P Y T H O N

What is Python ?

- It is powerful,general-purpose, high level,object oriented programming language.
- It is developed by Guido van Rossum .
- It is released in 1991.

Features of Python ?

1. **Platform Independent:**Python is called platform independent because a Python program can be run on different kind of platform for example Window os,Linux os etc.
2. **Object Oriented:**Python supports object oriented programming structure so it is called object oriented.
3. **Flexible:**An application developed in Python can be modified as per user requirement so it is called flexible programming language.
4. **Structure Oriented:**To write program in Python there is a fixed structure so it is called structure oriented.
5. **Portable:**A Python program written in one system can be run in any other system.In simple a Python program can be transferred from one system to another.
6. **Simple:**Python is very simple and easy to learn.

Application of Python ?

- Web Application
- Software development
- Database GUI Application
- Scientific and Numeric Computing
- Business Applications
- Console Based Application

Why Use Python ?

- It is very simple and easy to learn.
- It powerful,fast and secure.
- It has very simple syntax.
- It is powerful scripting language.
- It can be run on different kind of platform for example Window,Mac,Linux etc.

First Program

- It is very simple to write and execute any Python Program.
- Python Program can be executed directly in the command line .

```
print("Hello Python")
```

Indentation in Python

- C ,C++,Java etc languages use braces to indicate blocks of code for class and function definitions or flow control.
- But Python uses indentation to indicate a block of code.
- We can use at least one space for indentation.

```
a=5
b=10
if a>b:
    print("a is greter than b")
else:
    print("b is greater than a")
```

- Python code will raise an error if you skip indentation.

```
a=5
b=10
if a>b:
#this lin will raise an error
#because no indentation
print("a is greter than b")
else:
    print("b is greater than a")
"""
**Output**
IndentationError: expected an indented block
"""
```

- You can use number of spaces for indentation but spaces must be same in the same block of code.

```
a=5
b=10
if a>b:
         print("a is greter than b")
         print("means b is less than a")
else:
         print("b is greater than a")
         print("means a is less than b")
"""
**Output**
b is greater than a
means a is less than b
"""
```

Run Program in Python IDLE

- Download latest Python:For downlod visit the link: *https://www.python.org/downloads/*
- Install it onto your system.
- Now go to start and search:IDLE
- After that you will get an Editor like this:
- Now write code and Press Enter to execute.

Run Program in Command Prompt

- Download latest Python:For downlod visit the link: *https://www.python.org/downloads/*
- Install it onto your system.
- Now open any editor,write python code and save it into your system directory with .py extension.
- **Example:D:/pythonpro/program.py**.
- program.py file:
- Now go to start search :**cmd**
- After that you will get a window litk this:
- Now go to the directory where you have stored your python file and execute it like this:

Run Program in PyCharm

- Download latest PyCharm:For downlod visit the link: *https://www.jetbrains.com/pycharm/download/*
- Install it onto your system.

Variable in Python

- It is a name of storage space which is used to store data.
- It's value may be changed.
- It always contains last value stored to it.
- There is no need to declare a variable in python.
- Variable is created by assigning a value to it.

Variable Initialization

```
name="Tom"
rollno=205
marks=85.6
```

- Here name rollno and marks are the name of variables.
- Value of name is Tom,rollno is 205 and marks is 85.6

Printing value of variable

- We can print the value of a variable using print() function.

```
#creating variables
name="Tom"
rollno=205
marks=85.6
#printing value of varibales
print("Name:",name)
print("Roll:",rollno)
print("Marks:",marks)

"""
**Output**
Name: Tom
Roll: 205
Marks: 85.6
"""
```

Assigning single value to multiple variable

- We can assign a single value to multiple variables using single statement in python.

```
#creating variables
a=b=c=90
#printing value of varibales
print("a:",a)
print("b:",b)
print("c:",c)
```

```
"""
**Output**
a: 90
b: 90
c: 90
"""
```

Assigning multiple values to multiple variables

- We can also assign multiple values to multiple variables using single statement in python.

```
#creating variables
a,b,c=70,80,90
#printing value of varibales
print("a:",a)
print("b:",b)
print("c:",c)

"""
**Output**
a: 70
b: 80
c: 90
"""
```

Rules to define a variable

- The first letter of a variable should be alphabet or underscore(_).
- The first letter of variable should not be digit.
- After first character it may be combination of alphabets and digits.
- Blank spaces are not allowed in variable name.
- Variable name should not be a keyword.
- Variable names are case sensitive for example marks and MARKS are different variables.

Local Variable

- A variable declared inside the body of the function is called local variable.
- Local variable can be used only inside that function in which it is defined.

```
#function definition
def add():
    #creating variable
    x=50
    y=30
    z=x+y
    print("Add=",z)
#Calling funtion
add()
#This line will raise an error
#because x is local variable
#and it can't be accessed
#outside function
print(x)

"""
**Output**
Add= 80
NameError: name 'x' is not defined
"""
```

Global Variable

- A variable which is created outside a function is called global variable.
- It can be used anywhere in the program.

```
#creating global variable
x=50
y=30
def add():
    #accessing global variable
    #inside a function
    print("Inside function Sum=",x+y)
#Calling funtion
add()
#accessing global variable
#outside a function
print("Outside function Sum=",x+y)

"""
**Output**
Inside function Sum= 80
Outside function Sum= 80
"""
```

Local and Global variable with same name

```
#creating global variable
x=50
def add():
    x=20
    # this line will print 20
    print("Inside function x=",x)
#Calling funtion
add()
#this line will print 50
print("Outside function x=",x)

"""
**Output**
Inside function x= 20
Outside function x= 50
"""
```

global keyword

- Simply we can't modify the value of global variable inside a function but by using global keyword we can modify the value of global variable inside a function.

```
#creating global variable
x=50
def add():
    global  x
    #updating value of x
    x=20
    # this line will print 20
    print("Inside function x=",x)
#Calling funtion
add()
#this line will print 20
print("Outside function x=",x)

"""
**Output**
Inside function x= 20
Outside function x= 20
"""
```

User input in Python

1. **input()** is a predefined function which is used to take user input in Python.
2. Default user input is of type string.

User input of String Value

```
name=input("Enter your name:")
print("Your name:",name)
#Printing of data type of name
print(type(name))

"""
**Output**
Enter your name:Aayushi
Your name: Aayushi
<class 'str'="">
"""
</class>
```

User input of Integer Value

1.**int()** is a predefined function which is used to convert string into integer.

#method 1

```
number=input("Enter any number:")
print("Type of number:",type(number))
#converting string into integer
num=int(number)
print("Given Number:",num)
#Printing of data type of num
print("Type of number:",type(num))

"""
**Output**
Enter any number:204
Type of number: <class 'str'="">
Given Number: 204
Type of number: <class 'int'="">
"""
</class></class>
```

```
#method 2
     number=int(input("Enter any number:"))
     print("Given Number:",number)
     #Printing of data type of number
     print("Type of number:",type(number))


     """
     **Output**
     Enter any number:204
     Given Number: 204
     Type of number: <class 'int'="">
     """
     </class>
```

Userinput of Float Value

1.**float()** is a predefined function which is used to convert string into float.

```
     marks=float(input("Enter your marks:"))
     print("Your Marks is:",marks)
     #Printing of data type of marks
     print("Type of number:",type(marks))

     """
     **Output**
     Enter your marks:84.2
     Your Marks is: 84.2
     Type of number: <class 'float'="">
     """
     </class>
```

Keywords in Python

- The word which is pre-defined in the library is called keyword.
- It's functionality is also pre-defined.
- Keyword can not be used as a variable,function name ,class name or as an any identifier.
- Example of keywords are if,else,for,def etc.

List of keywords in Python

```
asset def class continue break global for if from import nonlocal in not
is lambda True False None and as else finally elif del except raise try
or return pass
```

Operator

It is a special symbol which is used to perform logical or mathematical operation on data or variable.

Operand

- It is a data or variable on which the operation is to be performed.

Types of Operator

⇒Arithmetic Operators
⇒Relational Operators
⇒Logical Operators
⇒Assignment Operators
⇒Bitwise Operators
⇒Membership Operators
⇒Identity Operators

Arithmetic Operators

Symbol	Operation	Example
+	*Addition*	*x+y*
-	*Subtraction*	*x-y*
*	*Multiplication*	*x*y*
/	*Division*	*x/y*
%	*Modulus*	*x%y*
*	*Exponent*	*2*3=8*

```
x=5
y=2
print("x+y=",x+y)
print("x-y=",x-y)
print("x*y=",x*y)
print("x/y=",x/y)
print("x%y=",x%y)
print("x*y=",x*y)

"""
**Output**
x+y= 7
x-y= 3
x*y= 10
x/y= 2.5
x%y= 1
x**y= 25
"""
```

Relational Operators

Symbol	Operation	Example
==	*Equal to*	*2==3 returns 0*
!=	*Not equal to*	*2!=3 returns 1*
>	*Greaterthan*	*2>3 returns 0*
<	*Less than*	*2<3 returns 1*
>=	*Greater than or equal to*	*2>=3 returns 0*
<=	*Less than or equal to*	*2<=3 returns 1*

Logical Operators

(x>y)and(x>z) Here this expression returns true if both conditions are true.
(x>y)or(x>z) Here this expression returns true if any one or both conditions are true.
not(x>y) Not operator reverses the state means if the condition is true it returns false and if the condition is false it returns true.

```
#Show result according to percent
     x=int(input("Enter first number:"))
     y=int(input("Enter second number:"))
     z=int(input("Enter third number:"))
     if x>y and x>z:
         print(x," is greatest")
     if y>x and y>z:
         print(y," is greatest")
     if z>x and z>y:
         print(z," is greatest")

     """
     **Output**
     Enter first number:58
     Enter second number:86
     Enter third number:27
     86  is greatest
     """
```

Assignment Operators

Symbol	Example	Same as
=	*x=y*	*x=y*
+=	*x+=y*	*x=x+y*
-=	*x-=y*	*x=x-y*
=	*x=y*	*x=x*y*
/=	*x/=y*	*x=x/y*
%=	*x%=y*	*x=x%y*
**=*	*x=y*	*x=x*y*

```
x1 = 9
y1 = 4
x1 += y1 #x1=x1+y1
print(x1)
x2 = 9
y2 = 4
x2 -= y2 #x2=x2-y2
print(x2)
x3 = 9
y3 = 4
x3 *= y3 #x3=x3*y3
print(x3)
x4 = 9
y4 = 4
x4 /= y4 #x4=x4/y4
print(x4)
x5 = 9
y5 = 4
x5 %= y5 #x5=x5%y5
print(x5)
x6=2
y6=3
x6*=y6 #x6=x6*y6
print(x6)

"""
**Output**
13
5
36
2.25
1
8
"""
```

Bitwise Operators

Symbol	Operation	Example
&	Bitwise AND	x&y
|	Bitwise OR	x|y
<<	Shift Left	x<<2
>>	Shift Right	x>>2
^	X-OR	x^y

```
x=6
y=3
print("x&y=",x&y)
print("x|y=",x|y)
print("x>>2=",x>>2)
print("x<<2=",x<<2)
print("x^2=",x^2)
```

```
"""
**Output**
x&y= 2
x|y= 7
x>>2= 1
x<<2= 24
x^2= 4
"""
```

Membership Operators

- These operators are used to check specified element is present in a sequence(string,list,tuples etc) or not.
- It returns boolean values(True/False).

operator	Description
in	*Returns true if it finds specified element in sequence otherwise returns false.*
not in	*Returns true if it does not find specified element in sequence otherwise returns false.*

```
list=["Ravi","Shyam","Ashish"]
print("Ravi" in list)#true
print("Tom" in list)#false

print("Ravi" not in list)#false
print("Tom" not in list)#true

"""
**Output**
True
False
False
True
"""
```

Identity Operators

- These operators are used to compare two objects.
- It returns boolean values(True/False).

operator	Description
is	*It returns true if both variables are the same object otherwise returns false.*
is not	*It returns true if both variables are not the same object otherwise returns false.*

```
x=10
y=10
z=20
#true because of same identity of a and y
print(x is y)
#false because of same identity of a and y
print(x is not y)
#false because a and b have not same identity
print(x is z)
#true because a and b have not same identity
print(x is not z)

"""
**Output**
True
False
False
True
"""
```

Data Types in Python

- It is a type of data which is used in the program.
- There is no need of data type to declare a variable in Python.
- type() is predefined function of Python which is used to get data type of any data.

```
name="Rocky"
rollno=205
marks=85.6
print("Name Type:",type(name))
print("Rollno Type:",type(rollno))
print("Marks Type:",type(marks))

"""
**Output**
Name Type:
Rollno Type:
Marks Type:
"""
```

Type of Data Type

Python contains following data types:

- *Numbers*
- *String*
- *List*
- *Tuple*
- *Dictionary*

Comments in Python

- It is used to explain the code to make it more readable.
- It is not considered as a part of program,in other word we can say that compiler ignores the comment.
- Python comments are statements that are not executed by the compiler.

Types of comments in Python

- *Single line comment*
- *Multiline comment*

Single line comment

- It is used to comment only one line.
- Hash symbol(#) is used for single line comment.

```
#this is a single line comment
print("you are excellent");
```

Multiline comment

- Multiline comment is used to comment a block of code.
- Triple quotes (""") are used to comment multiline.
- Multiline comment starts with """ and ends with """.
- The text between """ and """ is not executed by the compiler.

```
"""
This is a multiline comment
Write a program to add two
numbers and stores it into
the third number
"""
x=50
y=30;
z=x+y
print("Sum=",z)

"""
**Output**
Sum= 80
"""
```

If statement

Syntax:

```
if  condition  :
    body
```

- It is used to test the condition and the result is based on the condition.
- If the condition is true its body will execute otherwise does not execute.

Example 1:

```
no=int(input("Enter any number:"))
if no>5:
    print("Number is greater than 5")

/*
Output
Enter any number:10
Number is greater than 5
*/
```

Example 2: Check given number is positive or negative or zero

```
no=int(input("Enter any number:"))
if no>0:
    print("Number is positive")
if no<0:
    print("Number is negative")
if no==0:
    print("Number is zero")
/*
Output
    Run1
        Enter any number:5
        Number is positive
    Run2
        Enter any number:-5
        Number is negative
*/
```

If else statement

Syntax:

```
if  condition  :
    body _of_if
else:
    body_of_else
```

- It is used to test the condition and gives the output in both situation either condition true or false.
- If the condition is true body of if will execute otherwise body of else execute.

Example 1:

```
no=int(input("Enter any number:"))
if no>5:
    print("Number is greater than 5")
else:
    print("Number is less than 5")
/*
Output
Enter any number:7
Number is greater than 5
*/
```

In the case of if at the place of condition always zero and non-zero value is checked in which zero means condition false and non-zero means condition true.

/Example 1/

```
if 10:
    print("Hello")
else:
    print("Hi")

/*
 ### Output ###
Hello
because 10 is non-zero value
*/
```

/Example 2/

```
if 0:
    print("Hello")
else:
    print("Hi")

/*
 ### Output ###
Hi
*/
```

/Example 3/

```
if 'A':
    print("Hello")
else:
    print("Hi")

/*
 ### Output ###
 Hello
 because ASCII value of A is 65
 which is non-zero
*/
```

Example 2: Check given number is even or odd.

```
no=int(input("Enter any number:"))
if no%2==0:
    print("Number is even")
else:
    print("Number is odd")
/*
Output
Enter any number:15
Number is odd
*/
```

If else if ladder statement

Syntax:

```
if  condition  :
    statement1
elif condition:
    statement2
elif condition:
    statement3
else:
    statement4
```

- It is used to test the condition.
- If executes only one condition at a time.
- The condition which is true first from the top will execute.

Example 1:

```
no=7
if no>10:  /false/
    print("Hello1")
elif no>5: /true/
    print("Hello2")
elif no>0: /true/
    print("Hello3")
else:
    print("Helllo4")

/*
 ### Output ###
 Hello2
 because it executes only one condition
 which becomes true first from top.
 */
```

else part will execute if all the conditions are false

```
no=-5
if no>10:
    print("Hello1")
elif no>5:
    print("Hello2")
elif no>0:
    print("Hello3")
else:
    print("Hello4")

/*
 ### Output ###
 Hello4
 because  all conditions are false.
*/
```

Example 2:Show result according to percent

```
percent=float(input("Enter your percentage:"))
if percent>=60:
    print("First division")
elif percent>=45:
    print("Second division")
elif percent>=33:
    print("Third division")
else:
    print("Sorry!!! You are fail.")

/*
### Output ###
Enter your percentage:56
Second division
*/
```

Nested if statement

Syntax:

```
if condition:
   #statements
   if condition:
       #statements
       if condition:
           #statements
```

- It is used to test the condition.
- One if inside another if is called nested if.

Example :

```
/Example 1/
     no=5
     if no>2:/true/
         if no<3:/false/
             print("Hello")
         print("Hi")
     /*
     ### Output ###
     Hi
     */
/Example 2/
     no=5
     if no>2:/true/
         if no>3:/true/
             print("Hello")
         print("Hi")
     /*
     ### Output ###
     Hello
     Hi
     */
/Example 3/
     no=5
     if no<2:/false/
         if no>3:/true/
             print("Hello")
         print("Hi")
     /*
     ### Output ###
     No output because outer if
     condition is false
     */
```

Example 2:Find greatest value in three number

```
print("Enter three number:")
a=int(input())
b=int(input())
c=int(input())
if a>b:
    if a>c:
        print(a," is greatest")
if b>a:
    if b>c:
        print(b," is greatest")
if c>a:
    if c>b:
        print(c," is greatest")
/*
### Output ###
Enter three number:
45
85
24
85  is greatest
*/
```

For loop

- For loop is used for sequential traversal.
- It can be used to traverse string or array.
- Iterating over a sequence is called traversal.
- For loop is used to iterate over a sequence(list, string tuple etc).

Syntax:

```
for variable in sequence:
    body of for_loop
```

- Here for is keyword.
- variable will take the value of the item inside the sequence on each iteration.
- Here sequence may be string , array etc.

Example:For loop with range function

- range() function is used to generate sequence of numbers.
- Sequence of range function is range(start,stop,step_size).
- Default step_size is 1.

/This will generate numbers from 5 to 9/

```
for x in range(5,10):
    print(x)
/*
### Output ###
5
6
7
8
9
*/
```

**/*Here 5 is initial value 10 is final value
and step size is 2 so the output will be 5 7 9*/**

```
for x in range(5,10,2):
    print(x)
/*
### Output ###
5
7
9
*/
```

Example:For loop with String

```
str="Easy"
for x in str:
    print(x)
/*
### Output ###
     E
     a
     s
     y
*/
```

Example:For loop with list

```
student=["Ravi","Rocky","Amisha"]
for x in student:
    print(x)
/*
### Output ###
     Ravi
     Rocky
     Amisha
*/
```

Example:Table of any number

```
/taking user input of no/
no=int(input("Enter any number:"))
print("Table of ",no," is given below")
for i in range(1,11):
    print(i*no)
/*
### Output ###
     Enter any number:5
     Table of  5  is given below
     5
     10
     15
     20
     25
     30
     35
     40
     45
     50
*/
```

Example:For loop with else

- We can also use else statement with for loop but it is not compulsory.
- Else part will execute if the items in the sequence used in for loop exhausts.

```
Student=["Amisha","Rinkal","Mahima"]
for name in Student:
    print(name)
else:
    print("This is else block")
/*
### Output ###
  Amisha
  Rinkal
  Mahima
  This is else block
*/
```

Else part is ignored if break statement terminate the loop

- *For better understanding see the example.*

```
Student=["Amisha","Rinkal","Mahima"]
for name in Student:
    if name=="Rinkal":
        break
    else:
        print(name)
else:
    print("This is else block")
/*
### Output ###
  Amisha
  because when if conditon will
  become true (when name=Rinkal) the
  break will terminate the loop
*/
```

While loop

Syntax:

```
while condition:
    body of loop
```

- It's body will execute until the given condition is true.

Example:

```
i=1
while i<=10:
    print(i)
    i=i+1
/*
        1
        2
        3
        4
        5
        6
        7
        8
        9
        10
*/
```

Example 2: Find factorial of any number

```
/*
factorial of 5=1x2x3x4x5
factorial of 6=1x2x3x4x5x6
factorial of N=1x2x3x....xN
*/
i=1
fact=1
no=int(input("Enter any number:"))
while i<=no:
    fact=fact*i
    i=i+1
print("Factorial of ",no," is ",fact)
/*
        Enter any number:6
        Factorial of  6  is  716
*/
```

Example:While loop with else

- When the condition of while fails then the else part is executed.

```
i=1
while i<=5:
    print(i)
    i=i+1
else:
    print("This is else part")
/*
   1
   2
   3
   4
   5
   This is else part
*/
```

- **Else part is ignored when break statement terminate the loop**

```
i=1
while i<=5:
    if i==3:
        break
    print(i)
    i=i+1
else:
    print("This is else part")
/*
   1
   2
   because when i will become 3
   break will terminate the loop
*/
```

While loop with string

```
str="Easy"
i=0
while i < len(str):
    print(str[i])
    i=i+1
/*
     E
     a
     s
     y
*/
```

While loop with list

```
girls=["Ritu","Ajeeta","Sonali"]
i=0
while i < len(girls):
    print(girls[i])
    i=i+1
/*
     Ritu
     Bulbul
     Sonali
*/
```

Jump Statement

- It is used to transfer the control from one point to another point in the program.

break statement

- It is used to transfer the control out of the body of loop.
- In other word we can say that it terminates the current loop.
- break statement are mostly used with loop(for loop or while loop).

Example 1:Program without break

```
for i in range(1,11):
    print(i,end=" ")
/*
     1 2 3 4 5 6 7 8 9 10
*/
```

Example 2:Same Program with break

```
for i in range(1,11):
    if i==5:
        break
    else:
        print(i,end=" ")
/*
     1 2 3 4
*/
```

continue statement

- It is used to skip the next statement and continue the loop.
- continue statement are mostly used with loop(for,while).

Example 1:

```
for i in range(1,11):
    if i==5:
        continue
    else:
        print(i,end=" ")
/*
     1 2 3 4 6 7 8 9 10
     5 will not print because at this
     time condition  will become true
     and loop will continue printing
*/
```

Example 2:

```
for i in range(1,11):
    if i>=5:
        continue
    else:
        print(i,end=" ")
/*
     1 2 3 4
     because condition of  if  will remain
     true when anil will be either 5 or
     greater than 5
*/
```

Example 3:

```
for i in range(1,11):
    if i<=5:
        continue
    else:
        print(i,end=" ")
/*
     6 7 8 9 10
     because condition of if is true
     whene value of i is either 5
     or less than 5
*/
```

Example 4:

```
for i in range(1,11):
    if i!=5:
        continue
    else:
        print(i,end=" ")
/*
     5
     because else part will
     execute if and only if the
     value of i is 5
*/
```

List in python

It is a collection of data of different data type.
It is used to store list of values.
A list is created by putting list of comma-separated values between square brackets.
Create list

```
str_list=["Apple","Orange","Mango"]
int_list=[15,25,36,84,59]
float_list=[2.3,5.6,1.4,9.6]
mixed_list=["Easy",205,25.3]
```

Access values of list using index
Value of list can be accessed using index number.
Index number is always an integer value and starts with 0.

```
fruit_list=["Apple","Orange","Mango"]
print("I like ",fruit_list[0])
print("I like ",fruit_list[2])
"""
**Output**
I like  Apple
I like  Mango
"""


int_list=[5,10,15,20,25,30,35,40,45,50]
#Print will start at index 1 (included) and end at index 4 (not included).
print(int_list[1:4])
"""
**Output**
[10, 15, 20]
"""
```

Access value of list using negative index
Negative indexes start from the end of the list.
Negative index always starts with -1.
For example fruit_list=["Apple","Orange","Mango"] here index of Mango ,Orange and Apple are -1,-2 and -3.

```
fruit_list=["Apple","Orange","Mango"]
print(fruit_list[-3])#Apple
print(fruit_list[-2])#Orange
print(fruit_list[-1])#Mango
"""
**Output**
Apple
Orange
Mango
"""
```

Access values of list using loop

```
fruit_list=["Apple","Orange","Mango"]
for name in fruit_list:
    print("I like ",name)
"""
**Output**
I like  Apple
I like  Orange
I like  Mango
"""
```

Update item of list

```
fruit_list=["Apple","Orange","Mango"]
print("Before Updation")
print(fruit_list)
#this line will replace Orange with Banana
fruit_list[1]="Banana"
print("After Updation")
print(fruit_list)
"""
**Output**
Before Updation
['Apple', 'Orange', 'Mango']
After Updation
['Apple', 'Banana', 'Mango']
"""
```

Length of list
len() function is used to get length of list.

```
fruit_list=["Apple","Orange","Mango"]
print("Length of list is ",len(fruit_list))
"""
**Output**
Length of list is  3
"""
```

Add items into list
append() function is used to add new items into list.

```
fruit_list=["Apple","Orange","Mango"]
print("Before insertion")
print(fruit_list)
#this line will add  Banana at the end of list
fruit_list.append("Banana")
print("After insertion")
print(fruit_list)
"""
**Output**
Before insertion
['Apple', 'Orange', 'Mango']
After insertion
['Apple', 'Orange', 'Mango', 'Banana']
"""
```

Add item at particular index
insert() function is used to add new items into list at particular index.

```
fruit_list=["Apple","Orange","Mango"]
print("Before insertion")
print(fruit_list)
#this line will add  Banana at index 1
fruit_list.insert(1,"Banana")
print("After insertion")
print(fruit_list)
"""
**Output**
Before insertion
['Apple', 'Orange', 'Mango']
After insertion
['Apple', 'Banana', 'Orange', 'Mango']
"""
Delete item from list
remove() function is used to delete or remove item from list.

fruit_list=["Apple","Orange","Mango"]
print("Before deletion")
print(fruit_list)
#this line will delete  Orange from list
fruit_list.remove("Orange")
print("After deletion")
print(fruit_list)
"""
**Output**
Before deletion
['Apple', 'Orange', 'Mango']
After deletion
['Apple', 'Mango']
"""
Delete item using index
pop() function is used to delete or remove item from list using index.

fruit_list=["Apple","Orange","Mango"]
print("Before deletion")
print(fruit_list)
#this line will delete  Orange from list
fruit_list.pop(1)
print("After deletion")
print(fruit_list)
"""
**Output**
Before deletion
['Apple', 'Orange', 'Mango']
After deletion
['Apple', 'Mango']
"""
pop() function will delete last item if we do not pass index

fruit_list=["Apple","Orange","Mango"]
print("Before deletion")
print(fruit_list)
#this line will delete  Mango from list
fruit_list.pop()
print("After deletion")
print(fruit_list)
"""
**Output**
Before deletion
['Apple', 'Orange', 'Mango']
After deletion
['Apple', 'Orange']
"""
```

del keyword is also used to delete item using index.

```
fruit_list=["Apple","Orange","Mango"]
print("Before deletion")
print(fruit_list)
#this line will delete  Orange from list
del fruit_list[1]
print("After deletion")
print(fruit_list)
"""
**Output**
Before deletion
['Apple', 'Orange', 'Mango']
After deletion
['Apple', 'Mango']
"""
```

del keyword is also used to delete all the items of list.

```
fruit_list=["Apple","Orange","Mango"]
print("List Items")
print(fruit_list)
#this line will delete  all the items of the list
del fruit_list
print("Deleted successfully")
"""
**Output**
List Items
['Apple', 'Orange', 'Mango']
Deleted successfully
"""
```

Clear List

clear() function is used to clear or empty the list.

```
fruit_list=["Apple","Orange","Mango"]
print("Before clear")
print(fruit_list)
#this line will empty the list
fruit_list.clear()
print("After clear")
print(fruit_list)
```

```
"""
**Output**
Before clear
['Apple', 'Orange', 'Mango']
After clear
[]
"""
```

Copy one list into another
copy() function is used to copy one list into another.

```
list1=["Apple","Orange","Mango"]
print("list1 items")
print(list1)
#this line will copy list1 items into list2
list2=list1.copy()
print("list2 items")
print(list2)
"""
**Output**
list1 items
['Apple', 'Orange', 'Mango']
list2 items
['Apple', 'Orange', 'Mango']
"""
```

Join two lists using + symbol
We can join two list using plus(+) operator.

```
list1=["Apple","Orange","Mango"]
list2=["Cherry","Grapes","Melon"]
#this line will join list1 and list2
list3=list1+list2
print("list3 items")
print(list3)
"""
**Output**
list3 items
['Apple', 'Orange', 'Mango', 'Cherry', 'Grapes', 'Melon']
"""
```

Join two lists using extend function
extend() function is also used to join two list.

```
list1=["Apple","Orange","Mango"]
list2=["Cherry","Grapes","Melon"]
#this line will join list1 and list2
list1.extend(list2)
print("list1 items")
print(list1)
"""
**Output**
list1 items
['Apple', 'Orange', 'Mango', 'Cherry', 'Grapes', 'Melon']
```

```
"""
Join two lists using append function
append() function is also used to join two list.

list1=["Apple","Orange","Mango"]
list2=["Cherry","Grapes","Melon"]
#this line will add list2 items into list1
for name in list2:
  list1.append(name)
print("list1 items")
print(list1)
"""
**Output**
list1 items
['Apple', 'Orange', 'Mango', 'Cherry', 'Grapes', 'Melon']
"""
```

List Function

Python contains the following list functions.
1.len()
It is used to get the numbers of elements in list.

```
fruit_list=["Apple","Orange","Mango"]
print ("List elements : ", fruit_list)
#this line will print length of list
print("Length of list is ",len(fruit_list))
"""
**Output**
List elements :  ['Apple', 'Orange', 'Mango']
Length of list is  3
"""
```

2.max()
It is used to get maximum value from the list.
In case of string focus on ASCII value of first letter of list items.

```
fruit_list=["Apple","Orange","Mango"]
print("Fruits list :",fruit_list)
print ("Max elements : ", max(fruit_list))

animal_list=["Zebra","Dog","Elephant"]
print("Animal list :",animal_list)
print ("Max elements : ", max(animal_list))

int_list=[45,85,36]
print("int list :",int_list)
print ("Max elements : ", max(int_list))

"""
**Output**
Fruits list : ['Apple', 'Orange', 'Mango']
Max elements :  Orange
Animal list : ['Zebra', 'Dog', 'Elephant']
Max elements :  Zebra
int list : [45, 85, 36]
Max elements :  85
"""
```

3.min()
It is used to get minimum value from the list.
In case of string focus on ASCII value of first letter of list items.

```
fruit_list=["Apple","Orange","Mango"]
print("Fruits list :",fruit_list)
print ("Min elements : ", min(fruit_list))

animal_list=["Zebra","Dog","Elephant"]
print("Animal list :",animal_list)
print ("Min elements : ", min(animal_list))
```

```
int_list=[45,85,36]
print("int list :",int_list)
print ("Min elements : ", min(int_list))

"""
**Output**
Fruits list : ['Apple', 'Orange', 'Mango']
Min elements :  Apple
Animal list : ['Zebra', 'Dog', 'Elephant']
Min elements :  Dog
int list : [45, 85, 36]
Min elements :  36
"""
```
4.list()
It is used to convert sequence types (tuple) into list.

```
#tuple is created using parentheses
fruit_tuple=("Apple","Orange","Mango")
print("Tuple Items:",fruit_tuple)
#this line convert tuple into list
fruit_list=list(fruit_tuple)
print("List Items :",fruit_list)
"""
**Output**
Tuple Items: ('Apple', 'Orange', 'Mango')
List Items : ['Apple', 'Orange', 'Mango']
"""
```

List Methods

Python contains the following list methods.

1.append()

It is used to add the new element at the end of the list.

```
fruit_list=["Apple","Orange","Mango"]
print("Fruits list :",fruit_list)
#this line will add Cherry at the end of list
fruit_list.append("Cherry")
print("New Fruits list :",fruit_list)
"""
**Output**
Fruits list : ['Apple', 'Orange', 'Mango']
New Fruits list : ['Apple', 'Orange', 'Mango', 'Cherry']
"""
```

2.clear()

This function is used to empty the list.

```
fruit_list=["Apple","Orange","Mango"]
print("Fruits list :",fruit_list)
#this line will empty the list
fruit_list.clear()
print("New Fruits list :",fruit_list)
"""
**Output**
Fruits list : ['Apple', 'Orange', 'Mango']
New Fruits list : []
"""
```

3.count()

This method counts the number of occurrence of particular item in a list.

```
number_list=[10,16,40,50,60,40,16,30,16,10,50]
print("Numbers list :",number_list)
print("Total Count of 16 :",number_list.count(16))
"""
**Output**
Numbers list : [10, 16, 40, 50, 60, 40, 16, 30, 16, 10, 50]
Total Count of 16 : 3
"""
```

4.copy()

This function copies the elements of one list into another.

```
list1=["Apple","Orange","Mango"]
print("List1 items:",list1)
#this line will copy list1 items into list2
list2=list1.copy()
print("List2 items:",list2)
"""
**Output**
List1 items: ['Apple', 'Orange', 'Mango']
```

```
List2 items: ['Apple', 'Orange', 'Mango']
"""
5.extend()
This function is used to join two list.

list1=["Apple","Orange","Mango"]
list2=["Cherry","Grapes","Melon"]
#this line will join list1 and list2
list1.extend(list2)
print("list1 items")
print(list1)
"""
**Output**
list1 items
['Apple', 'Orange', 'Mango', 'Cherry', 'Grapes', 'Melon']
"""
6.index()
It returns the lowest index of given element.

number_list=[10,30,16,50,60,40,16,30,16,10,50]
print("Index of 16 :",number_list.index(16))
"""
**Output**
Index of 16 : 2
Note:16 is present at index 2 , 6 and 8
"""
7.insert()
insert() function is used to add new items into list at particular index.

fruit_list=["Apple","Orange","Mango"]
print("Before insertion")
print(fruit_list)
#this line will add  Banana at index 1
fruit_list.insert(1,"Banana")
print("After insertion")
print(fruit_list)
"""
**Output**
Before insertion
['Apple', 'Orange', 'Mango']
After insertion
['Apple', 'Banana', 'Orange', 'Mango']
"""
8.pop()
This function deletes the element of given index.
It deletes last item if we do not pass index

fruit_list=["Apple","Orange","Mango","Cherry"]
print("Fruits list :",fruit_list)
#this line will delete  last element Cherry
fruit_list.pop()
print("Fruits list :",fruit_list)
```

```
#this line will delete element at index 1(Orange)
fruit_list.pop(1)
print("Fruits list :",fruit_list)
"""
**Output**
Fruits list : ['Apple', 'Orange', 'Mango', 'Cherry']
Fruits list : ['Apple', 'Orange', 'Mango']
Fruits list : ['Apple', 'Mango']
"""
9.remove()
remove() function is used to delete or remove item from list.

fruit_list=["Apple","Orange","Mango"]
print("Before deletion")
print(fruit_list)
#this line will delete  Orange from list
fruit_list.remove("Orange")
print("After deletion")
print(fruit_list)
"""
**Output**
Before deletion
['Apple', 'Orange', 'Mango']
After deletion
['Apple', 'Mango']
"""
10.reverse()
This function reverses elements of the list.

fruit_list=["Apple","Orange","Mango","Cherry"]
print("Fruits list :",fruit_list)
#this line will reverse the list items
fruit_list.reverse()
print("Reverse Fruits list :",fruit_list)
"""
**Output**
Fruits list : ['Apple', 'Orange', 'Mango', 'Cherry']
Reverse Fruits list : ['Cherry', 'Mango', 'Orange', 'Apple']
"""
11.sort()
This function Sorts the list.
By using this function we can display the list items in ascending order
or descending order.

#Example 1:Ascending order
str_list=["Apple","Orange","Mango","Cherry"]
int_list=[58,20,46,36]
char_list=['E','A','S','Y']
print("List item before sorting")
print(str_list)
print(int_list)
print(char_list)
```

```
#Sort the lists in ascending order
str_list.sort()
int_list.sort()
char_list.sort()
print("List item after sorting")
print(str_list)
print(int_list)
print(char_list)
"""
**Output**
List item before sorting
['Apple', 'Orange', 'Mango', 'Cherry']
[58, 20, 46, 36]
['E', 'A', 'S', 'Y']
List item after sorting
['Apple', 'Cherry', 'Mango', 'Orange']
[20, 36, 46, 58]
['A', 'E', 'S', 'Y']
"""

#Example 2:Descending order
str_list=["Apple","Orange","Mango","Cherry"]
int_list=[58,20,46,36]
char_list=['E','A','S','Y']
print("List item before sorting")
print(str_list)
print(int_list)
print(char_list)
#Sort the lists in descending order
str_list.sort(reverse=True)
int_list.sort(reverse=True)
char_list.sort(reverse=True)
print("List item after sorting descending order")
print(str_list)
print(int_list)
print(char_list)
"""
**Output**
List item before sorting
['Apple', 'Orange', 'Mango', 'Cherry']
[58, 20, 46, 36]
['E', 'A', 'S', 'Y']
List item after sorting descending order
['Orange', 'Mango', 'Cherry', 'Apple']
[58, 46, 36, 20]
['Y', 'S', 'E', 'A']
"""
```

Sets in python

It is an unordered collection of data of different data types.
Set does not contain duplicate data.
A set is created using curly brackets.
Create set

```
str_set={"Apple","Orange","Mango"}
int_set={15,25,36,84,59}
float_set={2.3,5.6,1.4,9.6}
mixed_set={"Easy",165,25.3}
```

Access values of set using loop
We can't access items of set using index value because it is unordered collection of data.
We cannot be sure in which order the items will appear because sets are unordered.
For better understanding see the example.

```
fruit_set={"Apple","Orange","Mango"}
print("Fruit set :",fruit_set)
"""
**Output**
Run 1
Fruit set : {'Mango', 'Orange', 'Apple'}
Run 2
Fruit set : {'Apple', 'Mango', 'Orange'}
"""
```

Update item of set
We can not change the value of set.

```
fruit_set={"Apple","Orange","Mango"}
#this line will generate error
#because we can't change the value of set
fruit_set[1]="Banana"
#TypeError: 'set' object does not support item assignment
```

Length of set
len() function is used to get length of set.

```
fruit_set={"Apple","Orange","Mango"}
print("Length of set is ",len(fruit_set))
"""
**Output**
Length of set is  3
"""
```

Add items into set
add() function is used to add new item in a set.

```
fruit_set={"Apple","Orange","Mango"}
print("Fruit Set:",fruit_set)
#this line will add
#Cherry at the end of set
fruit_set.add("Cherry")
print("Fruit Set:",fruit_set)
```

```
"""
**Output**
Fruit Set: {'Orange', 'Mango', 'Apple'}
Fruit Set: {'Orange', 'Mango', 'Cherry', 'Apple'}
"""
```

Add more than one item to a set
update() function is used to add more than one item to a set.
update() and add() function discard the duplicate elements.

```
fruit_set={"Apple","Orange","Mango"}
print("Fruit Set:",fruit_set)
#Add multiple items to a set
fruit_set.update(["Cherry","Banana","Apple"])
print("Fruit Set:",fruit_set)
"""
**Output**
Fruit Set: {'Mango', 'Orange', 'Apple'}
Fruit Set: {'Orange', 'Banana', 'Mango', 'Cherry', 'Apple'}
"""
```

Delete item from set using remove() function
remove() function is used to remove specified item from a set.

```
fruit_set={"Apple","Orange","Mango"}
print("Fruit Set:",fruit_set)
#this line will remove Orange from set
fruit_set.remove("Orange")
print("Fruit Set:",fruit_set)
"""
**Output**
Fruit Set: {'Mango', 'Apple', 'Orange'}
Fruit Set: {'Mango', 'Apple'}
"""
```

Delete item from set using discard() function
discard() function is also used to remove specified item from a set.

```
fruit_set={"Apple","Orange","Mango"}
print("Fruit Set:",fruit_set)
#this line will remove Orange from set
fruit_set.discard("Orange")
print("Fruit Set:",fruit_set)
"""
**Output**
Fruit Set: {'Mango', 'Apple', 'Orange'}
Fruit Set: {'Mango', 'Apple'}
"""
```

The difference between remove() and discard() function is that if the specified element does not exist in the set then remove() function will raise an error but discard() function will not raise an error.
Join two sets
We can join two set using union() function.

```
set1={"Apple","Orange","Mango"}
set2={"Cherry","Grapes","Melon"}
#this line will join set1 and set2
set3=set1.union(set2)
print("set3 items")
print(set3)
"""
**Output**
set3 items
{'Orange', 'Grapes', 'Cherry', 'Mango', 'Apple', 'Melon'}
"""
Program to search particular element in set

fruit_set = {"Apple", "Orange", "Mango"}
str=input("Enter any string to search:")
if str in fruit_set:
  print(str," is found")
else:
  print("Not found")
"""
**Output**
Enter any string to search:Apple
Apple  is found
"""
Clear set
clear() function is used to clear or empty the set.

fruit_set={"Apple","Orange","Mango"}
print("Before clear")
print(fruit_set)
#this line will empty the list
fruit_set.clear()
print("After clear")
print(fruit_set)
"""
**Output**
Before clear
{'Apple', 'Orange', 'Mango'}
After clear
set()
"""
Delete set
del keyword is also used to delete set completely.

fruit_set={"Apple","Orange","Mango"}
print("Set Items")
print(fruit_set)
#this line will delete  set
del fruit_set
print("Deleted successfully")
"""
**Output**
```

```
Set Items
{'Apple', 'Orange', 'Mango'}
Deleted successfully
"""
Some commom Operation on a set

Set_A={4,5,6,9}
Set_B={1,2,5,6,8}
#This operation will return
#common elements in both set
print("AND or Intersection Operation:",Set_A & Set_B)
#This operation will combine
#elements of both set and discard duplicate elements
print("OR or Union Operation:",Set_A | Set_B)
#this operation will return Set_A elements
#that does not exist in the Set_B
print("Set difference:",Set_A-Set_B)
"""
**Output**
AND or Intersection Operation: {5, 6}
OR or Union Operation: {1, 2, 4, 5, 6, 8, 9}
Set difference: {9, 4}
"""
```

Tuple in python

It is a collection of data of different data types.
We can not change the value of tuples.
It is used to store tuple of values.
A tuple is created using parentheses.
Create tuple

```
str_tuple=("Apple","Orange","Mango")
int_tuple=(15,25,36,84,59)
float_tuple=(2.3,5.6,1.4,9.6)
mixed_tuple=("Easy",205,25.3)
```

Access values of tuple using index
Value of tuple can be accessed using index number.
Index number is always an integer value and starts with 0.

```
fruit_tuple=("Apple","Orange","Mango")
print("I like ",fruit_tuple[0])
print("I like ",fruit_tuple[2])
"""
**Output**
I like  Apple
I like  Mango
"""

int_tuple=(5,10,15,20,25,30,35,40,45,50)
#Print will start at index 1 (included) and end at index 4 (not
included).
print(int_tuple[1:4])
"""
**Output**
(10, 15, 20)
"""
```

Access value of tuple using negative index
Negative indexes start from the end of the tuple.
Negative index always starts with -1.
For example fruit_tuple=("Apple","Orange","Mango") here index of Mango ,Orange and Apple are -1,-2 and -3.

```
fruit_tuple=("Apple","Orange","Mango")
print(fruit_tuple[-3])#Apple
print(fruit_tuple[-2])#Orange
print(fruit_tuple[-1])#Mango
"""
**Output**
Apple
Orange
Mango
"""
```

Access values of tuple using loop

```
fruit_tuple=("Apple","Orange","Mango")
for name in fruit_tuple:
    print("I like ",name)
"""
**Output**
I like  Apple
I like  Orange
I like  Mango
"""
```

Update item of tuple
We can not change the value of tuple.

```
fruit_tuple=("Apple","Orange","Mango")
#this line will generate error
#because we can't change the value of tuple
fruit_tuple[1]="Banana"
```

We can update the value of tuple using list.

```
fruit_tuple=("Apple","Orange","Mango")
print("Tuple Before Updation:",fruit_tuple)
#Convert tuple into list
fruit_list=list(fruit_tuple)
#Update Orange with Banana
fruit_list[1]="Banana"
#Convert list into tuple
fruit_tuple=tuple(fruit_list)
print("Tuple after Updation:",fruit_tuple)

"""
**Output**
Tuple Before Updation: ('Apple', 'Orange', 'Mango')
Tuple after Updation: ('Apple', 'Banana', 'Mango')
"""
```

Length of tuple
len() function is used to get length of tuple.

```
fruit_tuple=("Apple","Orange","Mango")
print("Length of tuple is ",len(fruit_tuple))
"""
**Output**
Length of tuple is  3
"""
```

Add items into tuple
We can't add new item to tuple once a tuple is created.
Tuples are unchangeable.

```
fruit_tuple=("Apple","Orange","Mango")
#this line will generate an error
fruit_tuple.append("Banana")
```

Delete item from tuple
We can't delete item from tuple because it is unchangeable.
But we can delete a tuple completely using del keyword.

```
fruit_tuple=("Apple","Orange","Mango")
```

```
print("Fruit tuple:",fruit_tuple)
del fruit_tuple;
print("Deleted successfully")
#this line will generate error
print(fruit_tuple)
"""
**Output**
Fruit tuple: ('Apple', 'Orange', 'Mango')
Deleted successfully
NameError: name 'fruit_tuple' is not defined
"""
Join two tuples using + symbol
We can join two tuple using plus(+) operator.

tuple1=("Apple","Orange","Mango")
tuple2=("Cherry","Grapes","Melon")
#this line will join tuple1 and tuple2
tuple3=tuple1+tuple2
print("tuple3 items")
print(tuple3)
"""
**Output**
tuple3 items
('Apple', 'Orange', 'Mango', 'Cherry', 'Grapes', 'Melon')
"""
Program to search particular element in tuple

fruit_tuple = ("Apple", "Orange", "Mango")
str=input("Enter any string to search:")
if str in fruit_tuple:
  print(str," is found")
else:
  print("Not found")
"""
**Output**
Enter any string to search:Orange
Orange  is found
"""
```

Tuple Function

Python contains the following tuple functions.
1.len()
It is used to get the numbers of elements in tuple.

```
fruit_tuple=("Apple","Orange","Mango")
print ("tuple elements : ", fruit_tuple)
#this line will print length of tuple
print("Length of tuple is ",len(fruit_tuple))
"""
***Output***
tuple elements :  ('Apple', 'Orange', 'Mango')
Length of tuple is  3
"""
```

2.max()
It is used to get maximum value from the tuple.
In case of string focus on ASCII value of first letter of tuple items.

```
fruit_tuple=("Apple","Orange","Mango")
print("Fruits tuple :",fruit_tuple)
print ("Max elements : ", max(fruit_tuple))

animal_tuple=("Zebra","Dog","Elephant")
print("Animal tuple :",animal_tuple)
print ("Max elements : ", max(animal_tuple))

int_tuple=(45,85,36)
print("int tuple :",int_tuple)
print ("Max elements : ", max(int_tuple))

"""
***Output***
Fruits tuple : ('Apple', 'Orange', 'Mango')
Max elements :  Orange
Animal tuple : ('Zebra', 'Dog', 'Elephant')
Max elements :  Zebra
int tuple : (45, 85, 36)
Max elements :  85
"""
```

3.min()
It is used to get minimum value from the tuple.
In case of string focus on ASCII value of first letter of tuple items.

```
fruit_tuple=("Apple","Orange","Mango")
print("Fruits tuple :",fruit_tuple)
print ("Min elements : ", min(fruit_tuple))

animal_tuple=("Zebra","Dog","Elephant")
print("Animal tuple :",animal_tuple)
print ("Min elements : ", min(animal_tuple))
```

```
int_tuple=(45,85,36)
print("int tuple :",int_tuple)
print ("Min elements : ", min(int_tuple))

"""
***Output***
Fruits tuple : ('Apple', 'Orange', 'Mango')
Min elements :  Apple
Animal tuple : ('Zebra', 'Dog', 'Elephant')
Min elements :  Dog
int tuple : (45, 85, 36)
Min elements :  36
"""
```

4.tuple()

It is used to convert list into tuple.

```
#tuple is created using parentheses
fruit_list=["Apple","Orange","Mango"]
print("List Items:",fruit_list)
#this line convert list into tuple
fruit_tuple=tuple(fruit_list)
print("Tuple Items :",fruit_tuple)
"""
***Output***
List Items: ['Apple', 'Orange', 'Mango']
Tuple Items : ('Apple', 'Orange', 'Mango')
"""
```

Dictionary in python

It is an unordered collection of data of different data types.
Items of dictionary can be changed.
A dictionary is created using curly brackets.
Dictionary items are in the form of key-value pairs.
Syntax

```
dict_name={key1:value1,key2:value2,key3:value3,.....}
```

Create dictionary

```
student={
    "name":"Rocky Singh",
    "rollno":305,
    "percent":85.6
}
print(student)
"""
***Output***
{'name': 'Rocky Singh', 'rollno': 305, 'percent': 85.6}
"""
```

Access items of dictionary
We can access items of dictionary using key name.

```
student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6
}
print("Student Name:",student["name"])
"""
***Output***
Student Name: Amisha
"""
```

We can also access items of dictionary using get() function.

```
student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6
}
print("Student Name:",student.get("name"))
"""
***Output***
Student Name: Amisha
"""
```

Access values of dictionary using loop
Printing key.

```
student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6
}
```

```
for key in student:
    print(key)
"""
***Output***
name
rollno
percent
"""
Printing key and value.

student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6
}
for key in student:
    print(key,"=",student[key])
"""
***Output***
name = Amisha
rollno = 305
percent = 85.6
"""
Update items of dictionary
We can change the value of dictionary.

student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6
}
print("Dictionary before Update:\n",student)
#this line will replace 85.6 with 92.3
student["percent"]=92.3
print("Dictionary after Update:\n",student)
"""
***Output***
Dictionary before Update:
 {'name': 'Amisha', 'rollno': 305, 'percent': 85.6}
Dictionary after Update:
 {'name': 'Amisha', 'rollno': 305, 'percent': 92.3}
"""
Length of dictionary
len() function is used to get length of dictionary.

student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6,
    "branch":"Computer Science"
}
print("Length of dictionary:",len(student))
```

```
"""
***Output***
Length of dictionary: 4
"""
Add items into dictionary
we can also add new item to a dictionary using ney key.

student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6,
}
print("Dictionary:",student)
#adding new item to a dictionary
student["branch"]="Computer Science"
print("Dictionary:",student)
"""
***Output***
Dictionary: {'name': 'Amisha', 'rollno': 305, 'percent': 85.6}
Dictionary: {'name': 'Amisha', 'rollno': 305, 'percent': 85.6, 'branch':
'Computer Science'}
"""
Delete item from dictionary using pop() function
pop() function is used to remove specified item from a dictionary.

student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6,
}
print("Dictionary before deletion:\n",student)
#Delete item from dictionary
student.pop("rollno")
print("Dictionary after deletion:\n",student)
"""
***Output***
Dictionary before deletion:
 {'name': 'Amisha', 'rollno': 305, 'percent': 85.6}
Dictionary after deletion:
 {'name': 'Amisha', 'percent': 85.6}
 """

Delete item from dictionary using del keyword
del keyword is also used to remove specified item from a dictionary.

student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6,
}
print("Dictionary before deletion:\n",student)
#Delete item from dictionary
```

```
del student["rollno"]
print("Dictionary after deletion:\n",student)
"""
***Output***
Dictionary before deletion:
 {'name': 'Amisha', 'rollno': 305, 'percent': 85.6}
Dictionary after deletion:
 {'name': 'Amisha', 'percent': 85.6}
 """
```

Clear dictionary
clear() function is used to clear or empty the dictionary.

```
student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6,
}
print("Dictionary before clear:\n",student)
#this line will empty the dictionary
student.clear()
print("Dictionary after clear:\n",student)
"""
***Output***
Dictionary before clear:
 {'name': 'Amisha', 'rollno': 305, 'percent': 85.6}
Dictionary after clear:
 {}
 """
```

Delete dictionary
del keyword is also used to delete dictionary completely.

```
student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6,
}
print("Dictionary :\n",student)
#this line will empty the dictionary
del student
print("Deleted successfully")
"""
***Output***
Dictionary :
 {'name': 'Amisha', 'rollno': 305, 'percent': 85.6}
Deleted successfully
 """
```

Nested dictionary
We can also create nested dictionary.

```
studentList={
```

```
"Amisha":{
    "rollno":305,
    "percent":85.6
     },
"Rocky":{
    "rollno":166,
    "percent":83.6
     }
}
print("Amisha Info :",studentList["Amisha"])
print("Rocky Info :",studentList["Rocky"])
"""
***Output***
Amisha Info : {'rollno': 305, 'percent': 85.6}
Rocky Info : {'rollno': 166, 'percent': 83.6}
 """
```

Dictionary Methods

Python contains the following dictionary methods.

1.clear()

clear() function is used to clear or empty the dictionary.

```
student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6,
}
print("Dictionary before clear:\n",student)
#this line will empty the dictionary
student.clear()
print("Dictionary after clear:\n",student)
"""
***Output***
Dictionary before clear:
 {'name': 'Amisha', 'rollno': 305, 'percent': 85.6}
Dictionary after clear:
 {}
 """
```

2.copy()

This method returns a shallow copy of the dictionary.

```
dict1={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6,
}
print("Dictionary 1:",dict1)
#copy dictionary1 into dictionary2
dict2=dict1.copy()
print("Dictionary 2:",dict2)
"""
***Output***
Dictionary 1: {'name': 'Amisha', 'rollno': 305, 'percent': 85.6}
Dictionary 2: {'name': 'Amisha', 'rollno': 305, 'percent': 85.6}
 """
```

3.fromkeys()

This method creates a new dictionary with specified key and value.

```
keyset=('key1','key2','key3')
dictionary1=dict.fromkeys(keyset)
print("Without Value",dictionary1)
dictionary2=dict.fromkeys(keyset,50)
print("With Value",dictionary2)
```

4.get()

It returns the value of the specified key.

```
student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6
}
print("Student Name:",student.get("name"))
"""
***Output***
Student Name: Amisha
"""
5.items()
It retirns all the key-value pair of the dictionary.

student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6,
}
print("Dictionary:",student.items())

"""
***Output***
Dictionary:   dict_items([('name',   'Amisha'),   ('rollno',   305),
('percent', 85.6)])
 """

6.key()
It returns all the keys of the dictionary.

student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6,
}
print("Dictionary:",student.keys())

"""
***Output***
Dictionary: dict_keys(['name', 'rollno', 'percent'])
 """

7.setdefault()
This method is used to set a default value to a key.
If the specified key is present in the dictionary then it retunrs its
value.

student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6,
}
#rollno is present in dictionary
#so it will return 305
```

```
x=student.setdefault("rollno")
print("Dictionary:",student)
print("Rollno:",x)
"""
***Output***
Dictionary: {'name': 'Amisha', 'rollno': 305, 'percent': 85.6}
Rollno: 305
 """
```

If the specified key is not present in the dictionary then it insert key into dictionary with default value 'None'.

```
student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6,
}
print("Dictionary:",student)
student.setdefault("Branch")
print("Dictionary:",student)
"""
***Output***
Dictionary: {'name': 'Amisha', 'rollno': 305, 'percent': 85.6}
Dictionary: {'name': 'Amisha', 'rollno': 305, 'percent': 85.6, 'Branch': None}
 """
```

We can also set value of key using setdefault function.

```
student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6,
}
print("Dictionary:",student)
#setting key with value
student.setdefault("Branch","Computer Science")
print("Dictionary:",student)
"""
***Output***
Dictionary: {'name': 'Amisha', 'rollno': 305, 'percent': 85.6}
Dictionary: {'name': 'Amisha', 'rollno': 305, 'percent': 85.6, 'Branch': 'Computer Science'}
 """
```

8.values()
It returns all the values of the dictionary.

```
student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6,
}
```

```
print("Dictionary:",student.values())

"""
***Output***
Dictionary: dict_values(['Amisha', 305, 85.6])
 """
```

9.update()
This method updates the dictionary with the key and value pairs.

```
#creating dictionary
student={
    "name":"Amisha",
    "rollno":305,
    "percent":85.6,
}
print("Student Info:",student)
#updating student info
student.update({"Branch":"Computer Science"})
print("Updated Student Info:",student)
"""
***Output***
Student Info: {'name': 'Amisha', 'rollno': 305, 'percent': 85.6}
Updated Student Info: {'name': 'Amisha', 'rollno': 305, 'percent': 85.6,
'Branch': 'Computer Science'}
 """
```

String in Python

String is a collection of characters.
It is created using single quotes or double quotes or triple quotes.
Creating String

```
#string with single quotes
str1='I love PyTHON'
#string with double quotes
str2="I love PyTHON"
#string with triple quotes
str3='''I love PyTHON'''
```

Accessing or Printing String
We can print string using print() function.
We can also print particular character of string using index number.
String are stored in the form of array.

```
#creating string
str="PyTHON"
#printing string
print("String:",str)
#printing first character
print("First character:",str[0])
#printing second character
print("Second character:",str[1])
#printing last character
print("Last character:",str[-1])
"""
***Output***
String: PyTHON
First character: P
Second character: y
Last character: N
"""
```

Accessing particular character or range of characters from string.
We can access range of characters from string using slicing operator colon(:).

```
#creating string
str="I love PyTHON"
#printing string
print("String:",str)
#printing indexing 2 to 6 character
print("str[2:6]:",str[2:6])
#printing character between
# 7th and 2nd last character
print("str[7:-1]:",str[7:-1])
"""
***Output***
String: I love PyTHON
str[2:6]: love
```

```
str[7:-1]: PyTHO
"""

Update String
We can update string value by reassigning new value to the same variable.
But we can,t update the particular character of the string.

#creating string
str="Python Programming"
print("Original String:",str)
#update string str
str="Java Programming"
print("Updated String:",str)
"""
***Output***
Original String: Python Programming
Updated String: Java Programming
"""
But we can,t update the particular character of the string.

#creating string
str="Python Programming"
print("Original String:",str)
#updating character P with M
#this line will raise an error
str[0]="M"
print("Updated String:",str)
"""
***Output***
TypeError: 'str' object does not support item assignment
"""
Multiline String
We can create multiline string using three double quotes.

#creating string
str="""Hello friends.
you can learn python easily.
"""
#printing string
print("Multiline String:",str)
"""
***Output***
Multiline String: Hello friends.
you can learn python easily.
"""

We can also create multiline string using three single quotes.

#creating string
str='''Hello friends.
you can learn python easily.
'''
```

```
#printing string
print("Multiline String:",str)
"""
***Output***
Multiline String: Hello friends.
you can learn python easily.
"""
```

Search String
in keyword is used to check specified character or group of characters is present or not.
in keyword returns true if specified character is present otherwise returns false.

```
#creating string
str="I love python programming"
search_str=input("Enter any string to search:")
if  search_str in str:
    print(search_str," is present")
else:
    print(search_str,"is not present")
"""
***Output***
Enter any string to search:python
python  is present
"""
```

Check specified string is not present in the string.

```
 #creating string
str="I love python programming"
print("java" not in str)
print("python" not in str)
"""
***Output***
True
False
"""
```

String Concatenation
We can combine two or more than two string using plus (+) operator.

```
#creating string
str1="I love "
str2="python programming"
str3=str1+str2
print("String1:",str1)
print("String2:",str2)
print("String3:",str3)
"""
***Output***
String1: I love
String2: python programming
```

```
String3: I love python programming
"""
```

We can't combine string with numeric value.

```
#creating string
book="xyz"
price=550
#this line will raise an error
str3="The price of book "+book+" is Rs."+price
print("Combined String:",str3)
"""
***Output***
 str3="The price of book "+book+" is Rs."+price
TypeError: can only concatenate str (not "int") to str
"""
```

To combine string with numeric value format() function is used.

```
#creating string
book="xyz"
price=550
#combining string with numeric value
str3="The price of book {} is Rs. {}".format(book,price)
print("Combined String:",str3)
"""
***Output***
String: The price of book xyz is Rs. 550
"""
```

String repetition.
asterisk (*) symbol is used to concate multiple copies of the same string.

```
str="Python ";
#print Python 5 times
print(str*5)
"""
***Output***
Python Python Python Python Python
"""
```

String Methods

Python contains the following string methods.
1.capitalize()
It converts the first letter of the string into uppercasse.

```
str="easy softwares"
print(str.capitalize())
"""
***Output***
Easy softwares
"""
```

2.casefold()
It converts string into lowercase.

```
str="Easy Softwares"
print(str.casefold())
"""
***Output***
easy softwares
"""
```

3.center()
It is used to align the string to the center.
It has two parameters width and fillchar in which fillchar is optional.

```
#Example1
str="Easy"
print("Original String:",str)
#without fillchar
print("Centered String:",str.center(20))
"""
***Output***
Original String: Easy
Centered String:         Easy
"""

#example2
str="Easy"
print("Original String:",str)
#with fillchar
#filling space with @
print("Centered String:",str.center(10,"@"))
"""
***Output***
Original String: Easy
Centered String: @@@Easy@@@
Note:total width of the output is 10
"""
```

4.endswith()
It returns boolean value(True/False).

It the given string ends with specified string returns true otherwise false.

```
str="Easy Softwares"
print(str.endswith("softwares"))
print(str.endswith("Softwares"))
"""
***Output***
False
True
"""
```

5.startswith()
It returns boolean value(True/False).
It the given string starts with specified string returns true otherwise false.

```
str="Easy Softwares"
print(str.startswith("easy"))
print(str.startswith("Easy"))
"""
***Output***
False
True
"""
```

6.find()
It is used to search the specified string in a string.
If the specified string is found then it returns the position of where it is found.

```
str="you can learn python easily."
#statrting index of python is 14
#this line will search python in string
print(str.find("python"))
#this line will search python from index 10
print(str.find("python",10))
#this line will search python from index 15
print(str.find("python",15))
#this line will search can between index 3 to 10
print(str.find("can",3,10))

"""
***Output***
14
14
-1
4
"""
```

7.index()
This method is same as find but it raises an error when the specified string is not found.

```
str="you can learn python easily."
#statrting index of python is 14
#this line will search python in string
print(str.index("python"))
#this line will search python from index 10
print(str.index("python",10))
#this line will search can between index 3 to 10
print(str.index("can",3,10))
"""
***Output***
14
14
-1
4
"""
```

8.format()
It is used to format the string.
We can insert specified value inside the string using format () method.
The specified value is inserted inside string using placeholder.
The placeholder is identified using numbered indexes {0} or empty placeholders{}.

```
#Example 1:Empty placeholder
name="Tom"
pro="Python"
print("my name is {} and i love {}".format(name,pro))
"""
***Output***
my name is Tom and i love Python
"""



#Example 2:Numbered indexes
name="Tom"
pro="Python"
print("my name is {0} and i love {1}".format(name,pro))
"""
***Output***
my name is Tom and i love Python
"""



#Example 3:Reverse indexes
#focus on output for better understanding
name="Tom"
pro="Python"
print("my name is {1} and i love {0}".format(name,pro))
"""
***Output***
my name is Python and i love Tom
"""
```

9.isalnum()
It is used to check specified string is alphanumeric or not.
String that contains only alphabet and number is called alphanumeric.
It returns boolean value(True/False).

```
str1="easy123"
str2="easy@123"
print(str1.isalnum())
print(str2.isalnum())
"""
***Output***
True
False
"""
```

10.isalpha()
It is used to check specified string is alphabetic or not.
It returns boolean value(True/False).
It returns true if string is alphabetic otherwise returns false.

```
str1="easy"
str2="easy@123"
print(str1.isalpha())
print(str2.isalpha())
"""
***Output***
True
False
"""
```

11.isdecimal()
It is used to check all the characters of string are decimal or not.
It returns boolean value(True/False).
It returns true if all characters are decimal otherwise returns false.

```
str1="easy"
str2="123"
print(str1.isdecimal())
print(str2.isdecimal())
"""
***Output***
False
True
"""
```

12.isdigit()
It is used to check all the characters of string are digit or not.
It returns boolean value(True/False).
It returns true if all characters are digit otherwise returns false.

```
str1="easy"
```

```
str2="123"
print(str1.isdigit())
print(str2.isdigit())
"""
***Output***
False
True
"""
```

13.isidentifier()
It returns true if the specified string is valid identifier otherwise returns false.

```
str1="easy" #valid
str2="123"  #invalid
str3="abc123"#valid
str4="*abc"  #invalid
str5="ab@cd" #invalid
print(str1.isidentifier())
print(str2.isidentifier())
print(str3.isidentifier())
print(str4.isidentifier())
print(str5.isidentifier())
"""
***Output***
True
False
True
False
False
"""
```

14.islower()
It returns true if all the characters of the string is in lowercase otherwise returns false.

```
str1="python"
str2="Python"
print(str1.islower())
print(str2.islower())
"""
***Output***
True
False
"""
```

15.isnumeric()
It returns true if all the characters of the string are numeric character otherwise returns false.

```
str1="python"
str2="12345"
print(str1.isnumeric())
print(str2.isnumeric())
"""
***Output***
False
True
"""
```

16.isupper()

It returns true if all the characters are in uppercase otherwise returns false.

```
str1="python"
str2="PYTHON"
print(str1.isupper())
print(str2.isupper())
"""
***Output***
False
True
"""
```

17.isspace()

It returns true if all the characters are white space otherwise returns false.

```
str1="     "
str2="PYTHON"
print(str1.isspace())
print(str2.isspace())
"""
***Output***
True
False
"""
```

18.len()

It is used to length of the string.

```
str1="Easy"
str2="Pyhton"
print("Length of str1:",len(str1))
print("Length of str2:",len(str2))
"""
***Output***
Length of str1: 4
Length of str2: 6
"""
```

19.lower()

It is used to convert all the characters of a string to Lower case.

```
str1="Easy"
str2="PyTHON"
print("str1:",str1.lower())
print("str2:",str2.lower())
"""
```

```
***Output***
str1: easy
str2: python
"""
20.upper()
It is used to convert all the characters of a string to Upper case.

str1="Easy"
str2="PyTHON"
print("str1:",str1.upper())
print("str2:",str2.upper())
"""
***Output***
str1: EASY
str2: PYTHON
"""
21.swapcase()
It converts lowercase characters into uppercase and uppercase characters
into lowercase.

str1="Easy"
str2="PyTHON"
print("str1:",str1.swapcase())
print("str2:",str2.swapcase())
"""
***Output***
str1: eASY
str2: pYthon
"""
22.strip()
It removes unwanted white-space from string.

str="    PyTHON    "
print("Without strip:",str,"Programming")
print("With strip:",str.strip(),"Programming")
"""
***Output***
Without strip:     PyTHON     Programming
With strip: PyTHON Programming
"""
23.lstrip()
It removes left side unwanted white-space from string.

str="    PyTHON    "
print("Without strip:",str,"Programming")
print("With strip:",str.lstrip(),"Programming")
"""
***Output***
Without strip:     PyTHON     Programming
With strip: PyTHON     Programming
"""
24.rstrip()
```

It removes right side unwanted white-space from string.

```
str="    PyTHON    "
print("Without strip:",str,"Programming")
print("With strip:",str.rstrip(),"Programming")
"""
***Output***
Without strip:     PyTHON     Programming
With strip:     PyTHON Programming
"""
```

25.replace()
It replaces the old string with new string.

```
str="I love Java"
print("Original String:",str)
#replacing java with python
str=str.replace("Java","Python")
print("New String:",str)
"""
***Output***
Original String: I love Java
New String: I love Python
"""
```

26.split()
It is used to break the sentenc into words using separator.
The default separator is white space.
split() function returns list.

```
str="I love Python"
print("Original String:",str)
mylist=str.split();
print("New String:",mylist)
"""
***Output***
Original String: I love Python
New String: ['I', 'love', 'Python']
Note:New string is in the form of list
"""
```

We can use any characater as a separator.

```
#asterisk as a separator
str="I*love*Python"
print("Original String:",str)
mylist=str.split("*");
print("New String:",mylist)
"""
***Output***
Original String: I love Python
New String: ['I', 'love', 'Python']
Note:New string is in the form of list
"""
```