Python Programming

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

Academic Year : 2020-2021

INPUT-OUTPUT STATEMENTS - 3

Agenda:

1. **Output Statements : print() function**
2. **'sep' attribute**
3. **'end' attribute**
4. **Replacement Operator ({})**

Guido Van Rossum

Learning Mantra

If you really strong in the basics, then remaining things will become so easy.

4. Output Statements : 'print()' function

- We can use print() function to display output to the console for end user sake.
- Multiple forms are there related to print() function.

Form-1:print() without any argument

- Just it prints new line character (i.e.,\n)

Eg:

```
print('karthi')
print()                    # prints new line character
print('sahasra')
```

Output:

```
karthi

sahasra
```

see the difference in below code:

```
print('karthi')

#print()

print('sahasra')
```

Output:

```
karthi
sahasra
```

Form-2: print() function to print of string argument

Eg:

print("Hello World") ➔Hello World

- We can use escape characters also.

Eg:

```
print("Hello \nWorld")
print("Hello\tWorld")
```

Output:

```
Hello
World
Hello	World
```

- **We can use repetition operator (*) in the string.**

Eg:

```
print(10*"Hello")
print("Hello"*10)
```

Output:

```
HelloHelloHelloHelloHelloHelloHelloHelloHelloHello
HelloHelloHelloHelloHelloHelloHelloHelloHelloHello
```

- **We can use + operator also.**

Eg:

```
print("Hello"+"World")          ➔HelloWorld
```

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.

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

Eg:

```
a,b,c=10,20,30
print("The Values are :",a,b,c) # here, we are passing 4 arguments to the print function.
```

Output:

```
The Values are : 10 20 30
```

5. 'sep' attribute:

Form-4: print() with 'sep' attribute:

- By default output values are separated by space. If we want we can specify separator by using "**sep**" attribute.
- '**sep**' means separator.

Eg:

```
a,b,c=10,20,30
print(a,b,c)            ➔ 10 20 30
print(a,b,c,sep=',')    ➔ 10,20,30
print(a,b,c,sep=':')    ➔ 10:20:30
print(a,b,c,sep='-')    ➔ 10-20-30
```

6. 'end' attribute:

Form-5: print() with 'end' attribute:

Eg:

```
print("Hello")
print("Karthi")
print("Sahasra")
```

Output:

```
Hello
Karthi
Sahasra
```

- If we want output in the same line with space, we need to use end attribute.
- default value of 'end' attribute is newline character. (That means, if there is no end attribute, automatically newline character will be printed).

Eg:

```
print("Hello",end=' ')
print("Karthi",end=' ')       # if end is space character
print("Sahasra")
```

Output:

```
Hello Karthi Sahasra
```

Eg: Program to demonstrate both 'sep' and 'end' attributes.

```
print(10,20,30,sep = ':', end = '***')
print(40,50,60,sep = ':') # default value of 'end' attribute is '\n'
print(70,80,sep = '**',end = '$$')
print(90,100)
```

Output:

```
10:20:30***40:50:60
70**80$$90 100
```

Eg:

```
print('karthi' + 'sahasra')        ➔karthisahasra
print('karthi','sahasra')          ➔karthi sahasra
print(10,20,30)                    ➔10 20 30
```

Form-6: 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)          ➔[10, 20, 30, 40]
print(t)          ➔(10, 20, 30, 40)
```

Form-7: print(String, variable list):

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

Eg:

```
s="Karthi"
a=6
s1="java"
s2="Python"
print("Hello",s,"Your Age is",a)              ➔Hello Karthi Your Age is 6
print("You are learning",s1,"and",s2)         ➔You are learning java and Python
```

8.Printing formatted string

Form-8: print(formatted string)

%i ====>int

%d ====>int

%f =====>float

%s ======>String type

Syntax:

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

Eg:

```
a=10
b=20
c=30
print("a value is %i" %a)
print("b value is %d and c value is %d" %(b,c))
```

Output:

```
a value is 10
b value is 20 and c value is 30
```

Eg:

```
s="Karthi"
list=[10,20,30,40]
print("Hello %s ...The List of Items are %s" %(s,list))
```

Output:

```
Hello Karthi ...The List of Items are [10, 20, 30, 40]
```

9. Replacement Operator ({ }):

Form-9: print() with replacement operator { }

Eg:

```
a,b,c,d = 10,20,30,40            # print a=10,b=20,c=30,d=40
print('a = {},b = {},c = {},d = {}'.format(a,b,c,d))
```

Output:

```
a = 10,b = 20,c = 30,d = 40
```

Eg:

```
name = "Karthi"
salary = 100000
sister = "Sahasra"
print("Hello {} your salary is {} and Your Sister {} is waiting".format(name,salary,sister))
print("Hello {0} your salary is {1} and Your Sister {2} is waiting".format(name,salary,sister))
print("Hello {1} your salary is {2} and Your Sister {0} is waiting".format(name,salary,sister))
print("Hello {2} your salary is {0} and Your Sister {1} is waiting".format(salary,sister,name))
print("Hello {x} your salary is {y} and Your Sister {z} is waiting".format(x=name,y=salary,z=sister))
```

Output:

```
Hello Karthi your salary is 100000 and Your Sister Sahasra is waiting
Hello Karthi your salary is 100000 and Your Sister Sahasra is waiting
Hello 100000 your salary is Sahasra and Your Sister Karthi is waiting
Hello Karthi your salary is 100000 and Your Sister Sahasra is waiting
Hello Karthi your salary is 100000 and Your Sister Sahasra is waiting
```

Eg:

```
price = 70.56789
print('Price value = {}'.format(price))        ➔Price value = 70.56789
print('Price value = %f'%price)                ➔Price value = 70.567890
print('Price value = %.2f'%price)              ➔Price value = 70.57
```

Note:

- We will display only two digits after decimal point, this type of customization is possible with formatted strings only.

Any question?

If you try to practice programs yourself, then you will learn many things automatically

Spend few minutes and then enjoy the study

Thank You