:

Method Description

seed() Initialize the random number generator

getstate() Returns the current internal state of the random number generator

setstate() Restores the internal state of the random number generator

getrandbits() Returns a number representing the random bits

randrange() Returns a random number between the given range

randint() Returns a random number between the given range

shuffle() Takes a sequence and returns the sequence in a random order

sample() Returns a given sample of a sequence

random() Returns a random float number between 0 and 1

uniform() Returns a random float number between two given parameters

Ex :

|  |  |
| --- | --- |
| # random number generation  a=random.randint(1,10)  print("Random integer between 1- 10",a)  b=[1,23,4,45,45,656,56,43]  random.shuffle(b)  print("The b is shuffled ",b)  c=random.choice(b)  print("Choosing a choice from a list ",c) | 7  [1, 43, 656, 56, 23, 45, 4, 45]  56 |

|  |
| --- |
| **Prg:**  import random  integer\_num=random.randint(1,100)  print("Ramdom integer from 1 - 100 " , integer\_num)  float\_num=random.random()  print("Ramdom Float no argument needed " , float\_num)  range\_num=random.randrange(1,100)  print("Ramdom integer from 1 - 50 " , integer\_num)  list\_org=[1,2,3,4,5,6,7,8,9,10]  print("Original list :" , list\_org)  shiuffle\_list=random.shuffle(list\_org)  print("Original list shuffled :" , list\_org)  pick\_any=random.choice(list\_org)  print("Pick any fom the list : " , pick\_any ) |

**Regular Expressions :**

* Used to filter text/ string and it is a way for data cleaning and wrangling
* You have to import the re package

/^[Reg]ular[Ex]pression $/

* Identify exact piece of text
* Recommended to use raw strings

1. **Functions :**

* search : Find a match (object ) in the string
* find-all : return list with all matches
* split : return list separate by match
* sub : Find every match in it and replace it
* match : Match return the number the first occurrence of text in string
* group : group function used to merge the match found
* (findall and match are similar but the match finds the first occurrence of string )

Modifiers :

.. : Set of missed characters

^ : Start with

$ : Ends with

\* : Zero or more occurrence

+ : One or more occurrence

. : Any character

Identifier :

\d : Any number or digit

\s : First white spaces

\w : return string from [A-Z] or [a-z] or [0-9]

\W : return a String with no character

\b : return match found at beg or end

\B : Found the match found at middle

|  |  |
| --- | --- |
| import re  a="The rain in Spain in in in in inThe "  b="1 Hi Hello 23 BYe 345 tataaaaa goodbye 567567 words of India "  #findall function  j=re.findall("[Q-U][a-n]\w" , a)  c=re.findall("in" , a)  # use of raw string  g=re.findall(r"\BT\w+",a)  h=re.findall(r"[A-T][a-z]" ,a )  i=re.findall(r"\d{3,5}\s\w.{4}",b)  print("[Q-U][a-n]\w :",j)  print("in : ",c)  print("r\BT\w+ : ",g)  print("r[A-T][a-z] :" , h)  print("r\d{3,5}\s\w+ : " , i)  #using split  d=re.split("\s",a)  e=re.split("\W", a)  print("\s : ",d)  print("\W : ",e)  #usage of sub  f=re.sub("in","as" ,a , 3) # in converted as in string "a" and only in 3 occurences  print("in,as ,a , 3 : ",f)  # use of search  k="MIya maya Maiyer Mainer"  l=re.search(r"M[ae][yi]\w+" , k)  print("rM[ae][yi]\w+ :" , l.group())  #use of compile  m=re.compile(r"m[ai][yi]\w+")  n=m.findall(k)  print("pattern is : rm[ai][yi]\w+ and the n is m.findall " , n) | [Q-U][a-n]\w : ['The', 'The']  in : ['in', 'in', 'in', 'in', 'in', 'in', 'in', 'in']  r\BT\w+ : ['The']  r[A-T][a-z] : ['Th', 'Sp', 'Th']  r\d{3,5}\s\w+ : ['345 tataa', '67567 words']  \s : ['The', 'rain', 'in', 'Spain', 'in', 'in', 'in', 'in', 'inThe', '']  \W : ['The', 'rain', 'in', 'Spain', 'in', 'in', 'in', 'in', 'inThe', '']  in,as ,a , 3 : The raas as Spaas in in in in inThe  rM[ae][yi]\w+ : Maiyer  pattern is : rm[ai][yi]\w+ and the n is m.findall ['maya'] |

|  |
| --- |
| Detailed program :  import re  a="The rain in Spain in in in in inThe Qkshay "  b="1 Hi Hello 23 BYe 345 tataaaaa goodbye 567567 words of India "  #findall function  """/\*re.findall() to find all occurrences of a pattern that matches a capital letter between Q and U followed by a lowercase letter between a and n, and then followed by any word character."""  j=re.findall("[Q-U][a-n]\w+" , a)  print("[Q-U][a-n]\w+ :",j)  j=re.findall("[Q-U][a-n]\w" , a)  print("[Q-U][a-n]\w :",j)  j=re.findall("[Q-U][a-n]" , a)  print("[Q-U][a-n] :",j)  """  \w+ for finding till the completion of the pattern  \w for pattern with one echarecter at the end  noting only the pattern  """  c=re.findall("in" , a)  print("in : ",c)  #findall finds all the occurence of the pattern  # use of raw string  g=re.findall(r"\BT\w+",a)  print("r\BT\w+ : ",g)  """  \B: Matches a non-word boundary. | \B matches at any position that is not at the beginning or end of a word.  T: Matches the letter 'T'.  \w+: Matches one or more word characters.  """  h=re.findall(r"[A-T][a-z]" ,a )  print("r[A-T][a-z] :" , h)  """  [A-T] Starts from A to T  [a-z] next letter can ve from a-z  r for the rae string  """  i=re.findall(r"\d{3,5}\s\w.{4}",b)  print("r\d{3,5}\s\w+ : " , i)  """  This line finds all sequences of 3 to 5 digits followed by a space, then followed by any word character and exactly 4 characters after it in string b. The result is stored in the list i.  It had left the 23 as it needs a digit from 3 to 5  """  #using split  d=re.split("\s",a)  print("split(\"\\s\",a) : ",d)  e=re.split("\W", a)  print("split(\"\\W\",a) : ",d)  """  \s :The split helps in spliting the string based on the parameter passed as the input here we have given spaces  \W : the split is done on the Word by Word basis  """  #usage of sub  f=re.sub("in","as" ,a , 3) # in converted as in string "a" and only in 3 occurences  print("sub \(in,as ,a , 3\) : ",f)  # use of search  k="MIya maya Mainer Miiyer "  l=re.search(r"M[ia][yi]\w+" , k)  #find the first occurence and not the entire text  #pattern here mentioned is start with M and next letter can be a or e and 3rd letter can be y or i and followed by the wole word matches pattern  print("search\(r\"M[ie][yi]\w+\"\) :" , l.group())  #group used for the merging of the found charecter  #use of compile  m=re.compile(r"m[ai][yi]\w+")  print("compile\(r\"m[ai][yi]\w+\"\) " , m)  n=m.findall(k)  print("pattern is : rm[ai][yi]\w+ and the n is m.findall " , n)  """  re.compile() is used to compile the regular expression pattern r"m[ai][yi]\w+". The resulting compiled pattern object is stored in the variable m.  n = m.findall(k) : This line uses the compiled pattern m to find all occurrences of the pattern in the string k using findall(). The resulting matches are stored in the list n.  """ |

**UNPACK A COLLECTION :**

* If you have a collection of values in a list, tuple etc. Python allows you to extract the values into variables. This is called unpacking.

Ex:

fruits = ["apple", "banana", "cherry"]  
x, y, z = fruits  
print(x)  
print(y)  
print(z)

we can use print stmt for showing it in output screen .

**Variable in Python :**

1. Variable can hold values

Ex : a=234

1. Variable can be local and global :

Ex global var =1

var=132

1. A variable can refer the data kept in memory or another program loaded in memory
2. A variable can hold the functionality (A special feature of the python )

Ex : var = lambda input : input+10

print(var(12))

**Local Variable :**

* Local variable : variable created inside the method
* In function we can use the global variable outside the method by the \*\*global class\*\*

Example :

def my\_function():

# Define a local variable inside the function

x = 10

print("Inside the function, x is:", x)

# Call the function

my\_function()

**Global variable :**

* Global variable : variable in a python file (module ) is always global
* We can also access the global variable in other python module by importing
* Variables that are created outside of a function (as in all of the examples above) are known as global variables.

Ex : x = "awesome"

def myfunc():

print("Python is " + x)

myfunc()

If you create a variable with the same name inside a function, this variable will be local, and can only be used inside the function. The global variable with the same name will remain with the original value.

We can also use global keyword :

Ex: #with use of global keyword

b=93 #global

def myfu():

global b

b=334

print("Global var altered value " , b )

myfu()

print(b)

**DATA TYPES :**

Variables can store data of different types, and different types can do different things. Python has the following data types built-in by default, in these categories:

|  |  |
| --- | --- |
| Text Type: | str |
| Numeric Types: | int, float, complex |
| Sequence Types: | list, tuple, range |
| Mapping Type: | dict |
| Set Types: | set, frozenset |
| Boolean Type: | bool |
| Binary Types: | bytes, bytearray, memoryview |

Ex:

x = "Hello World" #str

x = 20 #int

x = 20.5 #float

x = 1j #complex

x = ["apple", "banana", "cherry"] #list

x = ("apple", "banana", "cherry") #tuple

x = range(6) #range

x = {"name" : "John", "age" : 36} #dict

x = {"apple", "banana", "cherry"} #set

**TYPE CONVERSION :**

Converting from one type to another with the int(), float(), and complex() methods

Ex :

x = 1 # int

y = 2.8 # float

z = 1j # complex

#convert from int to float:

a = float(x)

#convert from float to int:

b = int(y)

#convert from int to complex:

c = complex(x)

print("The actual value of x is " , x, " and it is converted to ",a, " and it is of type " ,type(a))

print("The actual value of y is " , y, " and it is converted to ",a, " and it is of type " ,type(b))

print("The actual value of z is " , z, " and it is converted to ",a, " and it is of type " ,type(c))

**Convert a variable value to binary :**

Ex:

salary=12323

bin\_bytes=bytes(salary)

**Boolean :**

If you evaluate any expression bool return true or false

|  |  |
| --- | --- |
| print(bool("Hello")) print(bool(15))  bool("abc") bool(123) bool(["apple", "cherry", "banana"]) | bool(False)  bool(None)  bool(0)  bool("")  bool(())  bool([])  bool({}) |

Irfan Python exercise :

"""

Created on Sun Mar 31 08:40:36 2024

@author: aksha

BigData Scenarios

"""

"""

Variables

1. Create 2 variables with x as 100 & y as 10 respectively and find the Multiplication and division of both and store in some val as z and z1.

x=100

y=10

z=x\*y

z1=x/y

print(z)

print(z1)

2. Create a as 2000 and find the division of a by y (created in step 1) and reassign a with the divided result (200).

a=2000

print("Entered value is " , a)

y=10

a=int(a/y)

print("After operation " , a)

3. Prove Python is Dynamically Typed Language: Create x:int=100, then assign the x to y, but the datatype of y has to be of type string. (think about using some function like str()). Print the type of y and x

x:int=100

y:str=str(x)

print(type(x))

print(type(y))

4. Prove Python has dynamic inference feature

x=5

print(type(x)) # without any datatype x is automatically identified as integer

y="String"

print(type(y)) # without any datatype y is identified as string

5. Prove Python is Strongly Typed Language

x = 18 # automatically x is identified as integer by python due to dynamix inference

y = "2"

#Python's strong typing nature where operations between incompatible types result in errors

z=x+int(y) # converting the string to integer

print( z ) # Here the operations cant be done because the x is integer and y is string

6. Create variables a,b,c,d assigned with 10,20,30,40 respectively

a,b,c,d=10,20,30,40

print(a)

print(b)

print(c)

print(d)

7. Prove Python variables are case sensitive

#These variable are case sensitive

var=12 # var is a variable

Var=23 # Var is another variable

print(var)

print(Var)

8. Prove variable name can’t start with numbers or cannot contains special character other than \_

#9var=34 The variable cant start with numbers

#$var=34 The variable cant start with special charecters

9. Show some examples of when do we use single, double and triple (single/double) quotes

string1='Akshaya Baalaji'

print(string1) # with single quote

string2="Akshaya Baalaji"

print(string2) # with double quote which work as same as single quote

##string3="""#This is a multi line value

# HI

# Bye

# See

# you

"""

print(string3) # for multi line value

10. Show an examples to use arithmetic, comparison, relational and logical operators.

# Arithmetic operators

x = 10

y = 3

print("Arithmetic operators:")

print("x + y =", x + y) # Addition

print("x - y =", x - y) # Subtraction

print("x \* y =", x \* y) # Multiplication

print("x / y =", x / y) # Division

print("x // y =", x // y) # Floor Division

print("x % y =", x % y) # Modulus

print("x \*\* y =", x \*\* y) # Exponentiation

print()

# Comparison operators

print("Comparison operators:")

print("x == y is", x == y) # Equal to

print("x != y is", x != y) # Not equal to

print("x > y is", x > y) # Greater than

print("x < y is", x < y) # Less than

print("x >= y is", x >= y) # Greater than or equal to

print("x <= y is", x <= y) # Less than or equal to

print()

# Relational operators

print("Relational operators:")

a = [1, 2, 3]

b = [1, 2, 3]

c = a

print("a is b is", a is b) # Identity - checks if a and b reference the same object

print("a is c is", a is c) # Identity - checks if a and c reference the same object

print("a == b is", a == b) # Equality - checks if the values of a and b are the same

print()

# Logical operators

print("Logical operators:")

p = True

q = False

print("p and q is", p and q) # Logical AND

print("p or q is", p or q) # Logical OR

print("not p is", not p) # Logical NOT

Conditional Structures

11. Write a program to find the greatest of 3 numbers

#using list built in functions

make\_list=[a,b,c] # making the list from the variables

print(max(make\_list));

#we can take the greatest by sorting and taking the last element in list

make\_list.sort(); #sort will sort the list

print("Taking the last elemrent from the list ",make\_list[-1])

#using simple conditional structure

a=10

c=10

b=30

if a==b and a==c :

print("All are same ")

elif a>=b and a>=c :

print(a, " is the greatest ")

elif b>=a and b>=c :

print(b, " is the greatest ")

else :

print(c, " is the greatest ")

#using nested conditional structure

a=900

b=900

c=100

if a==b and b==c:

print("All are same ")

else :

if a>=b and a>=c:

print("A is greater ")

elif b>=a and b>=c :

print("B is greater ")

else :

print("C is greater ")

12. Write a single program to find the given number is even or

whether it is negative and print the output as

(the given number is even but not negative or the given number is not even but negative or

the given number is neither negative nor even)

number=int(input("Enter a number : "))

if number%2==0 and number>0 :

print("The given number is even but not negative")

elif number<0 and number%2!=0 :

print("The given number is not even but negative")

elif number>0 and number%2!=0:

print("The given number is not negative and not even")

else:

print("The given number is neither negative nor even")

13. Write a nested if then else to print the course fees - check if student choosing bigdata, then fees is 25000,

if student choosing spark then fees is 15000, if the student choosing datascience then check if machinelearning then 25000

or if deep learning then 45000 otherwise if both then 25000+25000.

course=input("Enter the course you want to register : ")

subcourse:str

if course in 'bigdata':

print("Your fees is 25000")

elif course in 'spark':

print("Your fees is 15000")

elif course in 'datascience':

subcourse=input("..............Subcourses available................\nmachine learning or deep learning want or both \nEnter the course want to register in data science : ")

if subcourse in 'machine learning':

print("Your fees is 25000")

elif subcourse in 'deep learning':

print("Your fees is 45000")

else :

print("The fees is 20000+25000")

else:

print("Enter the correct course ")

14. Check whether the given string is palindrome or not (try to use some function like reverse).

For eg: x="madam" and y="madam", if x matches with y then print as "palindrome" else "not a palindrome".

string = input("Enter a string ")

reverse\_way1 = string[::-1]

print("The string you Entered : " , string)

print("The Reverse string done by indexing : " , reverse\_way1)

reverse\_way2:str=""

for i in string :

reverse\_way2 = i + reverse\_way2

print("The Reverse string done by for loop : ",reverse\_way2)

print()

if reverse\_way1 in string :

print("Entered string is a Palindrome ")

else :

print("Entered string is not a palindrome ")

15. Check whether the x=100 is an integer or string. (try to use some functions like str or upper function etc

to execute this use case) or use isinstanceof(variablename,datatype) function.

string="100"

x=100

#isinstance check string is a datatype of str type

if isinstance(string, str):

print("It is a string ")

#isinstance check x is a datatype of int type

if isinstance(x , int ):

print("It is a integer ")

Control Statements

16. Write a program using for loop to print even numbers and odd numbers in the below range of data

(generate using range function) [5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

output should be with even as 6,8,10,12,14,16,18,20 and odd as 5,7,9,11,13,15,17,19.

for i in range (5,21,1):

if i%2==0 :

print (i , end=" ")

print()

else:

print (i , end=" ")

even=[]

odd=[]

for i in range (5,21,1):

if i%2==0 :

even.append(i)

else:

odd.append(i)

print("Odd list is " , odd)

print("Even list is " , even)

17. Write a while loop to loop from 0 till 21 with the increment of 3,

the result should be exactly 3,6,9,12,15,18 and store this result in a list

i=0

incremented\_list=[]

while i<21:

i=i+3

incremented\_list.append(i)

print("The list " , incremented\_list)

18. Write a for or while loop to print the cube of 4,

result should be 4\*4\*4=64 (initiate some variable outside the loop with 4 and loop through 3 times to achieve the result)

loop=3

i=0

number=int(input("Enter a number : "))

cube\_while=1

while i<loop:

cube\_while=cube\_while\*number

i=i+1

print(cube\_while)

print(f"The number of time iterated {loop} and the cube found out is {cube\_while}" )

cube\_for=1

for i in range (0,loop,1):

cube\_for=cube\_for\*number

print(f"The number of time iterated {loop} and the cube found out is {cube\_for}" )

19. Create a list as sal\_lst=[10000,20000,30000,10000,15000], loop through the list

and add 1000 bonus to the salary and store in another list sal\_bonus\_lst=[11000,21000,31000,11000,16000]

then store the bonus applied salary in another list where sal>11000

sal\_lst=[10000,20000,30000,10000,15000]

bonus=1000

add=0

sal\_bonus\_list=[]

print("The list before bonus " , sal\_lst)

for i in sal\_lst:

add=i+bonus

sal\_bonus\_list.append(add)

print("The list after adding bonus " , sal\_bonus\_list)

20. Write a do while loop to print “Inceptez technologies” n number of times as per the

input you get from the user. Minimum it has to be printed at least one time regardless of the user input.

i=0

loop=int(input("Enter the number of time loop has to happen : "))

#if the loop is 0 it prints infinite time

if loop < 1:

loop = 1

while True :

print("Inceptez Technologies ")

i=i+1

if i==loop :

break

21. From the given list of list of elements produce the following output using nested for loop

lst1=[[10,20],[30,40,50],[60,70,80]], calculate the sum of all number, calculate the min value and

the max value of all the elements in the lst1.

lst1=[[10,20],[30,40,50],[60,70,80]]

#to make the array of array to single array

list2=sum(lst1,[])

print(list2)

print("The max element in array is " , max(list2))

print("The min element in array is " , min(list2))

sum\_of\_ele=0

for i in lst1 :

for j in i:

sum\_of\_ele=sum\_of\_ele+j

print("The sum of the element is : " , sum\_of\_ele )

22. Create a looping construct to create 3 tables upto 10 values. Output should be like this…

1 x 3 = 3

2 x 3 = 6

3 x 3 = 9

.

.

10 x 3 = 30

number = int(input("Enter the number for table : "))

#using simple for

for i in range (0,11,1):

print(f"{i} x {number} = " , number\*i)

#using nested for loop

table=int(input("Enter the number for table "))

for i in range (0,11,1):

for j in range (0,11,1):

if j == table:

print(f"{i} x {table} = " , j\*i)

#using nested for and using continue constructs

number = int(input("Enter the number for table : "))

print("Nested loop :")

# using nested for loop

for i in range (1,number+1,1):

for j in range (0,11,1):

if(i==5 or i ==10):

continue

else :

print (f"{i} x {j} = ", i\*j )

"""

"""

Pattern printing :

n = int(input("Enter the number for the pattern "))

for i in range (0,n , 1):

for j in range (0,i,1):

print(" " , end=" ")

for k in range (n,i,-1):

print("\*" , end=" ")

print()

"""

"""

#Usecase2 related to exit controlled loop (do while loop) + break & continue:

#Create a scheduler program to run a code minimum once or continue to run multiple hours + skipping odd hours

start\_time = int(input("Enter the start time in 24 hours format "))

end\_time = int(input("Enter the end time in 24 hours format "))

#using for loop

print("Using for loop")

for i in range (start\_time , end\_time+1 ,1):

if i%2==0 :

print("Script is running ")

else :

continue;

#using while loop

print("Using do while ")

while start\_time <= end\_time:

if start\_time%2==0 :

print("Script is running " , start\_time)

start\_time=start\_time+1

else :

start\_time=start\_time+1

continue;

"""

"""

print("Usecase3 : Complex Tuple -

tuple(familyid:int,familyname:str,familymembers:list[dict{k:v,k:v,k:dict{k:v,k:v}}])")

dictionary={"Family id " : 1 , "Family\_name" : "AB Family" ,"fammily\_members" : [{"Father" : "Senthilraj" , "Mother name" : "Vijiyalakshmi" , "Son" : "Akshaya Baalaji " }] }

# tuple elements / members are int , string , list [dictionary inside list ]

tuplee=(1 , "AB Family" ,[{"Father" : "Senthilraj" , "Mother name" : "Vijiyalakshmi" , "Son" : "Akshaya Baalaji " }] )

print(dictionary);

print("Complex Tuple created is : " , tuplee)

complextup=(1,

"Madison's Family",

[{"gender":"male","Relation":"self","personalinfo":{"title":"Ms","name":"Madison"}},

{"gender":"female","Relation":"spouse","personalinfo":{"title":"Mrs","name":"Elisa"}},

{"gender":"female","Relation":"daughter","personalinfo":{"title":"Miss","name":"Hanna","hobby":"book reading"}},

{"gender":"male","Relation":"son","personalinfo":{"title":"Master","name":"Dave","schooling":True}}])

print("1 id is ",complextup[0])

print("2 Family name is ",complextup[1])

print("3 gender of first list element is ",complextup[2][0].get("gender"))

print("4 relation of first list element is ",complextup[2][0].get("Relation"))

print("5 personalinfo of the first list element is ",complextup[2][0].get("personalinfo"))

print("5 i personalinfo of the first list element is ",complextup[2][0]["personalinfo"])

print("6 personalinfo title of the first list element is ",complextup[2][0]["personalinfo"]["title"])

print("7 personalinfo name of the second list element is " ,complextup[2][1]["personalinfo"]["name"])

print("8 personalinfo hobby of the third list element is " ,complextup[2][2]["personalinfo"]["hobby"])

print("9 personalinfo schooling info of the fourth list element is " ,complextup[2][3]["personalinfo"]["schooling"])

"""

Python scenarios

Collections: List, Dictionary, Tuple and Set

"""

"""

23. Create a list with a range of 10 values starting from 2 to 11 and prove mutability by updating the 3rd element

with 100 and prove resizable properties by adding 100 in the 5th position.

listt=list(range(2,12))

print(listt)

#mutability by the updating the 3rd element with 100

listt[2] = 100

print("The list after the modification " , listt)

#add the element (100) in the 5th position

listt.insert(4,100)

print("LIst after the insertion of the fifth element " , listt)

24. Create a tuple of 2 fields eg. ("Inceptez","Technologies","Pvt","Ltd"),

prove immutability and non resizable nature, access the 2nd and 4th fields and store in another tuple.

tuple1=("Inceptez","Technologies","Pvt","Ltd")

print("Original tuple is : ",tuple1)

#cant be done due to python tuple immutable nature

#tuple1.insert(2,"12")

tuple\_new = (12,23)

tuple2=tuple1.\_\_add\_\_(tuple\_new)

print("Tuple after the adding with new tuple : " , tuple2)

#The tuple after the deletion is

tuple3=tuple(i for i in tuple2 if i!=23 )

print("The tuple after the deletion is : " , tuple3 )

#the tuple after the insertion

tuple4=tuple(i for i in tuple1 if i!="Technologies")

print("The tuple after the deletion is : ",tuple4)

25. Convert the list of tuples [("Inceptez","Technologies"),("Apple","Incorporation")]

to list of dictionary type, using for loop as given below [{"Inceptez":"Technologies"},{"Apple":"Incorporation"}] ,

once the list of dictionary is arrived print only "Incorporation" by passing "Apple" as a key using dict["Apple"] and

dict.get("Apple") and try with dict["Apple1"] and dict.get("Apple1") then find the difference between get and using[] notation.

list1=[("Inceptez","Technologies"),("Apple","Incorporation")]

list2=[{ i[0] : i[1] } for i in list1 ]

print("The original list is : " , list1)

print("List after converting to dictionary : " , list2)

#difference between get and []

print("The apple key has value : " , list2[1].get("Apple") )

print("The apple key has value : " , list2[1]["Apple"] )

26. Create a list of tuple as given below and delete all duplicate tuples of the list

lst=[("Inceptez","Technologies"),("Apple","Incorporation"),("Inceptez","Technologies"),("Inceptez","Technologies")]

lst=[("Inceptez","Technologies"),("Apple","Incorporation"),("Inceptez","Technologies"),("Inceptez","Technologies")]

print("The original tuple is : " , lst)

list2=list(set(lst))

print(set1)

27. Append ("Intel","Corp") in the above de duplicated list

lst=[("Inceptez","Technologies"),("Apple","Incorporation"),("Inceptez","Technologies"),("Inceptez","Technologies")]

print("The original tuple is : " , lst)

list2=list(set(lst))

print(list2)

tuple1=("Intel","Corp")

list2.append(tuple1)

print("The list after the appending : " , list2)

28. Convert the lst\_dict= [{"Inceptez":"Technologies"},{"Apple":"Incorporation"}] to

lst1=["Inceptez","Apple"] , think about using for loop, list() function, keys function and list append functions to achieve this.

lst\_dict= [{"Inceptez":"Technologies"},{"Apple":"Incorporation"}]

print("The original list of dictionary : " , lst\_dict)

list\_of\_key=[]

for i in lst\_dict:

list\_of\_key.extend(i.keys())

print(list\_of\_key)

29. Create a list of values lst=[10,20,40,30,20], find the first, last values of the list,

sort the list in ascending order, sort in descending order, print the minumum and maximum values of the descending sorted list,

find the sum of all elements in the list, remove the number 30 and 20 from the list.

from functools import reduce

lst=[10,20,40,30,20]

print("The first value of the list is " , lst[0])

print("The last value of the list is " , lst[::-1])

#for sorting the list we use the sort function

lst.sort()

print("The sorted list is " , lst)

lst.reverse()

print("The sorted list in reversae is " , lst)

# Max and minimun element in array is

print("The Max element in the array is " , lst[0])

print("The min element in an array is " , lst[::-1])

#sum of element in the list

max=reduce(lambda x,y : x+y , lst)

print("The Sum of all element in list is " , max)

#remove 30 and 20 from the list using the delete function

set1=set(lst)

print("The list after converting to the set " , set1)

lst=list(set1) # converting back the set to list

lst.remove(30)

#here the 20 is 2 times in the list so we have converted the list to set and again to list

lst.remove(20)

print("The list now is " , lst)

30. Do the above same (step 25) operation in the tuple of elements tup=(10,20,40,30,20)

31. Convert the string to list from str1="Inceptez Technologies Pvt Ltd" to lst\_str1=['Inceptez', 'Technologies', 'Pvt', 'Ltd']

str1="Inceptez Technologies Pvt Ltd"

lst\_str1=list(str1.split(" "))

print("The list after the string split " , lst\_str1)

lst\_str2=list(str1)

print("The list after directly converting it to list using list function " , lst\_str2)

32. With the below given data in the format of list(list(elements))

emplstlst= [["1", ("Arun","Kumar"), "10000"],["2", ("Bala","Mohan"), "12000"]]

Display the below output for all of the 5 given simple scenarios

a. Convert the first element of the above list into tuple

("1", ("Arun","Kumar"), "10000")

b. Print the second element's second element and reverse the first and last name as given below

("Mohan","Bala")

c. Convert the emplstlst into tuples(tuples)

emplstlst= (("1", ("Arun","Kumar"), "10000"),("2", ("Bala","Mohan"), "12000"))

d. Add all salary of the above list

22000

emplstlst= [["1", ("Arun","Kumar"), "10000"],["2", ("Bala","Mohan"), "12000"]]

tuple1=tuple(emplstlst[0])

print("The first element is converted to tuple type " , tuple1)

#the below is error because tuple is not mutable

#print("second element's second element and reverse the first and last name " , emplstlst[1][1].reverse())

lst2=list(emplstlst[1][1])

# reversing the list

lst2.reverse()

tuple1 = tuple(lst2)

print("second element's second element and reverse the first and last name " , tuple1)

emplstlst[0] = tuple(emplstlst[0])

emplstlst[1] = tuple(emplstlst[1])

main\_tuple= tuple(emplstlst)

print(main\_tuple)

"""

**Object Oriented Programming :**

1. Class

2. Object

3. Inheritance

4. Polymorphism

5. Abstraction

6. Encapsulation

**Class :**

* class ClassName() : name must be (InitCap) Initial Capital
* Functions inside the class is known as \*\*members\*\*
* Member function inside the class need at least one argument so it will pass the self function .
* Self is the keyword / parameter / argument to make the method associated with the class which is mandatory
* If we won’t pass no arguments it creates a error because the function need atleast one argument

Example :

|  |
| --- |
| # When using with Class.Funciton\_name  class ClassName():  def funciton(self): # Member / Child method  print("This is inside the function and the argument passed " , self )  ClassName.funciton("Hi")  # When using the object for the class the funciton / Member of the class need atleast one argument  class ClassName\_1():  def funciton(self, argument): # Member / Child method  print("This is inside the function " , self, "argument is " , argument )  object = ClassName\_1()# when using the () the class will be loaded into memory  object.funciton("Hi") |

IMPORTANT : self can be named as any in python

**Three ways to invoke class and method developed by us :**

1. Importing the entire module / class

2. Use the from package subpackage import module ( But this is only for accessing the particular module)

1. from filename import (class1 , class2 ) #here the class1 and class2 are only need to be imported from the filename (file)

3. Use from package import \* (For using all the program )

**Class** : Class is a template or blueprint that organizes / contains programs in a form of variable , method , functions

**Member** : Any program given inside the module or class

**Object** : Variable hold the instance / instantiation / initialization of a class is an object

**Constructor** : Constructor is a specialized methodology used to create object by loading class in memory

**Types of constructor :**

1. Default constructor

2. Parameterized constructor

3. Non-Parameterized constructor

1. Default Constructor :

When we are not creating the constructor it is created by python internally by default

Example :

|  |
| --- |
| class NewClass():  def \_\_init\_\_(self) :  pass |

1. Non-Parameterized Constructor :

Example :

|  |
| --- |
| class NewClass():  def \_\_init\_\_(self) :  self.x = 123  print("This is the default constructor called automatically when we create object for the class ")  def function\_1(self , a,b):  print("This is the function inside the class value fetched from constructor " , self.x , a , b )  obj = NewClass()  obj.function\_1("Argument\_\_1 " , "Argument\_\_2 ") |

1. Parameterized constructor :

Example :

|  |
| --- |
| class NewClass():  def \_\_init\_\_(self , parameter) :  self.x = parameter  print("This is the default constructor called automatically when we create object for the class ")  def function\_1(self , a,b):  print("This is the function inside the class \"" , self.x ,"\" this value fetched from constructor " , a , b )  obj = NewClass("This is a parameter")  obj.function\_1("Argument\_\_1 " , "Argument\_\_2 ") |