Input and Output

Output statements

To display output or results. Python provides the print() function. This fuction can be used in python in different formats.

The print() statement

when printf() function is called simply a blank line is displayed.

```
print()
```

Output

↵
Blank line.

Print ("string") :- A string represents a group of characters.

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

Output

Hello.

we can use Escape sequence with print().

```
print (" Hello \n Delhi")
```

Output

Hello
Delhi

```
print(" Hello\t Delhi")
```

Output

Hello Delhi

In python (*) is also called repetition operator to repeat the string

for Example

```
print (3*'Hi')
HiHiHi
```

In python (+) is used to join two string as.

```
print ("city Name = " + "Delhi")
cityName = Delhi
```

print (variable list)

we can also display the values of variable using ~~print()~~ print().

as

```
a, b = 2, 4
print (a)
2
print (a, b)
2 4
```

The values in the output are seperated by space by default. To seperate the output by comma, we should used 'sep' attribute as.

```
print (a, b, sep = ",")
2,4
print (a, b, sep = ':')
2:4
print (a, b, sep = '---')
2---4
```

Note *

* The print() function throws the cursor into next line after displaying the output we can ask the print() function not to throw the cursor into the next line as.

```
print ("Hello", end = ' ')
print ("Delhi", end = ' ')
print ("How are you", end = '')
```

Output

HelloDelhiHow are you.

```
print ("Hello", end = '\t')
print ("Hello Delhi", end = '\t')
```

Hello Hello Delhi

print (object)

we can pass objects like list, tuples or dict to the print() as.

```
let = [10, 'A', 'Hi']
print (let)
```

print ("string", variable)

```
a = 2.
print (a, "is the even Number")
```

⇒ 2 is the even Number

print (formatted string)

Syntax

* print ("formatted string" % (Variable list))

Let

```
x = 10
print ('value = %d' % x)
```

→ value = 10.

```
x = 10.2
print ('value = %f' % x)
```

→ value = 10.2

```
x = 10.213
print ('value = %.1f' % x)
```

→ value = 10.2

* print ('format string with replacement'
  .format (values))

```
n1, n2, n3 = 1, 2, 3
print ('number 1 = {0}'.format(n1))
```

→ number 1 = 1

```
print ('number 1 = {0}, number 2 = {1}'
  .format (n1, n2))
```

→ number 1 = 1, number 2 = 2

As an alternative we can also use names in the replacement fields. as.

```
print ('number 1 = {one}'.format(one = n1))
```

→ number 1 = 1

x=10
print (f'value of x = {x})

⇒ value of x = 10.

Input statements :-

To accept input from keyboard, python provides the input() function. This func takes a value from keyboard and returns it as a string.

for Example

```
str = input()
Hello Delhi
print(str)
```

output

```
Hello Delhi
```

Once the value comes to variable 'str' it can be converted into 'int' or 'float' etc. as.

~~str~~ str = input("Enter a Number")

⇒ Enter a Number 125

then

```
x = int(str)  # str is converted to integer
print(x)
```

⇒ 125 Ans

You can also use it as.

```
x = int(input())
y = float(input())
```

Accepting More than One Value

```
a, b = [int(x) for x in input().split()]
```

- `int` → datatype
- `x` → Temp variable
- `input()` → Input from Keyboard.

for Example

```
# accepting 3 numbers
var1, var2, var3 = [int(x) for x in input().
                    split()]
print("sum =", var1+var2+var3)
```

eval()

The eval() function takes a string and evaluates the result of the string by taking it as a python expression. If we pass the string to the eval() functions, it will evaluate the string and return results.

```
a, b = 5,10
result = eval("a+b-4")
print(result)
```

11 Ans

we can also use eval() and input() to accept objects like list or tuples. when the user types the list using [] brackets eval() will understand that it is a list

for example

```
lst = eval(input())
```

Input

```
[10, 20, 30]
```

Output

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

```
tuple = eval(input())
```

Input

```
(10, 20, 30)
```

Output

```
tuple = (10, 20, 30)
```