Inputs and Outputs in Python

Overview

Python makes it simple to interact with users or external services using:

- **`input()`** : Collects user input or data from other sources.
- **`print()`** : Outputs information to the screen or a file.

The `input()` Function

- The **`input()`** function is used to collect data from the user through the keyboard or console.
- The data entered is always returned as a **string**.

Basic Example

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

Prompting the User

- You can display a message or a question inside the `input()` function to guide the user.

```
age = input("How old are you? ")
print("You are " + age + " years old.")
```

Converting Input Data

Since **`input()`** returns data as a string, you often need to convert it into another data type for calculations or logical operations.

- **Convert to Integer:**

```
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
print("The sum is:", num1 + num2)
```

- **Convert to Float:**

```
price = float(input("Enter the price of the item: "))
print("The price with tax is:", price * 1.08)
```

The `print()` Function

- The **`print()`** function is used to display output in Python.
- It supports:
  - Printing multiple objects.
  - String concatenation.
  - Formatting output.

Basic Example

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

Printing Multiple Objects

Use **commas** to separate objects, and Python will print them with spaces by default.

```
a, b, c = 1, 2, 3
print(a, b, c)  # Output: 1 2 3
```

Custom Separators (`sep`)

The `sep` parameter defines the separator between printed objects.

```
print("apple", "banana", "cherry", sep=", ")
# Output: apple, banana, cherry
```

Custom End (`end`)

The `end` parameter defines what is printed at the end of the output (default is a newline `\n`).

```
print("Hello", end=" ")
print("World!")
# Output: Hello World!
```

String Concatenation with `print()`

- Use `+` to join strings:

```
first_name = "Tom"
last_name = "Jones"
print("Hello, " + first_name + " " + last_name)
# Output: Hello, Tom Jones
```

String Formatting

Python provides multiple ways to format strings dynamically.

Using `format()`

```
name = "Alice"
age = 30
print("Hello, {}. You are {} years old.".format(name, age))
# Output: Hello, Alice. You are 30 years old.
```

Using f-strings (Python 3.6+)

```
name = "Alice"
age = 30
print(f"Hello, {name}. You are {age} years old.")
# Output: Hello, Alice. You are 30 years old.
```

Practical Examples

Collecting and Printing Input

```
first_name = input("Enter your first name: ")
last_name = input("Enter your last name: ")
print(f"Hello, {first_name} {last_name}!")
```

Performing Arithmetic

```
num1 = int(input("Enter a number: "))
num2 = int(input("Enter another number: "))
print(f"The sum is {num1 + num2}")
```

Customizing Print Output

```
print("Python", "is", "fun!", sep="-", end=" 😊\n")
# Output: Python-is-fun! 😊
```