

## What are some of the ways to format strings when printing in python?

In Python, there are several ways to format strings for printing. Here are some commonly used methods:

### 1. f-Strings (Formatted String Literals)

- Introduced in Python 3.6, f-strings allow you to embed expressions inside string literals by prefixing the string with f.

Example:

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains three lines of Python code:

```
1 name = "Alice"
2 age = 25
3 print(f"My name is {name} and I am {age} years old.")
```

```
1 name = "Alice"
2 age = 25
3 print(f"My name is {name} and I am {age} years old.")
```

### 2. str.format() Method

- The format() method lets you format strings by placing placeholders {} within the string and passing values to replace them.

Example:

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains three lines of Python code:

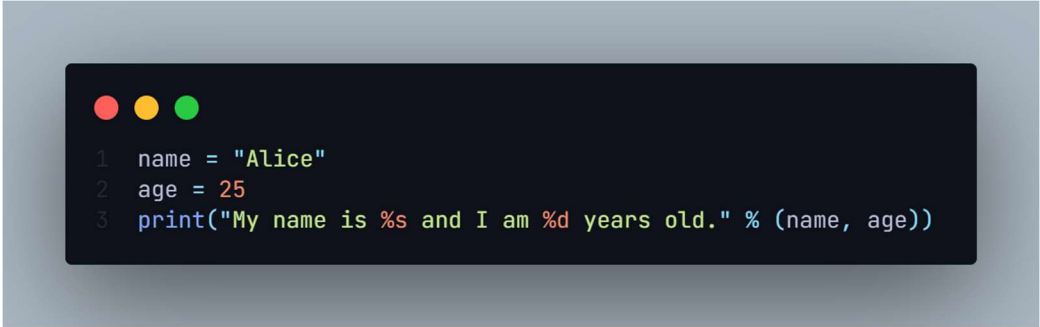
```
1 name = "Alice"
2 age = 25
3 print("My name is {} and I am {} years old.".format(name, age))
```

```
1 name = "Alice"
2 age = 25
3 print("My name is {} and I am {} years old.".format(name, age))
```

### 3. Old-style % Formatting (printf-style)

- This method uses % placeholders (like %s for strings, %d for integers) within the string, similar to C-style formatting.

Example:




```
1 name = "Alice"
2 age = 25
3 print("My name is %s and I am %d years old." % (name, age))
```

### 4. Template Strings (from string module)

- Template strings allow basic variable substitution using \$ placeholders. They are useful in cases where simpler formatting is needed.

Example:



```
1 from string import Template
2 template = Template("My name is $name and I am $age years old.")
3 print(template.substitute(name="Alice", age=25))
```

# What are variables in python and state the rules in their naming convention?

## Variables in Python

A **variable** in Python is a symbolic name that refers to a value or object in memory. It is used to store data that can be accessed and manipulated during the execution of a program.

## Rules for Naming Variables in Python

1. **Start with a letter or an underscore (\_):** A variable name must begin with a letter (a-z, A-Z) or an underscore (\_).
  - Valid: age, \_name
  - Invalid: 1age, @name
2. **Followed by letters, digits, or underscores:** After the first character, variable names can include letters, digits (0-9), and underscores.
  - Valid: age\_25, first\_name, myVar123
  - Invalid: first-name
3. **No Python Reserved Keywords:** You cannot use Python reserved words (keywords) as variable names, such as if, else, for, import, etc.
  - Invalid: class, True, and
4. **Case Sensitivity:** Python is case-sensitive, meaning that age and Age would be treated as different variables.
  - Valid: age, Age
  - Invalid: age and Age as the same variable
5. **No Special Characters:** Variable names cannot contain special characters like @, \$, %, etc.
  - Invalid: first@name, my\$var

6. **Avoid Starting with a Number:** Variable names cannot start with a number.

- Invalid: 2nd\_place, 4th\_var