Conditional Execution

A conditional execution is a set of commands that are executed only if a certain condition is met. In Python, the `if` statement is used to implement conditional execution. The syntax of the `if` statement is as follows:

```
if condition:
    statement1
    statement2
    ...
    statementN
```

The `condition` is an expression that evaluates to either `True` or `False`. If the condition is `True`, the statements inside the `if` block are executed

Comparison Operators

Comparison operators are used to compare two values and return a Boolean value (`True` or `False`). Here are some of the most common comparison operators in Python:

1. **Equal to (`==`)**: Returns `True` if two values are equal.
2. **Not equal to (`!=`)**: Returns `True` if two values are not equal.
3. **Greater than (`>`)**: Returns `True` if the first value is greater than the second value.
4. **Less than (`<`)**: Returns `True` if the first value is less than the second value.
5. **Greater than or equal to (`>=`)**: Returns `True` if the first value is greater than or equal to the second value.
6. **Less than or equal to (`<=`)**: Returns `True` if the first value is less than or equal to the second value.

Indentation

In Python, indentation is used to define blocks of code. The statements that are indented at the same level are considered part of the same block of code. The `if` statement and other control structures in Python use indentation to define the blocks of code that are executed conditionally.

```
if condition:
    """
    These statements are indented to indicate that they are part of the if block
    These statements are executed only if the condition is True
    """
    statement1
    statement2
    ...
    statementN
print("This statement is not indented and is executed regardless of the condition")
```

If Statement

The `if` statement is used to execute a block of code only if a certain condition is `True`. The syntax of the `if` statement is as follows:

```
if condition:
    statement1
    statement2
    ...
    statementN
```

Example:

```
x = 10
if x > 5:
    print("x is greater than 5")
```

In the above example, the condition `x > 5` is `True` because the value of `x` is `10`. Therefore, the statement `print("x is greater than 5")` is executed.

If-else Statement

The `if-else` statement is used to execute one block of code if a certain condition is `True` and another block of code if the condition is `False`. The syntax of the `if-else` statement is as follows:

```
if condition:
    statement1
    statement2
    ...
    statementN
else:
    statement1
    statement2
    ...
    statementN
```

Example:

```
x = 10
if x > 5:
    print("x is greater than 5")
else:
    print("x is less than or equal to 5")
```

In the above example, the condition `x > 5` is `True` because the value of `x` is `10`. Therefore, the statement `print("x is greater than 5")` is executed.

If-elif-else Statement

The `if-elif-else` statement is used to execute one block of code if a certain condition is `True`, another block of code if another condition is `True`, and a default block of code if none of the conditions are `True`. The syntax of the `if-elif-else` statement is as follows:

```
if condition1:
    statement1
    statement2
    ...
    statementN
elif condition2:
    statement1
    statement2
    ...
    statementN
else:
    statement1
    statement2
    ...
    statementN
```

Example:

```python
x = 10
if x > 5:
    print("x is greater than 5")
elif x < 5:
    print("x is less than 5")
else:
    print("x is equal to 5")
```

In the above example, the condition `x > 5` is `True` because the value of `x` is `10`. Therefore, the statement `print("x is greater than 5")

Nested If Statements

Nested `if` statements are `if` statements that are placed inside another `if` statement. They are used to check multiple conditions in a sequence. The syntax of nested `if` statements is as follows:

```python
if condition1:
    if condition2:
        statement1
        statement2
        ...
        statementN
    else:
        statement1
        statement2
        ...
        statementN
else:
    statement1
    statement2
    ...
    statementN
```