

## ▼ Learn Python in 2 hr

There are 16 programs to explain the various concepts in python programming such as:

- Syntex,
- Loop,
- if-else,
- Data Structures,
- Strings,
- File Handaling,
- Exception Handaling,
- Random Numbers,
- Command Line Argument
- Use of Libraries

### Self learning resource

Tutorial on Python (Byte of Python) [Click Here](#) Apni kaksha Python in One shot in 1.5 hours

## 1 Hello World

**Learning:** How to print and run python program

```
print ("Hello World")
print("hello thapar")
```

```
Hello World
hello thapar
```

**Assingment 1.1:** WAP to print your name three times

## ▼ 2 Add numbers and Concatinate strings

**Learning:** How to declare variable, add, concatenate and print the result.

### 2.1 Add two numbers

```
a = 10          #taken as integers only no input function
b = 220
c = a + b       # Add two numbers
print (a, " + ", b, " --> ", c)

10 + 220 --> 230
```

### ▼ 2.2 Concatinate two strings

```
a = "Bhagat"
b = " Singh"
c = a + b       # Concatinate two strings
print (a, " + ", b, " --> ", c)

Bhagat + Singh --> Bhagat Singh
```

### ▼ 2.3 Concatinate string with number

```
a = "Bhagat"
b = 100
c = a + str(b)      # Concatinate string with number
print (a, " + ", b, " --> ", c)
```

```
Bhagat + 100 --> Bhagat100
```

**Assingment 2.1:** WAP to add three numbers and print the result.

**Assingment 2.2:** WAP to concatenate three strings and print the result.

---

## ▼ 3 Input from user

**Learning:** How to take input from user

---

### 3.1 Input two strings from user and concatinate them

---

```
a = input("Enter First String: ")
b = input("Enter Second String: ")
c = a + b      # concatinate two strings
print (a, " + ", b, " --> ", c)
```

# Run the program with (1) Two strings and (2) Two numbers

```
Enter First String: First
Enter Second String: Second
First + Second --> FirstSecond
```

---

### ▼ 3.2 Input two numbers from user and add them

---

```
a = int(input("Enter First No: "))
b = int(input("Enter Second No: "))
c = a + b
print (a, " + ", b, " --> ", c)
```

```
Enter First No: 34
Enter Second No: 32
34 + 32 --> 66
```

---

## ▼ 4 Loop

**Learning:** Learn various loops.

---

### 4.1 While Loop

```
i=1
while i <= 10:
    print (i)
    i = i+1
```

```
1
2
3
4
5
6
7
8
9
10
```

---

### ▼ 4.2 Range Function

---

```
print ("range(10)      --> ", list(range(10)))
print ("range(10,20)   --> ", list(range(10,20)))
print ("range(0,20,2)  --> ", list(range(2,20,2)))
print ("range(-10,-20,2) --> ", list(range(-10,-20,2)))
print ("range(-10,-20,-2)--> ", list(range(-10,-20,-2)))

range(10)      --> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
range(10,20)   --> [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
range(0,20,2)  --> [2, 4, 6, 8, 10, 12, 14, 16, 18]
range(-10,-20,2) --> []
range(-10,-20,-2)--> [-10, -12, -14, -16, -18]
```

---

## ▼ 4.3 For loop

---

### 4.3.1 For loop - Version 1

```
for i in range(0,13)://table of 3
    print (3*i)

0
3
6
9
12
15
18
21
24
27
30
33
36
```

---

### ▼ 4.3.2 For loop - Version 2

---

```
for i in range(0,20,2)://increment of 2
    print (i)

0
2
4
6
8
10
12
14
16
18
```

---

### ▼ 4.3.3 For loop - Version 3

---

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

0
-1
-2
-3
-4
-5
-6
-7
-8
-9
```

---

## ▼ 4.4 Print table of 5

---

```
a=int(input("enter number"))
#int mei convert krna tha
for i in range(1,11):#agar list(range likhte to kya hota)
    print (a," * ", i , " = ", i * a)
```

```

enter number2
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20

```

## ▼ 4.5 Sum all numbers from 1 to 10

### 4.5.1 Version 1

```

s=0
for i in range(1,11):
    s=s+i
print ("Sum is --> ",s)

Sum is --> 55

```

### ▼ 4.5.2 Version 2

```

print ("Sum is --> ", sum(range(1,11)))

Sum is --> 55

```

**Assingment 4.1:** WAP to print the table of 7, 9.

**Assingment 4.2:** WAP to print the table of n and n is given by user.

**Assingment 4.3:** WAP to add all the numbers from 1 to n and n is given by user.

## ▼ 5 If-Else - Conditional Checking

**Learning:** if-else Condition

### 5.1 Input two numbers from user and compare them

```

a = int(input("Enter First No: "))
b = int(input("Enter Second No: "))
if a > b:
    print (a," > ",b)
else:
    print (a," < ",b)

Enter First No: 49
Enter Second No: 54
49 < 54

```

### ▼ 5.2 Check weather a number is odd or even

```

n = int(input("Enter a No: "))
if n % 2 == 0:
    print (n," is even")
else:
    print (n," is odd")

Enter a No: 54
54 is even

```

### ▼ 5.3 Check weather a number is prime or not

```

n = int(input("Enter a No: "))
f=0
for i in range(2, n//2 + 1):
    if n % i == 0:
        f=1
        break

if f==0:
    print ("Prime")
else:
    print ("Not Prime")

Enter a No: 56
Not Prime

```

## ▼ 5.4 Conditional Checking - Compare strings

```

a = input("Enter First String : ")
b = input("Enter Second String: ")

if a == b:
    print ("a == b")
elif a > b:
    print ("a > b")
else:
    print ("a < b")

Enter First String : a
Enter Second String: b
a < b

```

**Assingment 5.1:** WAP to find max among three numbers and input from user. [Try max() function]

**Assingment 5.2:** WAP to add all numbers divisible by 7 and 9 from 1 to n and n is given by the user.

**Assingment 5.3:** WAP to add all prime numbers from 1 to n and n is given by the user.

## ▼ 6 Functions

**Learning:** How to declare and call function

### 6.1 Add two numbers

```

def Add(a,b):
    c=a+b
    return c

print ("Add(10,20) -->", Add(10,20))
print ("Add(20,50) -->", Add(20,50))
print ("Add(80,200) -->", Add(80,200))

Add(10,20) --> 30
Add(20,50) --> 70
Add(80,200) --> 280

```

### ▼ 6.2 Prime number

```

#for n in range(2,101)
def IsPrime(n):
    for i in range(2, 101):
        if n%i==0:
            print(n)
            return 0
    return 1
for n in range(2,101)

```

```
File "<ipython-input-14-af8de3b36989>", line 1
    for n in range(2,101)
    ^
```

## ▼ 6.3 Add 1 to n

```
def AddN(n):
    s= sum(range(n+1))
    return s

print ("AddN(10)  --> ", AddN(10))
print ("AddN(20)  --> ", AddN(20))
print ("AddN(50)  --> ", AddN(50))
print ("AddN(200) --> ", AddN(200))

AddN(10)  -->  55
AddN(20)  -->  210
AddN(50)  -->  1275
AddN(200) -->  20100
```

**Assingment 6.1:** WAP using function that add all odd numbers from 1 to n, n is given by the user.

**Assingment 6.2:** WAP using function that add all prime numbers from 1 to n, n given by the user.

## ▼ 7 Math library

**Learning:** Use math library

```
import math as m
print ("exp(-200)  --> ", m.exp(-200)) # Exponential function
print ("log(100,2) --> ", m.log(100,2)) # Log
print ("log(100,10) --> ", m.log(100,10))# Log
print ("log10(100) --> ", m.log10(100)) # Log 10
print ("m.cos(30)  --> ", m.cos(30))   # cos
print ("m.sin(30)  --> ", m.sin(30))   # sin
print ("m.tan(30)  --> ", m.tan(30))   # tan
print ("m.sqrt(324) --> ", m.sqrt(324))
print ("m.ceil(89.9) --> ", m.ceil(89.9))
print ("m.floor(89.9)--> ", m.floor(89.9))

exp(-200)  -->  1.3838965267367376e-87
log(100,2)  -->  6.643856189774725
log(100,10) -->  2.0
log10(100)  -->  2.0
m.cos(30)   -->  0.15425144988758405
m.sin(30)   -->  -0.9880316240928618
m.tan(30)   -->  -6.405331196646276
m.sqrt(324) -->  18.0
m.ceil(89.9) -->  90
m.floor(89.9)-->  89
```

## ▼ 8 Strings

**Learning:** How to handle string

### 8.1 Indexing in string

```
var = 'Hello World!'
print ("var      --> ", var)
print ("var[0]   --> ", var[0])
print ("var[1:5] --> ", var[1:5])
print ("var[:5]  --> ", var[:5])

var      -->  Hello World!
var[0]   -->  H
var[1:5] -->  ello
var[:5]  -->  Hello W
```

### ▼ 8.2 String length, upper, lower

```
var = 'Hello World!'
print ("String --> ", var)
print ("Length --> : ", len(var))
print ("Upper --> : ", var.upper())
print ("Lower --> : ", var.lower())

String --> Hello World!
Length --> : 12
Upper --> : HELLO WORLD!
Lower --> : hello world!
```

### ▼ 8.3 String formatting

```
name=input("Enter your name: ")
age=int(input("Enter your age : "))
price=float(input("Enter the book price: "))
s="\nYour name is %s, age is %d and book price is %f" %(name.upper(),age,price)
print (s)
```

```
Enter your name: Rohan
Enter your age : 21
Enter the book price: 45.34
```

```
Your name is ROHAN, age is 21 and book price is 45.340000
```

### ▼ 8.4 String in Triple Quotes

```
para_str = """This is a long string that is made up of
several lines and non-printable characters such as
TAB ( \t ) and they will show up that way when displayed.
NEWLINES within the string, whether explicitly given like
this within the brackets [ \n ], or just a NEWLINE within
the variable assignment will also show up.
"""
print (para_str)
```

```
This is a long string that is made up of
several lines and non-printable characters such as
TAB ( ) and they will show up that way when displayed.
NEWLINES within the string, whether explicitly given like
this within the brackets [
], or just a NEWLINE within
the variable assignment will also show up.
```

### ▼ 8.5 String strip

```
var =" Indian Army "
```

```
print("String --> ", var)
print("Length --> ", len(var))
print("var strip --> ", var.strip())
print("Length of var after strip --> ", len(var.strip()))

String --> Indian Army
Length --> 18
var strip --> Indian Army
Length of var after strip --> 13
```

### ▼ 8.6 String split

```
var =" Indian, Army "
```

```
print("String --> ", var)
print("Length --> ", len(var))
print("var split --> ", var.split())
print("var split --> ", var.split(' '))
print("var split --> ", var.split(','))

# Strip + Split
```

```
print("var split --> ", var.strip().split(','))

String --> Indian, Army
Length --> 19
var split --> ['Indian,', 'Army']
var split --> ['', 'Indian,', '', '', 'Army', '', '', '', '']
var split --> [' Indian', ' Army ']
var split --> ['Indian', ' Army']
```

## ▼ 8.7 Count in string

```
var=" Indian Army "
print ("String --> ", var)
print ("Count of ' ' --> ", var.count(' '))
print ("Count of 'a' --> ", var.count('a'))
print ("Count of 'n' --> ", var.count('n'))

String --> Indian Army
Count of ' ' --> 6
Count of 'a' --> 1
Count of 'n' --> 1
```

## ▼ 8.8 Reverse a String

```
var="Indian Army"
print ("String --> ", var)
print ("var[::-1] --> ", var[::-1])
print ("var[::-2] --> ", var[::-2])
print ("var[::-1] --> ", var[::-1])
print ("var[::-2] --> ", var[::-2])

var=var[::-1]
print ("var after reverse --> ", var)

String --> Indian Army
var[::-1] --> Indian Army
var[::-2] --> Ida ry
var[::-1] --> ymrA naidnI
var[::-2] --> yr adI
var after reverse --> ymrA naidnI
```

## ▼ 8.9 Palindrome

```
s1="Indian Army"
s2="malayalam"
s3="madam"
s4="teacher"
print ("s1 --> ", s1==s1[::-1])
print ("s2 --> ", s2==s2[::-1])
print ("s3 --> ", s3==s3[::-1])
print ("s4 --> ", s4==s4[::-1])

s1 --> False
s2 --> True
s3 --> True
s4 --> False
```

## ▼ 9 Random Numbers/String

**Learning:** Generate Random Numbers/String

### 9.1 Generate random number between 0 and 1

```
import random as r
print (r.random())
print (r.random())
print (round(r.random(),4))
```



```
0.7512486175763348
0.5015832763034391
0.7335
```

## ▼ 9.2 Generate random integer number

```
import random as r
print (r.randint(1, 100))#every time different random number
print (r.randint(1, 100))
print (r.randint(-10, 10))
print (r.randint(-10, 10))
```

```
79
70
-10
-4
```

## ▼ 9.3 Generate random real number

```
import random as r
print (r.uniform(1, 100))#for decimal places printing
print (r.uniform(1, 100))
print (r.uniform (-10, 10))
print (r.uniform (-10, 10))
print (round(r.uniform (-10, 10),2))
```

```
97.39127815370512
99.35262788984427
1.5886268848088747
-4.72371568384482
3.54
```

## ▼ 9.4 Select sample from a list of elements

```
import random as r

A=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]#list se four random number select krna hai

print (r.sample(A, 4))#a mei se 4 select krne hai
print (r.sample(A, 2))#a mei se 2 select krne hai
print (r.sample(range(0,100), 2))
print (r.sample(range(-100,100), 5))
#print(r.sample(A,40)#error)
#sample and randint : sample mei ek particular unique set se select krte hai

[3, 4, 8, 5]
[2, 4]
[20, 33]
[-20, 52, -10, 41, -15]
```

## ▼ 9.5 Generate random string

```
import string as s#password forgot#string library
import random as r
print ("String      --> ",s.ascii_letters)#prints small and capital letter

passwd=r.sample(s.ascii_letters, 6)#ascii mei se 6 char select kro
print ("Selected Char --> ",passwd)#those selected chars are passwd

passwd1="".join(passwd)#joins char to form string#char concatenation
print ("passwd1      --> ",passwd1)

passwd2="+".join(passwd)#+ se join kro
print ("passwd2      --> ",passwd2)

passwd3="*".join(passwd)
print ("passwd3      --> ",passwd3)
```

```
String      -->  abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
Selected Char -->  ['b', 'n', 'T', 'z', 'f', 'Y']
```

```

passwd1    --> bnTzfY
passwd2    --> b+n+T+z+f+Y
passwd3    --> b*n*T*z*f*Y

```

## ▼ 9.6 Generate random digits

```

import string as s
import random as r
print ("Digits --> ",s.digits)#digits 0-9 hote hai na

otp=r.sample(s.digits, 5)#0-9 digits mei se 5 digits select kro as otp
print ("Selected num1 --> ",otp)#different char and digits selected everytime
otp="".join(otp)#join digits with space
print ("otp1          --> ",otp)

otp=r.sample(s.digits, 5)
print ("Selected num2 --> ",otp)
otp="".join(otp)
print ("otp2          --> ",otp)

otp=r.sample(s.digits, 5)
print ("Selected num2 --> ",otp)
otp="".join(otp)
print ("otp3          --> ",otp)

Digits --> 0123456789
Selected num1 --> ['1', '7', '8', '2', '9']
otp1          --> 17829
Selected num2 --> ['1', '3', '9', '5', '2']
otp2          --> 13952
Selected num2 --> ['0', '2', '1', '6', '3']
otp3          --> 02163

```

## ▼ 9.7 Generate random string + digits

```

import string as s
import random as r
print ("String + Digits --> ",s.ascii_letters + s.digits)#+ use kiya hai to include s.digits also#concatenation

mixPasswd=r.sample(s.ascii_letters + s.digits, 5)#select any 5
print ("\nSelected Str1 --> ",mixPasswd)#display selection
mixPasswd="".join(mixPasswd)
print ("mixPasswd1    --> ",mixPasswd)

mixPasswd=r.sample(s.ascii_letters + s.digits, 6)#6 tak koi digit aaya hi nhi
print ("\nSelected Str2 --> ",mixPasswd)
mixPasswd="".join(mixPasswd)
print ("mixPasswd2    --> ",mixPasswd)

splChar="#@!~%^&*()_+=[{}]"
mixPasswd=r.sample(splChar + s.ascii_letters + s.digits, 8)#spl char yaad rkho
print ("\nSelected Str3 --> ",mixPasswd)
mixPasswd="".join(mixPasswd)
print ("mixPasswd3    --> ",mixPasswd)

String + Digits --> abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

Selected Str1 --> ['e', 'K', 't', 'o', 'R']
mixPasswd1    --> eKtoR

Selected Str2 --> ['D', 'u', 'J', '5', 'T', 'M']
mixPasswd2    --> DuJ5TM

Selected Str3 --> ['p', 'W', '{', 'b', '_', '+', '6', '#']
mixPasswd3    --> pW{b_+6#

```

## ▼ 10 Exception Handling##runtime errors

**Learning:** How to handle exceptions

### 10.1 Error Generation

```

for i in range(-5,6):
    print ("100/",i," --> ", 100/i)#zero se divide kroge to error ayega..illegal termination
#suppose your software converts file to upper case..but fails due to file not found,diff format,file size
#cloud recording if connection lost then programming reqd

100/ -5 --> -20.0
100/ -4 --> -25.0
100/ -3 --> -33.333333333333336
100/ -2 --> -50.0
100/ -1 --> -100.0

-----
ZeroDivisionError                                Traceback (most recent call last)
<ipython-input-21-5eb017879ab5> in <cell line: 1>()
      1 for i in range(-5,6):
----> 2         print ("100/",i," --> ", 100/i)

ZeroDivisionError: division by zero

```

SEARCH STACK OVERFLOW

## ▼ 10.2 Exception handling for division by zero

```

for i in range(-5,6):
    try:#duty of coder
        print ("100/",i," --> ", 100/i)
    except:
        print ("error")#jese hi value 0 hui to except ne exception ko handle krliya#I can beep also

100/ -5 --> -20.0
100/ -4 --> -25.0
100/ -3 --> -33.333333333333336
100/ -2 --> -50.0
100/ -1 --> -100.0
error
100/ 1 --> 100.0
100/ 2 --> 50.0
100/ 3 --> 33.333333333333336
100/ 4 --> 25.0
100/ 5 --> 20.0

```

## ▼ 10.3 Exception handling for array out of index

```

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

for i in range(8):#5 elements se zyada print krne ki koshish(8)#array out of index
    try:
        print (i," --> ",L[i])
    except:
        print ("error")

0 --> 1
1 --> 2
2 --> 3
3 --> 4
4 --> 5
error
error
error

```

## ▼ 10.4 Exception handling for file not found or large size or diff format

```

fileName=input("Enter File Name: ")#convert text file to uppercase
fp=open(fileName) # Open the file in reading mode#open function crash as file not found
fp.close()
print ("Done")

```

```
Enter File Name: two.txt
```

```
FileNotFoundError
```

```
Traceback (most recent call last)
```

## 10.5 Exception handling for file not found

```
3 fp.close()

fileName=input("Enter File Name: ")
try:
    fp=open(fileName) # Open the file in reading mode
    fp.close()
except:
    print ("Error !! \"%s\" File Not Found"%(fileName))#%s is the name of the file formatted string

print ("Done")

Enter File Name: test.csv
Error !! "test.csv" File Not Found
Done
```

## 11 Data Structure 1 - List

**Learning:** How to use list, add, delete and search in the list.

**Note:** Read more about list and try yourself

### 11.1 List Declaration

```
L = ["Pratham", 'Sharma', 3.14, 3 ]
print ("Original List: ", L)
print ("Number of elements in list: ", len(L))

Original List: ['Pratham', 'Sharma', 3.14, 3]
Number of elements in list: 4
```

### 11.2 List Iteration

```
L = ["Pratham", 'Sharma', 3.14, 3 ]
print ("Original List: ", L)
i=0
while i < len(L):
    print (L[i])
    i+=1

Original List: ['Pratham', 'Sharma', 3.14, 3]
Pratham
Sharma
3.14
3
```

### 11.3 List Iteration using for loop

```
L = ["Pratham", 'Sharma', 3.14, 3 ]
print ("Original List: ", L)
for i in range(0, len(L)):
    print (L[i])

Original List: ['Pratham', 'Sharma', 3.14, 3]
Pratham
Sharma
3.14
3
```

### 11.4 List Iteration using for loop

```
L = ["Pratham", 'Sharma', 3.14, 3 ]
print ("Original List --> ", L)
for s in L:
    print (s)

Original List --> ['Pratham', 'Sharma', 3.14, 3]
Pratham
Sharma
3.14
3
```

## ▼ 11.5 Adding and deleting from list

```
L = ["Pratham", 'Sharma', 3.14, 3 ]
print ("Original List --> ", L)

L.append("Rahul")
print ("List After Adding --> ", L)

del L[1]#delete the element at index 1
print ("List After Deleting --> ", L)

Original List --> ['Pratham', 'Sharma', 3.14, 3]
List After Adding --> ['Pratham', 'Sharma', 3.14, 3, 'Rahul']
List After Deleting --> ['Pratham', 3.14, 3, 'Rahul']
```

## ▼ 11.6 Sum/Average of List

```
L=[3, 6, 9, 12, 5, 3, 2]
print ("Original List --> ", L)

print ("Sum --> ", sum(L))#poori array k sare elements ka sum
print ("Average --> ", sum(L)/len(L))#
print ("Average --> ", sum(L)//len(L))

print ("L * 3 --> ", L * 3) # Every element get tripled
print ("L + L --> ", L + L) # Every element get doubled

Original List --> [3, 6, 9, 12, 5, 3, 2]
Sum --> 40
Average --> 5.714285714285714
Average --> 5
L * 3 --> [3, 6, 9, 12, 5, 3, 2, 3, 6, 9, 12, 5, 3, 2, 3, 6, 9, 12, 5, 3, 2]
L + L --> [3, 6, 9, 12, 5, 3, 2, 3, 6, 9, 12, 5, 3, 2]
```

## ▼ 11.7 Min/Max/Sort the list

```
L=[3, 6, 9, 12, 5, 3, 2]
print ("Original List --> ", L)

print ("max --> ", max(L))#functions
print ("min --> ", min(L))

print ("\nBefore Sort --> ", L)
L.sort()#list mei change krdega permanent by default inc order

print ("After Sort (Asending) --> ", L)

L.sort(reverse=True)#to reverse the order
print ("After Sort (Desending) --> ", L)

Original List --> [3, 6, 9, 12, 5, 3, 2]
max --> 12
min --> 2

Before Sort --> [3, 6, 9, 12, 5, 3, 2]
After Sort (Asending) --> [2, 3, 3, 5, 6, 9, 12]
After Sort (Desending) --> [12, 9, 6, 5, 3, 3, 2]
```

## ▼ 11.8 Merge lists and select elements

```

L1 = [3, 6, 9]
L2 = [12, 5, 3, 2]
L3 = L1 + L2#concatenation #operator overloading
print ("L1 --> ",L1)
print ("L2 --> ",L2)
print ("L3 --> ",L3)

print ("\nL3[2:] --> ",L3[2:])#second element se start krega
print ("L3[2:5] --> ",L3[2:5])#2 to 5
print ("L3[: -1] --> ",L3[: -1])
print ("L3[::2] --> ",L3[::2])

L1 --> [3, 6, 9]
L2 --> [12, 5, 3, 2]
L3 --> [3, 6, 9, 12, 5, 3, 2]

L3[2:] --> [9, 12, 5, 3, 2]
L3[2:5] --> [9, 12, 5]
L3[: -1] --> [3, 6, 9, 12, 5, 3]
L3[::2] --> [3, 9, 5, 2]

```

## ▼ 11.9 Multiply all elements of the list by a constant

```

L = [12, 5, 3, 2, 7]#important
print ("Original List --> ", L)

newL = [ i * 5 for i in L ]#new list#list mei loop
print ("After Multiply with constant --> ", newL)

Original List --> [12, 5, 3, 2, 7]
After Multiply with constant --> [60, 25, 15, 10, 35]

```

## ▼ 11.10 Searching in the list

```

L=[3, 6, 9, 12, 5, 3, 2]
print ("Original List --> ", 6 in L)
print ("Original List --> ", 10 in L)
print ("Original List --> ", 12 in L)

if (6 in L) == True:
    print ("Present")
else:
    print ("Not Present")

if 10 in L == False:#equality least precedance#bracket
    print ("Not Present")
else:
    print ("Present")

Original List --> True
Original List --> False
Original List --> True
Present
Present

```

## ▼ 12 Data Structure 2 - Dictionary

**Learning:** How to use Dictionary, add, delete, search in Dictionary

**Note:** Read more about Dictionary and try yourself

### 12.1 Declare Dictionary

```

CGPA={1:8.9, 2:5.6, 4:6.7, 7:9.1, 8:5.3}
print ("Dictionary --> ", CGPA)
print ("Num of elements --> ", len(CGPA))

print ("CGPA of 1 --> ", CGPA[1])
print ("CGPA of 4 --> ", CGPA[4])
print ("CGPA of 7 --> ", CGPA[7])
print ("CGPA of 3 --> ", CGPA[3])#3 exist hi nhi krta #exception handling

```

```

Dictionary --> {1: 8.9, 2: 5.6, 4: 6.7, 7: 9.1, 8: 5.3}
Num of elements --> 5
CGPA of 1 --> 8.9
CGPA of 4 --> 6.7
CGPA of 7 --> 9.1
-----
KeyError                                Traceback (most recent call last)
<ipython-input-54-598d6ecb7ab2> in <module>()
      6 print ("CGPA of 4 --> ", CGPA[4])
      7 print ("CGPA of 7 --> ", CGPA[7])
----> 8 print ("CGPA of 3 --> ", CGPA[3])

KeyError: 3

```

## ▼ 12.2 Traverse dictionary

```

CGPA={1:8.9, 2:5.6, 4:6.7, 7:9.1, 8:5.3}#simple
for k in CGPA:
    print ("CGPA of ", k, " --> ", CGPA[k])

CGPA of 1 --> 8.9
CGPA of 2 --> 5.6
CGPA of 4 --> 6.7
CGPA of 7 --> 9.1
CGPA of 8 --> 5.3

```

## ▼ 12.3 Getting Keys and Values

```

CGPA={1:8.9, 2:5.6, 4:6.7, 7:9.1, 8:5.3}
print ("Dictionary --> ", CGPA)
print ("Keys --> ", list(CGPA.keys()))#list mei sari keys display kro
print ("Values --> ", list(CGPA.values()))

Dictionary --> {1: 8.9, 2: 5.6, 4: 6.7, 7: 9.1, 8: 5.3}
Keys --> [1, 2, 4, 7, 8]
Values --> [8.9, 5.6, 6.7, 9.1, 5.3]

```

## ▼ 12.4 Updating, Adding and Deleting from Dictionary

```

CGPA={1:8.9,2:5.6,4:6.7,7:9.1,8:5.3}
print ("Original Dictionary --> ", CGPA)

CGPA[4] = 9.2#directly add
print ("After Updating (4) --> ", CGPA)

CGPA[3] = 8.6
print ("After Adding (3) --> ", CGPA)

del CGPA[1]#deleted
print ("After Deleting (1) --> ", CGPA)

CGPA.clear()#whole dictionary empty {}
print ("After Clear --> ", CGPA)

del CGPA#check
print ("After Delete --> ", CGPA)

Original Dictionary --> {1: 8.9, 2: 5.6, 4: 6.7, 7: 9.1, 8: 5.3}
After Updating (4) --> {1: 8.9, 2: 5.6, 4: 9.2, 7: 9.1, 8: 5.3}
After Adding (3) --> {1: 8.9, 2: 5.6, 4: 9.2, 7: 9.1, 8: 5.3, 3: 8.6}
After Deleting (1) --> {2: 5.6, 4: 9.2, 7: 9.1, 8: 5.3, 3: 8.6}
After Clear --> {}
-----
NameError                                Traceback (most recent call last)
<ipython-input-57-81104b3723b2> in <module>()
     15
     16 del CGPA
----> 17 print ("After Delete --> ", CGPA)

NameError: name 'CGPA' is not defined

```

SEARCH STACK OVERFLOW

## ▼ 12.5 Checking for Key in Dictionary like hashmap

```
CGPA={1:8.9, 2:5.6, 4:6.7, 7:9.1, 8:5.3}
print ("Original Dictionary --> ", CGPA)
print ("Is Key 2 Present --> ", 2 in CGPA)
print ("Is Key 9 Present --> ", 9 in CGPA)

Original Dictionary --> {1: 8.9, 2: 5.6, 4: 6.7, 7: 9.1, 8: 5.3}
Is Key 2 Present --> True
Is Key 9 Present --> False
```

## ▼ 12.6 More example1

```
HomeTown={"Prashant":"Delhi", "Govind":"Gwalior", "Anil":"Morena", "Pankaj":"Agra"}
print ("Original Dictionary --> ", HomeTown)
print ("Home Town of Prashant is --> ", HomeTown["Prashant"])
print ("Home Town of Govind is --> ", HomeTown["Govind"])
print ("Home Town of Anil is --> ", HomeTown["Anil"])
print ("Home Town of Pankaj is --> ", HomeTown["Pankaj"])

Original Dictionary --> {'Prashant': 'Delhi', 'Govind': 'Gwalior', 'Anil': 'Morena', 'Pankaj': 'Agra'}
Home Town of Prashant is --> Delhi
Home Town of Govind is --> Gwalior
Home Town of Anil is --> Morena
Home Town of Pankaj is --> Agra
```

## ▼ 12.7 More example2

```
HomeTown={"Prashant":"Delhi", "Govind":"Gwalior", "Anil":"Morena", "Pankaj":"Agra"}
print ("Original Dictionary --> ", HomeTown)

for d in HomeTown:
    print ("Home Town of ", d, " is --> ", HomeTown[d])

Original Dictionary --> {'Prashant': 'Delhi', 'Govind': 'Gwalior', 'Anil': 'Morena', 'Pankaj': 'Agra'}
Home Town of Prashant is --> Delhi
Home Town of Govind is --> Gwalior
Home Town of Anil is --> Morena
Home Town of Pankaj is --> Agra
```

## ▼ 13 Data Structure 3 - Tuple

**Learning:** How to use Tuple, add, delete, search in Tuple

**Note:** Read more about Tuple and try yourself

### 13.1 Declare Tuple

```
# Method 1
T = ("Pratham", 'Sharma', 3.14, 3)

print ("T -->", T)
print ("Num of elements -->", len(T))
print ("Type of Object -->", type(T))

T --> ('Pratham', 'Sharma', 3.14, 3)
Num of elements --> 4
Type of Object --> <class 'tuple'>

# Method 2
T = tuple(["Pratham", 'Sharma', 3.14, 3]) # Convert list to tuple
#T = tuple(("Pratham", 'Sharma', 3.14, 3)) # Also Works

print ("T -->", T)
print ("Num of elements -->", len(T))
print ("Type of Object -->", type(T))

T --> ('Pratham', 'Sharma', 3.14, 3)
Num of elements --> 4
Type of Object --> <class 'tuple'>
```



## ▼ 13.2 Tuple Iteration

```
T = ("Pratham", 'Sharma', 3.14, 3)
print ("T -->", T)

i = 0
while i < len(T):
    print (T[i])
    i += 1

T --> ('Pratham', 'Sharma', 3.14, 3)
Pratham
Sharma
3.14
3
```

## ▼ 13.3 Tuple iteration using for loop

```
T = ("Pratham", 'Sharma', 3.14, 3)
print ("T -->", T)

for i in range(0, len(T)):
    print (T[i])

T --> ('Pratham', 'Sharma', 3.14, 3)
Pratham
Sharma
3.14
3
```

## ▼ 13.4 Tuple iteration using for loop

```
T = ("Pratham", 'Sharma', 3.14, 3)
print ("T -->", T)


for s in T:
    print (s)

T --> ('Pratham', 'Sharma', 3.14, 3)
Pratham
Sharma
3.14
3
```

## ▼ 13.5 Accessing/Selecting in Tuple

```
# Example 1:
T = (3, 6, 9, 12, 5, 3, 2)
print ("T      -->", T)

print ("T[1]  -->", T[1])
print ("T[2]  -->", T[2])
print ("T[-1] -->", T[-1])
print ("T[-2] -->", T[-2])

 T      --> (3, 6, 9, 12, 5, 3, 2)
T[1]    --> 6
T[2]    --> 9
T[-1]   --> 2
T[-2]   --> 3
```

```
# Example 2:
T = (3, 6, 9, 12, 5, 3, 2)
print ("T      -->", T)

print ("T[1:3] -->", T[1:3])
print ("T[2:]  -->", T[2:])
print ("T[2:5] -->", T[2:5])
print ("T[:2]  -->", T[:2])
```

```

print ("T[:-1] -->", T[:-1])
print ("T[-4:-1] -->", T[-4:-1])

T      --> (3, 6, 9, 12, 5, 3, 2)
T[1:3] --> (6, 9)
T[2:]  --> (9, 12, 5, 3, 2)
T[2:5] --> (9, 12, 5)
T[:2]  --> (3, 6)
T[:-1] --> (3, 6, 9, 12, 5, 3)
T[-4:-1] --> (12, 5, 3)

```

### ▼ 13.6 Sum/Average of Tuple

```

T = (3, 6, 9, 12, 5, 3, 2)
print ("T      -->", T)
print ("Sum     -->", sum(T))
print ("Average -->", sum(T)/len(T))
print ("Average -->", sum(T)//len(T))

T      --> (3, 6, 9, 12, 5, 3, 2)
Sum     --> 40
Average --> 5.714285714285714
Average --> 5

```

### ▼ 13.7 Min/Max in Tuple

```

# Example 1
T = (3, 6, 9, 12, 5, 3, 2)          # Integer Tuple
print ("T      -->", T)
print ("Max -->", max(T))
print ("Min -->", min(T))

```

```

T      --> (3, 6, 9, 12, 5, 3, 2)
Max --> 12
Min --> 2

```

```

# Example 2
T = ("Ram", "Shyam", "Human", "Ant")  # String Tuple
print ("T      -->", T)
print ("Max -->", max(T))
print ("Min -->", min(T))

```

```

T      --> ('Ram', 'Shyam', 'Human', 'Ant')
Max --> Shyam
Min --> Ant

```

### ▼ 13.8 Merging Tuples

```

T1 = (3, 6, 9)
T2 = (12, 5, 3, 2)

print ("T1 -->", T1)
print ("T2 -->", T2)

T3 = T1 + T2
print ("T3 -->", T3)

T4 = T1 + T2 + T1 + T2
print ("T4 -->", T4)

T1 --> (3, 6, 9)
T2 --> (12, 5, 3, 2)
T3 --> (3, 6, 9, 12, 5, 3, 2)
T4 --> (3, 6, 9, 12, 5, 3, 2, 3, 6, 9, 12, 5, 3, 2)

```

### ▼ 13.9 Merging part of Tuples

```

T1 = (3, 6, 9)
T2 = (12, 5, 3, 2)

print ("T1 -->", T1)

```

```

print ("T2 -->", T2)

T3 = T1[1:2] + T2[1:3]
print ("T3 -->", T3)

T4 = T1[:-2] + T2[:-3]
print ("T4 -->", T4)

T1 --> (3, 6, 9)
T2 --> (12, 5, 3, 2)
T3 --> (6, 5, 3)
T4 --> (3, 12)

```

### ▼ 13.10 Searching in the tuple

```

T = (3, 6, 9, 12, 5, 3, 2)
print ("T -->", T)

print ("6 in T -->", 6 in T)
print ("10 in T -->", 10 in T)
print ("12 in T -->", 12 in T)

T --> (3, 6, 9, 12, 5, 3, 2)
6 in T --> True
10 in T --> False
12 in T --> True

```

### ▼ 13.11 Adding element to Tuple (Error)

```

T = ("Pratham", 'Sharma', 3.14, 3)
print ("T -->", T)

T[2] = 900 # Error; 'tuple' object does not support item assignment
print ("T -->", T)

```

#Tuples are unchangeable. We cannot add items to it.

```

T --> ('Pratham', 'Sharma', 3.14, 3)
-----
TypeError                                Traceback (most recent call last)
<ipython-input-74-fb1d27ae8658> in <module>()
      2 print ("T -->", T)
      3
----> 4 T[2] = 900 # Error; 'tuple' object does not support item
assignment
      5 print ("T -->", T)
      6

TypeError: 'tuple' object does not support item assignment

```

SEARCH STACK OVERFLOW

### ▼ 13.12 Adding element to Tuple - (Jugaad)

```

T = ("Pratham", 'Sharma', 3.14, 3)
print ("T -->", T)

T1 = list(T)
T1.append(9.8)
T = tuple(T1)

print ("After Add -->", T)

T --> ('Pratham', 'Sharma', 3.14, 3)
After Add --> ('Pratham', 'Sharma', 3.14, 3, 9.8)

```

### ▼ 13.13 Inserting element in Tuple - (Jugaad)

```

T = ("Pratham", 'Sharma', 3.14, 3)
print ("T -->", T)

```

```
T1 = list(T)
T1.insert(2, "Rahul")
T = tuple(T1)

print ("After Insert -->", T)

T
--> ('Pratham', 'Sharma', 3.14, 3)
After Insert --> ('Pratham', 'Sharma', 'Rahul', 3.14, 3)
```

### ▼ 13.14 Deleting from Tuple (Error)

```
T = ("Pratham", 'Sharma', 3.14, 3)
print ("T      -->", T)

del T[1]

print ("After Delete -->", T)

T
--> ('Pratham', 'Sharma', 3.14, 3)
-----
TypeError                                Traceback (most recent call last)
<ipython-input-77-0baff4d8a5c8> in <module>()
      2 print ("T      -->", T)
      3
----> 4 del T[1]
      5
      6 print ("After Delete -->", T)

TypeError: 'tuple' object doesn't support item deletion
```

SEARCH STACK OVERFLOW

### ▼ 13.15 Deleting from Tuple - (Jugaad)

```
T = ("Pratham", 'Sharma', 3.14, 3)
print ("T      -->", T)

T1 = list(T)
del T1[1]
T = tuple(T1)

print ("After Delete -->", T)

T
--> ('Pratham', 'Sharma', 3.14, 3)
After Delete --> ('Pratham', 3.14, 3)
```

## ▼ 14 Data Structure 4 - Set

**Learning:** How to use Set, add, delete, search in Set

**Note:** Read more about Set and try yourself

### 14.1 Declare Set

```
s = set(['A', 'B', 'E', 'F', 'E', 'F' ])
print ("Original set      --> ", s)#unique elements printed
print ("Num of elements in set --> ", len(s))

Original set      --> {'E', 'A', 'F', 'B'}
Num of elements in set --> 4
```

### ▼ 14.2 Operations on Sets

```
a = set(['A', 'B', 'E', 'F' ])
b = set(["A", "C", "D", "E"])
print ("Original set a      --> ", a)
print ("Original set b      --> ", b)
print ("Union of a and b    --> ", a.union(b))#a ka union with b
print ("Intersection of a,b --> ", a.intersection(b))#a ki intersection with b
```

```

print ("Difference a - b --> ", a - b)
print ("Difference a - b --> ", a.difference(b))
print ("Difference b - a --> ", b - a)
print ("Difference b - a --> ", b.difference(a))
print ("Symetric Diff a - b --> ", a.symmetric_difference(b))
print ("Symetric Diff b - a --> ", b.symmetric_difference(a))

Original set a --> {'E', 'A', 'F', 'B'}
Original set b --> {'E', 'A', 'D', 'C'}
Union of a and b --> {'B', 'D', 'F', 'E', 'A', 'C'}
Intersection of a,b --> {'E', 'A'}
Difference a - b --> {'F', 'B'}
Difference a - b --> {'F', 'B'}
Difference b - a --> {'D', 'C'}
Difference b - a --> {'D', 'C'}
Symetric Diff a - b --> {'F', 'D', 'C', 'B'}
Symetric Diff b - a --> {'B', 'F', 'D', 'C'}

```

## ▼ 14.3 Add, delete, pop element from set

```

a = set(['A', 'B', 'E', 'F' ])
print ("Original set a --> ", a)
a.add("D")
print ("Set After Adding (D) --> ", a)
a.add("D")
print ("Set After Adding (D) --> ", a)
a.remove("D")
print ("Set After Deleting(D)--> ", a)
a.pop()
print ("Set After pop --> ", a)
a.pop()
print ("Set After pop --> ", a)

Original set a --> {'E', 'A', 'F', 'B'}
Set After Adding (D) --> {'B', 'D', 'F', 'E', 'A'}
Set After Adding (D) --> {'B', 'D', 'F', 'E', 'A'}
Set After Deleting(D)--> {'B', 'F', 'E', 'A'}
Set After pop --> {'F', 'E', 'A'}
Set After pop --> {'E', 'A'}

```

## ▼ 15 Command Line Argument

**Learning:** How to Take input from command line and process it

**Note:** Run the program at cmd line

### 15.1 Add two numbers given at cmd line

**Note:** To run the program at cmd line

- python Program.py 10 20

```

import sys#running in command line
print (sys.argv)
a = int(sys.argv[1])    # First Number
b = int(sys.argv[2])    # Second Number
c = a + b
print (a, " + ", b, " --> ", c)

['/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py', '-f', '/root/.local,
-----
ValueError                                Traceback (most recent call last)
<ipython-input-82-a3d67294dc12> in <module>()
      1 import sys
      2 print (sys.argv)
----> 3 a = int(sys.argv[1])    # First Number
      4 b = int(sys.argv[2])    # Second Number
      5 c = a + b

ValueError: invalid literal for int() with base 10: '-f'

```

SEARCH STACK OVERFLOW

## ▼ 15.2 Concatenate two strings given at cmd line

**Note:** To run the program at cmd line

- python Program.py FirstString SecondString

```
import sys
print (sys.argv)
s = sys.argv[1] + " " + sys.argv[2]
print (sys.argv[1], " + ", sys.argv[2], " --> ", s)
```

```
['/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py', '-f', '/root/.local/share/jupyter/runtime/kernel-2c370b3f-04bb-4872-8f9b-8559cbd8132b.json', '--', '-f /root/.local/share/jupyter/runtime']
```

## ▼ 15.3 Add all the numbers given at cmd line

**Note:** To run the program at cmd line

- python Program.py
- python Program.py 10
- python Program.py 10 20 30 40

```
import sys#best code
print (sys.argv)
sum=0
for s in sys.argv[1:]:#0th element
    sum += int(s)
```

```
print ("Sum is --> ", sum)
```

```
['/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py', '-f', '/root/.local/
```

```
ValueError                                Traceback (most recent call last)
```

```
<ipython-input-84-87063e0e9afe> in <module>()
```

```
3 sum=0
```

```
4 for s in sys.argv[1:]:
```

```
----> 5     sum += int(s)
```

```
6
```

```
7 print ("Sum is --> ", sum)
```

```
ValueError: invalid literal for int() with base 10: '-f'
```

SEARCH STACK OVERFLOW

## ▼ 16 File Handling

**Learning:** How to open the file, read the file and write in the file

### 16.1 Writing 1 to 10 in file

```
fp=open('result.txt','w') # Open the file in writing mode
for i in range(1,11):
    fp.write(str(i) + "\n") # Writing to the file line by line#new line appended
fp.close()
```

```
print ("Writing done !! \nOpen result.txt to view the content")
```

```
Writing done !!
Open result.txt to view the content
```

### ▼ 16.2 Read a file and print its content

```
fp=open('result.txt') # Open the file in reading mode(by default)
for line in fp: # print line by line
    print (line.strip())
fp.close()
```

```
1
2
3
4
```

5  
6  
7  
8  
9  
10

---

### ▼ 16.3 Read from one file, Convert it to upper case and write to other file

---

```
Readfp=open('result.txt')      # Open the file in reading mode
Writefp=open('abc.txt','w') # Open the file in writing mode
for line in Readfp:
    Writefp.write(line.upper())

Writefp.close()
Readfp.close()

print ("Writing done !! \nOpen result.txt to view the content")

Writing done !!
Open result.txt to view the content
```

