Input/Output in Python

1. Introduction:

Input and Output (I/O) operations are fundamental for any programming language. In Python, I/O allows programs to interact with users or external sources such as files. This section explains how to accept input from users and how to display output to the screen.

2. Importance of Input/Output:

- User interaction: Allows programs to take user-provided values.
- Data processing: Collects input, processes it, and provides output.
- File handling: Reads from or writes data to files for storage.

3. The print() Function - Output in Python:

The `print()` function is used to display output to the console.

Syntax:

```
print(object, ..., sep=' ', end='\n')
```

Example:

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

Customizing output:

```
print("Hello", "Python", sep=" - ", end=" END\n")
# Output: Hello - Python END
```

4. The input() Function - Taking User Input:

The `input()` function is used to accept data from the user as a string.

Example:

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

Important: input() always returns a string. If a number is expected, use type conversion:

```
age = int(input("Enter your age: "))
```

```
pi = float(input("Enter a decimal: "))
```

5. Reading Multiple Inputs:

You can read multiple values in one line by using split():

```
x, y = input("Enter two numbers: ").split()
print("Sum:", int(x) + int(y))
```

6. Type Conversion:

Input data may need to be converted to int, float, etc.

Example:

```
num = int(input("Enter a number: "))
print(num * 2)
```

If conversion fails, Python throws a ValueError. Always validate input for robustness.

7. Formatting Output:

Python supports f-strings for formatted output:

```
name = "Alice"
age = 25
print(f"{name} is {age} years old.")
```

Alternative: format() method

```
print("{} is {} years old".format(name, age))
```

8. Common Pitfalls:

- Forgetting to convert input to number
- Unexpected white space or split errors
- TypeError due to incorrect use of print()

9. Best Practices:

- Always validate user input
- Use meaningful prompts in input()

- Use f-strings for clarity in formatting

10. Practice Questions:

1. Write a program to input your name and greet the user.
2. Ask the user for two numbers and print their product.
3. Take a users birth year and calculate their age.
4. Read three space-separated strings and print them in reverse order.
5. Create a simple calculator that adds two numbers provided by the user.

11. Summary:

This section covered how to take input from users and display output effectively. Python's print() and input() functions are simple yet powerful tools for building interactive applications.