Chapter-5: Python Input and Output Statements

Input and Output Statements in Python

Programming elements are **three**:

1. Input
2. Process
3. Output

Input

Input is nothing but **reading information or data from different sources**.
These sources can be **keyboard, OCR, OMR, scanner, database, file**, etc.

Input is nothing but **giving data or information to a program**.

Process

Process is nothing but **performing operations on input data**.

Output

Output is nothing but **processed data or result**.
A program **writes or prints data**.

`print()` Statement or Function

`print()` is an **in-built function** in Python.
This function is used to **print data or information on the console** or **standard output device (monitor)**.

`print()` is a **standard output function**.
The values given to the function are called **arguments**.

`print()` function is used to print **one or more values**.

Arguments of `print()` Function

The `print()` function has **5 arguments**:

1. `values`
2. `sep`
3. `end`
4. `file`
5. `Flush`

`Example:`

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

a = 100
print(a)

x = 10
y = 20
z = 30

print(x, y, z)
print(x, y, z, sep=",")
print(x, y, z, sep="*")
print(x, y, z)
```

print(x, sep=":")
Output:
Hello
10
100
10 20 30
10,20,30
10*20*30
10 20 30
10

str Datatype and String Literal / Value

What is String?

A string is a collection of characters.
These characters can be alphabets, digits, or special characters.

In Python, string value / literal is represented in memory using the str data type.

Representation of String Literals in Python
In Python, string values / literals are represented in three ways:

Within single quotes (' ')

Within double quotes (" ")

Within triple single quotes (''' ''') or triple double quotes (""" """)

Within Single Quotes

Within single quotes, we can represent a **single-line string**.
Within single quotes, we can **embed or insert only double quotes**.

Example:

```
name = 'naresh'
print(name, type(name))

address = 'opp:AAA'
print(address)

emailid = 'naresh123@nareshit.com'
print(emailid, type(emailid))

str1 = 'python is "easy" language'
print(str1)
```

Output:

```
naresh <class 'str'>
opp:AAA
naresh123@nareshit.com <class 'str'>
python is "easy" language
```

Within Double Quotes

Within double quotes, we can represent a **single-line string**.
Within double quotes, we can **insert or embed single quotes**.

Example:

```
rollno = 1
name = "naresh"
course = "full stack python"
fee = 5000.0

print(rollno, name, course, fee)

str1 = "python 'easy' language"
print(str1)
```

```
str2 = "python programming language"
print(str2)
```

Output:

```
1 naresh full stack python 5000.0
python 'easy' language
python programming language
```

Triple single quotes or double quotes

Triple single quotes or double quotes are used for representing multiline strings.

```
>>> str1="""Python
programming
language"""
>>> print(str1)
Python
programming
language

>>> address="""Naresh IT
... Ameerpet
... Hyd"""
>>> print(address)
Naresh IT
Ameerpet
Hyd

>>> str2="""python programming"""
>>> print(str2)
```

```
python programming

>>> str3="""python programming"
... language"""
>>> print(str3)
python programming"
language
```

Escape Sequence

Escape sequences also called backslash character values/literals

Escape Sequence	Description
\n	New line
\t	Horizontal tab space
\v	Vertical tab space
\\	Backslash
\”	“
\’	‘
\b	Backspace
\r	Carriage Return
\a	Beep Sound

Python Escape Sequence Examples

```
str1="python \"programming\" language"
print(str1)

str2='python \'programming\' language'
print(str2)

str3="""full
stack
python"""
print(str3)
```

```
str4='full\nstack\npython'
print(str4)

str5="python\nprogramming\nlanguage"
print(str5)

print("\\")
print("C:\\folder1\\file1.txt")

print("Rollno\tName\tCourse")
print("1\tNaresh\tPython")
print("2\tSuresh\tJava")
print("3\tKishore\tC++")

print("a\vb")
print("python\bx")
print("\a\a\a\a\a")
```

Output

```
python "programming" language
python 'programming' language
full
stack
python
full
stack
python
python
programming
language
```

```
\
C:\folder1\file1.txt
Rollno  Name    Course
1       Naresh Python
2       Suresh Java
3       Kishore C++
a
b
Pythonx
```

Note: python does not support single character data type. In python a single character also represented using str data type.

Example:

```
s1="25"
s2="1.5"
s3="1+2"
s4="True"

print(type(s1), type(s2), type(s3), type(s4))

a="10"
b="5"

name="naresh"
rollno="101"
courseid="py123"
emailid="nareshit123@nareshit.com"

print(name, rollno, courseid, emailid)
```

Output

```
<class 'str'> <class 'str'> <class 'str'> <class
'str'>
naresh 101 py123 nareshit123@nareshit.com
```

String Types

The string which consist of only alphabets is called **alphabetic string**.

This string which consist of alphabets, digits is called **alphanumeric string**.

We cannot perform arithmetic operations on string.

Raw String

This string which is prefix with **r** or **R** is called raw string.

Syntax

```
r'string'
r"string"
r'''string'''
R"""string"""
```

Example

```
print("rollno\tname\tcourse")
print(r"rollno\tname\tcourse")

print("python\nprogramming\nlanguage")
print(r"python\nprogramming\nlanguage")
```

Output

```
rollno   name    course
rollno\tname\tcourse
```

```
python
programming
language
python\nprogramming\nlanguage
```

Note:
Raw string does not interpret escape sequences (characters).

f-string (OR) format string

The string prefix with **f** or **F** is called format string.
Format string is used for formatting output by inserting values.
This string is introduced in **Python 3.6** version.

Example

```
a = 10

b = 20

c = a + b


print("sum of", a, "and", b, "is", c)

print(f"sum of {a} and {b} is {c}")

print(f"{c} is sum of {a},{b}")
```

Output

```
sum of 10 and 20 is 30

sum of 10 and 20 is 30

30 is sum of 10,20
```

Inserting values within the string is done using **{} (curly braces) / replacement fields**.

```
{variable-name}

{expression}
```

Example

```
# print function print multiple values using sep

# default separator used by print is space

# sep argument can be used to change separator
```

```
rollno = 1
name = "naresh"
course = "python"
fee = 9000

print(rollno, name, course, fee)
print(rollno, name, course, fee, sep=",")
print(rollno, name, course, fee, sep="\t")
print(rollno, name, course, fee, sep="\n")
print(rollno, name, course, fee, sep="*")
```

Output

```
1 naresh python 9000
1,naresh,python,9000
1   naresh   python   9000
1
naresh
python
9000
1*naresh*python*9000
```

Example (end argument in print)

```
# print function uses end argument to insert value at
# the end of printing values
# default value of end is \n

print("PYTHON")
print("JAVA", end='.')
print("ORACLE")
print("PHP", end=';')
print("MONGODB", end=':')
print("AI")
```

Output

```
PYTHON
JAVA.ORACLE
PHP;MONGODB:AI
```

Example

```
print()
print()
print("A")
print()
print("B")
print()
```

Output

```
A

B
```

Comments

In Python, comments are represented using #.

Any line beginning with # is called a comment, which is not executed by the Python translator.

Python supports only single-line comments.

input() function

input() is an in-built function in python. This function is used to input values during runtime.

Input() function is used to input value of type string.
Input() function allows to input/read a single value

Syntax:

variable-name=input("prompt")

Example:

```
#Login Application

user=input("USERNAME :")
password=input("PASSWORD :")

print(user,type(user))
print(password,type(password))
```

Output

```
USERNAME :nit123
PASSWORD :nit
nit123 <class 'str'>
nit <class 'str'>
```

Example:

```
# Read Student Details and Print

rollno=input("Rollno :")
name=input("StudentName :")
course=input("StudentCourse :")

print(f'Rollno {rollno}')
print(f'StudentName {name}')
print(f'Student Course {course}')
```

Output

```
Rollno :123
StudentName :Naresh
StudentCourse :Java
Rollno 123
StudentName Naresh
Student Course Java
```

 # adding two numbers

```
num1=input("Enter First Number ")
num2=input("Enter Second Number ")

num3=num1+num2

print(f'Sum of {num1} and {num2} is {num3}')
```

Output

```
Enter First Number 25
Enter Second Number 99
Sum of 25 and 99 is 2599
```

We cannot perform arithmetic operations on strings.