Input/Output

❖ How to Take Input from User in Python

Sometimes a developer might want to take user input at some point in the program. To do this Python provides an input() function.

Syntax:

```
input('prompt')
```

where prompt is an optional string that is displayed on the string at the time of taking input.

Example:

```
# Taking input from the user
name = input("Enter your name: ")


# Output
print("Hello, " + name)
print(type(name))
```

Output:

```
Enter your name: Topper World
Hello, Topper World
<class 'str'>
```

❖ How to take Multiple Inputs in Python :

we can take multiple inputs of the same data type at a time in python, using map() method in python.

Example:

```
a, b, c = map(int, input("Enter the Numbers : ").split())
print("The Numbers are : ",end = " ")
print(a, b, c)
```

Output:

```
Enter the Numbers : 2 3 4
The Numbers are :  2 3 4
```

❖ How take inputs for the Sequence Data Types

In the case of List and Set the input can be taken from the user in two ways.

1. Taking List/Set elements one by one by using the append()/add() methods.
2. Using map() and list() / set() methods.

➢ Taking List/Set elements one by one

Take the elements of the List/Set one by one and use the append() method in the case of List, and add() method in the case of a Set, to add the elements to the List / Set.

Example:

```
List = list()
Set = set()
l = int(input("Enter the size of the List : "))
s = int(input("Enter the size of the Set : "))
print("Enter the List elements : ")
for i in range(0, l):
```

```
        List.append(int(input()))
print("Enter the Set elements : ")
for i in range(0, s):
        Set.add(int(input()))
print(List)
print(Set)
```

Output:

```
Enter the size of the List : 4
Enter the size of the Set  : 3
Enter the List elements :
9
0
1
3
Enter the Set elements :
2
9
1
[9, 0, 1, 3]
{9, 2, 1}
```

❖ How to Display Output in Python

Python provides the print() function to display output to the standard output devices.

Syntax:

```
print(value(s), sep= ' ', end = '\n', file=file, flush=flush)
```

➢ Parameters:

- **value(s) :** Any value, and as many as you like. Will be converted to string before printed
- **sep='separator' :** (Optional) Specify how to separate the objects, if there is more than one.Default :' '
- **end='end':** (Optional) Specify what to print at the end.Default : '\n'
- **file :** (Optional) An object with a write method. Default :sys.stdout
- **flush :** (Optional) A Boolean, specifying if the output is flushed (True) or buffered (False). Default: False

Example:

```
# Python program to demonstrate

# print() method

print("Topper World")
```

Output:

```
Topper World
```

❖ Formatting Output

Formatting output in Python can be done in many ways. Let's discuss them below:

➢ Using formatted string literals

We can use formatted string literals, by starting a string with f or F before opening quotation marks or triple quotation marks. In this string, we can write Python expressions between **{** and **}** that can refer to a variable or any literal value.

Example:

```
# Declaring a variable

name = "Topper World"


# Output

print(f'Hello {name}! How are you?')
```

Output:

```
Hello Topper World! How are you?
```

➢ Using format()

We can also_use format() function to format our output to make it look presentable. The curly braces **{ }** work as placeholders. We can specify the order in which variables occur in the output.

➢ Using % Operator

We can use '%' operator. % values are replaced with zero or more value of elements. The formatting using % is similar to that of 'printf' in the C programming language.

- %d – integer
- %f – float
- %s – string
- %x – hexadecimal
- %o – octal

Example:

```
# Taking input from the user
num = int(input("Enter a value: "))

add = num + 5

# Output
print("The sum is %d" %add)
```

Output:

```
Enter a value: 50
The sum is 55
```