Input And Output Statements

Reading dynamic input from the keyboard:

In Python 2 the following 2 functions are available to read dynamic input from the keyboard.

1. raw_input()
2. input()

1. raw_input():

This function always reads the data from the keyboard in the form of String Format. We have to convert that string type to our required type by using the corresponding type casting methods.

Eg:

```
x=raw_input("Enter First Number:")
print(type(x))    It will always print  str type only for any input type
```

2. input():

input() function can be used to read data directly in our required format.We are not required to perform type casting.

```
x=input("Enter Value)
type(x)
```

```
10 ===> int
"durga"===>str
10.5===>float
True==>bool
```

***Note: **But in Python 3 we have only input() method and raw_input() method is not available.**

Python3 input() function behaviour exactly same as raw_input() method of Python2. i.e every input value is treated as str type only.

raw_input() function of Python 2 is renamed as input() function in Python3

Eg:

```
>>> type(input("Enter value:"))
Enter value:10
<class 'str'>

Enter value:10.5
<class 'str'>

Enter value:True
<class 'str'>
```

Q. Write a program to read 2 numbers from the keyboard and print sum.

```
x=input("Enter First Number:")
y=input("Enter Second Number:")
i = int(x)
j = int(y)
print("The Sum:",i+j)

Enter First Number:100
Enter Second Number:200
The Sum: 300
```

```
x=int(input("Enter First Number:"))
y=int(input("Enter Second Number:"))
print("The Sum:",x+y)
```

```
print("The Sum:",int(input("Enter First Number:"))+int(input("Enter Second Number:")))
```

Q. Write a program to read Employee data from the keyboard and print that data.

```
eno=int(input("Enter Employee No:"))
ename=input("Enter Employee Name:")
esal=float(input("Enter Employee Salary:"))
eaddr=input("Enter Employee Address:")
married=bool(input("Employee Married ?[True|False]:"))
print("Please Confirm Information")
print("Employee No :",eno)
print("Employee Name :",ename)
print("Employee Salary :",esal)
print("Employee Address :",eaddr)
print("Employee Married ? :",married)

```

```
D:\Python_classes>py test.py
Enter Employee No:100
Enter Employee Name:Sunny
Enter Employee Salary:1000
Enter Employee Address:Mumbai
Employee Married ?[True|False]:True
Please Confirm Information
Employee No : 100
Employee Name : Sunny
Employee Salary : 1000.0
Employee Address : Mumbai
Employee Married ? : True
```

How to read multiple values from the keyboard in a single line:

```
a,b= [int(x) for x in input("Enter 2 numbers :").split()]
print("Product is :", a*b)

D:\Python_classes>py test.py
Enter 2 numbers :10 20
Product is : 200
```

Note: split() function can take space as seperator by default .But we can pass anything as seperator.

Q. Write a program to read 3 float numbers from the keyboard with , seperator and print their sum.

```
a,b,c= [float(x) for x in input("Enter 3 float numbers :").split(',')]
print("The Sum is :", a+b+c)

D:\Python_classes>py test.py
Enter 3 float numbers :10.5,20.6,20.1
The Sum is : 51.2
```

eval():

eval Function take a String and evaluate the Result.

Eg: x = eval("10+20+30")
print(x)

Output: 60

Eg: x = eval(input("Enter Expression"))
Enter Expression: 10+2*3/4
Output11.5

eval() can evaluate the Input to list, tuple, set, etc based the provided Input.

Eg: Write a Program to accept list from the keynboard on the display

```
1) l = eval(input("Enter List"))
2) print (type(l))
3) print(l)
```

Command Line Arguments

- argv is not Array it is a List. It is available sys Module.
- The Argument which are passing at the time of execution are called Command Line Arguments.

Eg: D:\Python_classes py test.py 10 20 30

Command Line Arguments

Within the Python Program this Command Line Arguments are available in argv. Which is present in SYS Module.

test.py	10	20	30

Note: argv[0] represents Name of Program. But not first Command Line Argument.
argv[1] represent First Command Line Argument.

Program: To check type of argv from sys

```
import argv
print(type(argv))
```

D:\Python_classes\py test.py

Write a Program to display Command Line Arguments

```
1) from sys import argv
2) print("The Number of Command Line Arguments:", len(argv))
3) print("The List of Command Line Arguments:", argv)
4) print("Command Line Arguments one by one:")
5) for x in argv:
6)     print(x)
7)
8) D:\Python_classes>py test.py 10 20 30
9) The Number of Command Line Arguments: 4
```

```
The List of Command Line Arguments: ['test.py', '10','20','30']
Command Line Arguments one by one:
test.py
10
20
30
```

```
from sys import argv
sum=0
args=argv[1:]
for x in args   :
   n=int(x)
   sum=sum+n
print("The Sum:",sum)

D:\Python_classes>py test.py 10 20 30 40
The Sum: 100
```

Note1: usually space is seperator between command line arguments. If our command line argument itself contains space then we should enclose within double quotes(but not single quotes)

Eg:

```
from sys import argv
print(argv[1])

D:\Python_classes>py test.py Sunny Leone
Sunny

D:\Python_classes>py test.py 'Sunny Leone'
'Sunny

D:\Python_classes>py test.py "Sunny Leone"
Sunny Leone
```

Note2: Within the Python program command line arguments are available in the String form. Based on our requirement,we can convert into corresponding type by using type casting methods.

Eg:

```
from sys import argv
print(argv[1]+argv[2])
print(int(argv[1])+int(argv[2]))
```

```
4)
5)  D:\Python_classes>py test.py 10 20
6)  1020
7)  30
```

Note3: If we are trying to access command line arguments with out of range index then we will get Error.

Eg:

```
1)  from sys import argv
2)  print(argv[100])
3)
4)  D:\Python_classes>py test.py 10 20
5)  IndexError: list index out of range
```

Note:

In Python there is argparse module to parse command line arguments and display some help messages whenever end user enters wrong input.

input()
raw_input()

command line arguments

output statements:

We can use print() function to display output.

Form-1: print() without any argument
Just it prints new line character

Form-2:

```
1)  print(String):
2)  print("Hello World")
3)  We can use escape characters also
4)  print("Hello \n World")
5)  print("Hello\tWorld")
6)  We can use repetetion operator (*) in the string
7)  print(10*"Hello")
8)  print("Hello"*10)
9)  We can use + operator also
10) print("Hello"+"World")
```

Note:

If both arguments are String type then + operator acts as concatenation operator.

If one argument is string type and second is any other type like int then we will get Error

If both arguments are number type then + operator acts as arithmetic addition operator.

Note:

```
1) print("Hello"+"World")
2) print("Hello","World")
3)
4) HelloWorld
5) Hello World
```

Form-3: print() with variable number of arguments:

```
1. a,b,c=10,20,30
2. print("The Values are :",a,b,c)
3.
4. OutputThe Values are : 10 20 30
```

By default output values are seperated by space.If we want we can specify seperator by using "sep" attribute

```
1. a,b,c=10,20,30
2. print(a,b,c,sep=',')
3. print(a,b,c,sep=':')
4.
5. D:\Python_classes>py test.py
6. 10,20,30
7. 10:20:30
```

Form-4:print() with end attribute:

```
1. print("Hello")
2. print("Durga")
3. print("Soft")
```

Output:

```
1. Hello
2. Durga
3. Soft
```

If we want output in the same line with space

```
print("Hello",end=' ')
print("Durga",end=' ')
print("Soft")
```

Output: Hello Durga Soft

Note: The default value for end attribute is \n,which is nothing but new line character.

Form-5: print(object) statement:

We can pass any object (like list,tuple,set etc)as argument to the print() statement.

Eg:

```
l=[10,20,30,40]
t=(10,20,30,40)
print(l)
print(t)
```

Form-6: print(String,variable list):

We can use print() statement with String and any number of arguments.

Eg:

```
s="Durga"
a=48
s1="java"
s2="Python"
print("Hello",s,"Your Age is",a)
print("You are teaching",s1,"and",s2)
```

Output:

```
1) Hello Durga Your Age is 48
2) You are teaching java and Python
```

Form-7: print(formatted string):

%i====>int
%d====>int
%f=====>float
%s======>String type

Syntax:

print("formatted string" %(variable list))

Eg 1:

```
1)  a=10
2)  b=20
3)  c=30
4)  print("a value is %i" %a)
5)  print("b  value is %d and c value is %d" %(b,c))
6)
7)  Output
8)  a value is 10
9)  b  value is 20 and c value is 30
```

Eg 2:

```
1)  s="Durga"
2)  list=[10,20,30,40]
3)  print("Hello %s ...The List of Items are %s"  %(s,list))
4)
5)  Output Hello Durga ...The List of Items are [10, 20, 30, 40]
```

Form-8: print() with replacement operator {}

Eg:

```
1)  name="Durga"
2)  salary=10000
3)  gf="Sunny"
4)  print("Hello {0} your salary is {1} and Your Friend {2} is waiting".format(name,salary,gf))
5)  print("Hello {x} your salary is {y} and Your Friend {z} is waiting".format(x=name,y=salary,z=
    gf))
6)
7)  Output
8)  Hello Durga your salary is 10000 and Your Friend Sunny is waiting
9)  Hello Durga your salary is 10000 and Your Friend Sunny is waiting
```