Comments in Python

Comments in Python are used to explain code, make it more readable, and to prevent execution of specific lines of code. They are not executed by the Python interpreter, which means they do not affect the output of the program.

Types of Comments in Python

1. Single-line Comments
2. Multi-line Comments
3. Docstrings

Single-line Comments

A single-line comment starts with a # symbol and extends to the end of the line. Everything following the # on that line is ignored by the Python interpreter.

Multi-line Comments

Python does not have a specific syntax for multi-line comments like some other languages. Instead, you can use multiple single-line comments, or use triple quotes (either ''' or """) to write a comment block. However, technically, triple-quoted strings are treated as string literals, not comments. If not assigned to a variable or used, they are ignored.

Docstrings

Docstrings are a special kind of comment used to document modules, classes, functions, and methods. They are written as the first statement inside a function, method, class, or module using triple quotes (''' or """). Unlike regular comments, docstrings are retained at runtime and can be accessed via the __doc__ attribute.

```
# This is a single-line comment
print("Hello, World!")  # This is an inline comment
```

```
Hello, World!
```

```
def block_comment():
  # This is a block level single line comment
  a = 10
  b = 20
  return a+b

print(block_comment())
```

```
30
```

```
# This is a multi-line comment
# It spans multiple lines
# Each line starts with a '#'


'''
This is a multi-line comment
It spans multiple lines
Using triple quotes
Note - If the triple-quoted block is not assigned to a variable, it is ignored
'''
print("Hello, World!")
```

```
Hello, World!
```

```
def add(a, b):
    """
    This function adds two numbers.
    Arguments:
    a -- first number
    b -- second number
    Returns:
    Sum of a and b
    """
    return a + b

print(add.__doc__)  # Prints the docstring of the function
```

```

    This function adds two numbers.
    Arguments:
    a -- first number
    b -- second number
    Returns:
    Sum of a and b
    
```