Python Help



This Documents Help to Understand the Basics of Python. Prepared & coded by

* Akshit Agarwal

2024

Use hash (#) for single line comments & triple quotes (‘’’ Comment ‘’’) for multi-line comment

##This is a single line comment  
  
*'''  
This  
is  
multiline  
comment  
'''*

Use a backslash (\) to continue the statement to the next line.

##Line Continuation  
  
Total = 1+2+3+4+5\  
 +6+7+8+9+0  
print(Total)

Python can execute multiple statements on a single line. Statements can be separated using semicolon (;)

#Multiple statements on a single line.   
  
x=5;y=10;z=x+y;  
print(z)

Type of variables can be changes runtime in python

#Type Interface  
  
var = 10  
print(type(var))  
var = "Hello World"  
print(type(var))  
var = 3.14  
print(type(var))

Declaration & Assignment of values to variable

# declaring a variable & assigning value  
  
name = "Jack"  
age = 25  
height = 6.1  
isAdult = True  
  
print("Name is ",name)  
print("Age is ",age)  
print("Height is ",height)  
print("isAdult is ",isAdult)

Taking inputs from the user

age = input("Enter You Age: ")  
print("Your Entered : ",age)

Commonly used Operators in python

Num\_1 = 15  
Num\_2 = 23  
  
add = Num\_1+Num\_2  
sub = Num\_1-Num\_2  
mul = Num\_1\*Num\_2  
div = Num\_2/Num\_1  
floor\_div = Num\_2//Num\_1  
mod = Num\_2%Num\_1  
expo = Num\_1\*\*2  
  
print("Addition : ", add)  
print("Subtraction : ", sub)  
print("Multiplication : ", mul)  
print("Division : ", div)  
print("Floor Division : ", floor\_div)  
print("Modulus : ", mod)  
print("Exponential : ", expo)

**Conditional Statements in python.**

1. IF statement

if age>=18:  
 print("Adult, You are eligible to vote")

1. IF-ELSE statement

if age>=18:  
 print("Adult, You are eligible to vote")  
else:  
 print("Not Adult, Not eligible to vote")

1. ELIF statement (else if statement)

age = 10  
if age<12:  
 print("You are a kid")  
elif age<18:  
 print("You are a teenager")  
else:  
 print("You are adult")

1. Nested if-else statements

number = 12  
if number >0:  
 print("Positive Number")  
 if number%2==0:  
 print("Even")  
 else:  
 print("odd")  
else:  
 print("Negative Number")

Assignment for Operators & Conditional Statements

#Assignment for Operators & Conditional Statements  
  
num1= float(input("Enter num1"))  
num2= float(input("Enter num2"))  
operation = input("Enter + or - or \* or / or // or % or \*\*")  
  
if operation == "+":  
 print(num1+num2)  
elif operation == "-":  
 print(num1-num2)  
elif operation == "\*":  
 print(num1\*num2)  
elif operation == "/":  
 print(num1/num2)  
elif operation == "//":  
 print(num1//num2)  
elif operation == "%":  
 print(num1%num2)  
elif operation == "\*\*":  
 print(num1\*\*num2)  
else:  
 print("Invalid Operation")

**Loops in Python**

1. FOR loops

#for loops  
  
for i in range(7):  
 print("i = ",i)

NOTE : This will print from 0 to 6. The last value is not included in the loop. Below code show if we want to start the loop from some specific number

for i in range(10,16):  
 print("i = ",i)

This code will print from 10 to 15. Again the last value will not be printed. Below code show if we want to step some number. DEFAULT STEP VALUE IS 1

for i in range(10,22,2):  
 print("i = ",i)

this code will print number as 10,12,14,16,18,20.This jumps two number. Below code respesents loopign backwards.

for i in range(10,0,-1):  
 print("i = ",i)

This code will print 10, 9,8,7,6,5,4,3,2,1. Keeping the jump as -1.

1. WHILE loop

#While loops  
count = 0  
while count<6:  
 print(count)  
 count = count + 1

1. BREAK & CONTINUE STATEMENT

#break & continue statements  
for i in range(20):  
 if i == 10:  
 break  
 print("Loop Number :",i)

OUTPUT :   
Loop Number : 0

Loop Number : 1

Loop Number : 2

Loop Number : 3

Loop Number : 4

Loop Number : 5

Loop Number : 6

Loop Number : 7

Loop Number : 8

Loop Number : 9

for i in range(10):  
 if i%2==0:  
 continue  
 print("Odd Numbers are",i)

OUTPUT:

Odd Numbers are 1

Odd Numbers are 3

Odd Numbers are 5

Odd Numbers are 7

Odd Numbers are 9

The ***break*** keyword breaks the current loop & exits the loop. Whereas the ***continue*** keyword skips the current iteration of the loop & executes the next loop

1. Nested FOR loops in python

##Netsed FOR loop  
  
for i in range(3):  
 for j in range(2):  
 print(f"i : {i} and j : {j}")  
 print("j++")  
 print("i++")  
print("EOL")

OUTPUT :

i : 0 and j : 0

j++

i : 0 and j : 1

j++

i++

i : 1 and j : 0

j++

i : 1 and j : 1

j++

i++

i : 2 and j : 0

j++

i : 2 and j : 1

j++

i++

EOL

Example for calculating the sun of first n natural numbers using while & for loop.

#Example of FOR & WHILE loop  
##Calculate the sum of first n natural number  
  
n=10;sum=0;count=1  
while count <= n :  
 sum = sum+count  
 count=count+1  
print("Sum of first ",n,"numbers is = ",sum) 🡪 O/P : Sum of first 10 numbers is = 55

res=0  
for i in range(20):  
 res = res + i  
print(res) 🡪 190  
print("EOFL")

**Inbuilt Data Structures of Python**

1. **LIST**

### List in Python.  
  
myList = []  
print(type(myList)) 🡪 <class 'list'>  
fruits = ["Apple", "Banana", "Chiku" , "Mango" , "Cherry"]  
print(fruits) 🡪 ['Apple', 'Banana', 'Chiku', 'Mango', 'Cherry'] # prints all the elements of the list   
print(fruits[2]) 🡪 Chiku # prints the element at 2nd index   
print(fruits[1:4]) 🡪 ['Banana', 'Chiku', 'Mango'] # prints the elements from index 1st till 4th (4th index will not be printed)  
print(fruits[1:]) 🡪 ['Banana', 'Chiku', 'Mango', 'Cherry'] # prints the all elements from index 1st till last  
print(fruits[-1])🡪 Cherry #prints the last element of the list.

* 1. List Methods

fruits.append("Kivi") # add the passed element at the end of the list  
print(fruits) 🡪 ['Apple', 'Banana', 'Chiku', 'Mango', 'Cherry', 'Kivi']  
  
fruits.insert(1,"Grapes") # add the passed element at the index 1st  
print(fruits) 🡪 ['Apple', 'Grapes', 'Banana', 'Chiku', 'Mango', 'Cherry', 'Kivi']  
  
fruits.remove("Chiku") # removes the first occurrence of the passed element  
print(fruits) 🡪 ['Apple', 'Grapes', 'Banana', 'Mango', 'Cherry', 'Kivi']  
  
popped\_fruit= fruits.pop() #removes & returns the last element of the list.

print(popped\_fruit) 🡪 Kivi

print(fruits)🡪 ['Apple', 'Grapes', 'Banana', 'Mango', 'Cherry']

indexMango=fruits.index("Mango") # returns the first index position of the element passed  
print("Index of Mango : ",indexMango) 🡪 Index of Mango : 3

fruits.insert(1,"Apple") #adds the passed element at the index passed.  
print("Total Apple in the list are ",fruits.count("Apple")) #counts the number element passed 🡪 Total Apple in the list are 2  
  
fruits.sort()#sorts the list in ascending order.  
print("Sorted Fruits : ",fruits) 🡪 Sorted Fruits : ['Apple', 'Apple', 'Banana', 'Cherry', 'Grapes', 'Mango']  
  
fruits.reverse()#it reverses the list  
print("Reversed List : ",fruits) 🡪 Reversed List : ['Mango', 'Grapes', 'Cherry', 'Banana', 'Apple', 'Apple']  
  
fruits.clear() #clears the list  
print(fruits) 🡪 []

1.2 Slicing the list

## Slicing the list.  
nums = [1,2,3,4,5,6,7,8,9,10]  
print("nums[2:5] =",nums[2:5]) 🡪 nums[2:5] = [3, 4, 5]  
print("nums[:5]=",nums[:5]) 🡪 nums[:5]= [1, 2, 3, 4, 5]  
print("nums[5:]=",nums[5:]) 🡪 nums[5:]= [6, 7, 8, 9, 10]  
print("nums[::2]=",nums[::2])🡪 nums[::2]= [1, 3, 5, 7, 9]  
print("nums[::-1]=",nums[::-1]) 🡪 nums[::-1]= [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]  
print("nums[::-2]=",nums[::-2]) 🡪 nums[::-2]= [10, 8, 6, 4, 2]

1.3 List Comprehension :

🡪 Basic List comprehension = [expression for item in iterable]

* With condition = [expression for item in iterable if condition]

##List Comprehension  
print("----------------------------------------------------------------")  
lst= []  
for x in range(10):  
 lst.append(x\*\*2)  
print(lst) 🡪 [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]  
  
even = [z\*\*2 for z in range(20) if z%2==0 ]  
print("Even List : " ,even) 🡪 Even List : [0, 4, 16, 36, 64, 100, 144, 196, 256, 324, 400]

**Dictionary in Python**

Two ways of defining dictionaries in python : by using curly braces or by using dict() method of python

### Dictionary In Python  
Dic1 = {}   
dic1 = dict()  
print(type(Dic1)) 🡪 <class 'dict'>  
print(type(dic1)) 🡪 <class 'dict'>

emp = {"Name":"Akshit","Age":"24","Gender":"Male"}

#Display the dictionary  
print(emp) 🡪 {'Name': 'Akshit', 'Age': '24', 'Gender': 'Male'}

#Display all the keys of the dict  
print(emp.keys()) 🡪 dict\_keys(['Name', 'Age', 'Gender'])

#display all the values of the dict  
print(emp.values()) 🡪 dict\_values(['Akshit', '24', 'Male'])

#display all the items of dict  
print(emp.items()) 🡪 dict\_items([('Name', 'Akshit'), ('Age', '24'), ('Gender', 'Male')])

**Coping a Dictionary**

this will not a new memory in system. emp1 will be used a ref to emp. any changes made to emp will also be changed to emp1

# to make a copy of a dictionary  
emp1 = emp   
print("EMP",emp) 🡪 EMP {'Name': 'Akshit', 'Age': '24', 'Gender': 'Male'}  
print("EMP1",emp1) 🡪 EMP1 {'Name': 'Akshit', 'Age': '24', 'Gender': 'Male'}  
  
emp["Age"]=10  
  
print("EMP",emp) 🡪 EMP {'Name': 'Akshit', 'Age': 10, 'Gender': 'Male'}  
print("EMP1",emp1) 🡪 EMP1 {'Name': 'Akshit', 'Age': 10, 'Gender': 'Male'}  
  
#Copy a dict This is know as shallow copy

emp\_new = emp.copy()  
emp["Age"]=25  
print("EMP",emp) 🡪 EMP {'Name': 'Akshit', 'Age': 25, 'Gender': 'Male'}  
print("EMP\_NEW",emp\_new) 🡪EMP\_NEW {'Name': 'Akshit', 'Age': 10, 'Gender': 'Male'}

**LAMBDA FUNCTIONS IN PYHTON**

Lambda functions are defined by the keyword lambda. These functions are commonly used for small functions which have multiple number of arguments but only one expression.

#LAMBDA FUNCTIONS  
print("-----------------------LAMBDA FUNCTIONS-----------------------")  
  
'''  
Syntax to define the lambda function   
  
fun\_name = lambda args : expressions  
example :   
sq\_num = lambda x : x\*\*2   
  
'''  
  
def add(num1 , num2):  
 return num1+num2  
print("Using add function ",add(10,20)) 🡪 Using add function 30   
  
add1 = lambda num1,num2 : num1+num2  
print("Using the Lambda function ",add1(50,30)) 🡪 Using the Lambda function 80

**MAP FUNCTION OF PYTHON**

The Map function is used to apply a given function to all the element of the list or any iterable.

print("-----------------------MAP FUNCTIONS-----------------------")  
'''  
Syntax to define the map function without lambda   
map (fun\_name,iterables)  
Example is shown below  
'''  
def sqr(sq):  
 return sq\*\*2  
sqr(10)  
  
sqrs = [1,2,3,4,5,6,7,8]  
print(list(map(sqr,sqrs))) 🡪 [1, 4, 9, 16, 25, 36, 49, 64]  
  
  
print("-----------------------MAP FUNCTIONS WITH LAMBDA-----------------------")  
'''  
Syntax to define the map function with lambda   
map (lambda args:expression ,iterables)  
Example is shown below  
'''  
n1=[1,2,3,4,5,6,7,8]  
print(list(map(lambda x:x\*\*2,n1))) 🡪 [1, 4, 9, 16, 25, 36, 49, 64]

**FILTER FUNCTION IN PYTHON**

The filter() function constructs an iterator from element of an iterable for which a function returns true. It is used to filter out elements from a list or any iterable based on conditions.

print("-----------------------FILTER FUNCTION-----------------------")  
'''  
Syntax to define filer function   
 filer(fun\_name,iterables) --> Regular Function  
 filer(lambda args:expression,iterables) --> Lambda function   
 filer(lambda args:expression and expression, iterables) --> Lambda function with multiple expression   
examples shown below   
'''  
n1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]  
  
  
def even(num):  
 if num % 2 == 0:  
 return True  
  
  
even\_n1 = list(filter(even, n1))  
print("EVEN N1: ", even\_n1)  
  
odd\_n1 = list(filter(lambda x: x % 2 != 0, n1))  
print("ODD N1 : ", odd\_n1)  
  
even\_gt\_n1 = list(filter(lambda x: x % 2 == 0 and x > 5, n1))  
print("Even & GT 5 :", even\_gt\_n1)