Input and Output in Python

Understanding input and output operations is fundamental to Python programming. With the print() function, we can display output in various formats, while the input() function enables interaction with users by gathering input during program execution.

Taking input in Python

Python's **input() function** is used to take user input. By default, it returns the user input in form of a string.

Example:

```
name = input("Enter your name: ")
print("Hello,", name, "! Welcome!")
```

Output

Enter your name: GeeksforGeeks
Hello, GeeksforGeeks ! Welcome!

The code prompts the user to input their name, stores it in the variable "name" and then prints a greeting message addressing the user by their entered name.

To learn more about taking input, please refer: Taking Input in Python

Printing Output using print() in Python

At its core, printing output in Python is straightforward, thanks to the print() function. This function allows us to display text, variables and expressions on the console. Let's begin with the basic usage of the print() function:

In this example, "Hello, World!" is a string literal enclosed within double quotes. When executed, this statement will output the text to the console.

```
print("Hello, World!")
```

Output

```
Hello, World!
```

Printing Variables

We can use the print() function to print single and multiple variables. We can print multiple variables by separating them with commas. **Example:**

```
# Single variable
s = "Bob"
print(s)

# Multiple Variables
s = "Alice"
age = 25
city = "New York"
print(s, age, city)
```

Output

```
Bob
Alice 25 New York
```

Take Multiple Input in Python

We are taking multiple input from the user in a single line, splitting the values entered by the user into separate variables for each value using the split() method. Then, it prints the values with corresponding labels, either two or three, based on the number of inputs provided by the user.

```
# taking two inputs at a time
x, y = input("Enter two values: ").split()
print("Number of boys: ", x)
print("Number of girls: ", y)

# taking three inputs at a time
x, y, z = input("Enter three values: ").split()
print("Total number of students: ", x)
print("Number of boys is : ", y)
print("Number of girls is : ", z)
```

Output

```
Enter two values: 5 10
Number of boys:  5
 Number of girls:  10
Enter three values: 5 10 15
Total number of students:  5
Number of boys is :  10
 Number of girls is :  15
```

How to Change the Type of Input in Python

By default input() function helps in taking user input as string. If any user wants to take input as int or float, we just need to typecast it.

Print Names in Python

The code prompts the user to input a string (the color of a rose), assigns it to the variable color and then prints the inputted color.

```
# Taking input as string
color = input("What color is rose?: ")
print(color)
```

Output

```
What color is rose?: Red
Red
```

Print Numbers in Python

The code prompts the user to input an integer representing the number of roses, converts the input to an integer using typecasting and then prints the integer value.

```
# Taking input as int
# Typecasting to int
n = int(input("How many roses?: "))
print(n)
```

Output

How many roses?: 88

Print Float/Decimal Number in Python

The code prompts the user to input the price of each rose as a floating-point number, converts the input to a float using typecasting and then prints the price.

```
# Taking input as float
# Typecasting to float
price = float(input("Price of each rose?: "))
print(price)
```

Output

Price of each rose?: 50.3050.3

Find DataType of Input in Python

In the given example, we are printing the type of variable x. We will determine the type of an object in Python.

```
a = "Hello World"
b = 10
c = 11.22
d = ("Geeks", "for", "Geeks")
e = ["Geeks", "for", "Geeks"]
```

```
f = {"Geeks": 1, "for":2, "Geeks":3}


print(type(a))
print(type(b))
print(type(c))
print(type(d))
print(type(e))
print(type(f))
```

Output

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'tuple'>
<class 'list'>
<class 'dict'>
```

Output Formatting

Output formatting in Python with various techniques including the format() method, manipulation of the sep and end parameters, f-strings and the versatile % operator. These methods enable precise control over how data is displayed, enhancing the readability and effectiveness of your Python programs.

Example 1: Using Format()

```
amount = 150.75
```

```
print("Amount: ${:.2f}".format(amount))
```

Output

```
Amount: $150.75
```

Example 2: Using sep and end parameter

```
# end Parameter with '@'
print("Python", end='@')
print("GeeksforGeeks")

# Seprating with Comma
print('G', 'F', 'G', sep='')

# for formatting a date
print('09', '12', '2016', sep='-')

# another example
print('pratik', 'geeksforgeeks', sep='@')
```

Output

```
Python@GeeksforGeeks
GFG
09-12-2016
pratik@geeksforgeeks
```

Example 3: Using f-string

```
name = 'Tushar'
age = 23
print(f"Hello, My name is {name} and I'm {age} years old.")
```

Output

```
Hello, My name is Tushar and I'm 23 years old.
```

Example 4: 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.

```
# 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: 50The sum is 55