Python String Formatting

UNDERSTANDING STRING MANIPULATION IN PYTHON

Introduction to String Formatting

WHAT IS STRING FORMATTING?

- String formatting in Python allows you to insert variables and expressions into strings.
- Useful for generating dynamic text.

WHY IS IT IMPORTANT?

- Improves code readability.
- Simplifies the process of constructing complex strings.

Basic String Concatenation (Review)

CONCATENATION EXAMPLE

name = "Alice“

greeting = "Hello, " + name + "!“

Output: Hello, Alice!

LIMITATIONS OF CONCATENATION

- Can become messy with multiple variables.
- Harder to read and maintain.

Introduction to String Formatting Methods

- Three Common Methods:

1) % Operator (Old style)

2) str.format() Method (Newer style)

3) f-Strings (Newest and most efficient)

% Operator for String Formatting

- Basic Syntax:

  "Hello, %s!" % name

- Example with Multiple Variables:

  age = 21

  print(“My name is %s and I am %d years old." % (name, age))

Output: My name is Alice and I am 21 years old.

- %s : string, %d : int, %f : float
- Pros and Cons:

Pros: Simple for basic tasks.

Cons: Less readable with many variables; not as flexible.

str.format() Method

Basic Syntax:

- "Hello, {}!".format(name)

Positional and Keyword Arguments:

- Positional: "{}, you have {} new messages.".format(name, count)
- Keyword: "{name}, you have {count} new messages.".format(name="Alice", count=5)

Example:

- "My name is {} and I am {} years old.".format(name, age)

- Output: My name is Alice and I am 21 years old.

Advantages:

- More flexible than the % operator.
- Easier to read and maintain.

f-Strings (Formatted String Literals)

- **Basic Syntax :** f"Hello, {name}!“
- **Example with Expressions:**

  f"My name is {name} and I am {age} years old."
  Output: My name is Alice and I am 21 years old.

- **Advanced Example:**

  f"In 10 years, I will be {age + 10} years old."
  Output: In 10 years, I will be 31 years old.

- **Benefits of f-Strings:**

  Most concise and readable.
  Supports expressions directly inside the curly braces.

Formatting Numbers and Strings

- **Controlling Decimal Places:**
  - Example: f"Pi is approximately {3.14159:.2f}"
  - **Output:** Pi is approximately 3.14

Padding and Alignment:

- Left-align: f"{name:<10}"
- Right-align: f"{name:>10}"
- Center-align: f"{name:^10}"

Comparison of Methods

SIDE-BY-SIDE EXAMPLE

- % Operator:

"My name is %s and I am %d." % (name, age)

str.format():

"My name is {} and I am {}.".format(name, age)

f-Strings:

f"My name is {name} and I am {age}."

WHICH ONE TO USE?

- % Operator: Legacy code.
- str.format(): Versatile for Python 3.5 and earlier.
- f-Strings: Recommended for Python 3.6 and later.

Common Mistakes and Tips

- **Common Errors:**

  - Mismatched placeholders and variables.
  - Incorrectly formatted f-Strings.

- **Tips:**

  - Always double-check placeholder types.
  - Use f-Strings for clarity and conciseness.

Practice Exercise

- **Task1:**

Create a string that says, "John has 3 apples and 2 oranges.“

- **Task2:**

Use an f-string to create a sentence that says: "The result of 8 multiplied by 7 is 56.“

- Task3:

Given a = 5 and b = 3, use an f-string to create a sentence that says: "The sum of 5 and 3 is 8, and their product is 15."

THANK YOU

HAPPY LEARNING!