**Format Operators in Python**

In Python, **format operators** are used to insert values into strings in a formatted way. Python provides different approaches for formatting strings:

**1. % Formatting (Old Style)**

The % operator, also known as **printf-style formatting**, is an older method for formatting strings.

**Syntax:**

"format\_string" % (values)

**Example:**

name = "Alice"

age = 25

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

**Output:**

My name is Alice and I am 25 years old.

**Common Format Specifiers:**

| **Specifier** | **Description** |
| --- | --- |
| %s | String |
| %d | Integer |
| %f | Float |
| %.2f | Float (2 decimal places) |
| %x | Hexadecimal (lowercase) |
| %X | Hexadecimal (uppercase) |

**Example with Floating Point Precision:**

pi = 3.14159

print("Pi value: %.2f" % pi) # Output: Pi value: 3.14

**2. .format() Method (New Style)**

Introduced in Python 2.6 and widely used in Python 3, the .format() method provides more flexibility.

**Syntax:**

"{}".format(values)

**Example:**

name = "Bob"

age = 30

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

**Output:**

My name is Bob and I am 30 years old.

**Using Positional and Keyword Arguments:**

print("Hello {1}, I am {0}".format("Alice", "Bob")) # Positional

print("Hello {name}, I am {friend}".format(name="Alice", friend="Bob")) # Keyword

**Output:**

Hello Bob, I am Alice

Hello Alice, I am Bob

**Specifying Precision:**

pi = 3.14159

print("Pi value: {:.3f}".format(pi)) # Output: Pi value: 3.142

**3. f-strings (Formatted String Literals) – Best Practice**

Introduced in Python 3.6, **f-strings** provide a more readable and efficient way to format strings.

**Syntax:**

f"string {variable}"

**Example:**

name = "Charlie"

age = 28

print(f"My name is {name} and I am {age} years old.")

**Output:**

My name is Charlie and I am 28 years old.

**Using Expressions Inside f-strings:**

x, y = 5, 10

print(f"Sum of {x} and {y} is {x + y}")

**Output:**

Sum of 5 and 10 is 15

**Specifying Precision:**

pi = 3.14159

print(f"Pi value: {pi:.3f}") # Output: Pi value: 3.142

**Using f-strings with Dictionaries:**

person = {"name": "Eve", "age": 24}

print(f"Her name is {person['name']} and she is {person['age']} years old.")

**Output:**

Her name is Eve and she is 24 years old.

**Comparison of Formatting Methods**

| **Method** | **Pros** | **Cons** |
| --- | --- | --- |
| % Operator | Simple for basic formatting | Less readable, limited functionality |
| .format() | More flexible, supports named placeholders | Slightly verbose |
| f-strings | Most readable, supports expressions, best choice | Only available in Python 3.6+ |

**Best Practice:**

* **Use f-strings** for most cases (modern, readable, and efficient).
* Use .format() if working with older Python versions.
* Avoid % formatting unless maintaining very old code.