1. Comparison operators
2. Branching
3. Logical operators

1. Comparison operations

Comparison operations are essential in programming. They help compare values and make decisions based on the results.

Equality operator

The equality operator `==` checks if two values are equal. For example, in Python:

```
1. 1
2. 2
3. 3
```

```
1. age = 25
2. if age == 25:
3.     print("You are 25 years old.")
```

Copied!

Here, the code checks if the variable age is equal to 25 and prints a message accordingly.

Inequality operator

The inequality operator `!=` checks if two values are not equal:

```
1. 1
2. 2
```

```
1. if age != 30:
2.     print("You are not 30 years old.")
```

Copied!

Here, the code checks if the variable age is not equal to 30 and prints a message accordingly.

Greater than and less than

You can also compare if one value is greater than another.

```
1. 1
2. 2
```

Branching is like making decisions in your program based on conditions. Think of it as real-life choices.

The IF statement

Consider a real-life scenario of entering a bar. If you're above a certain age, you can enter; otherwise, you cannot.

```
1. 1
2. 2
3. 3
4. 4
5. 5
```

```
1. age = 20
2. if age >= 21:
3.     print("You can enter the bar.")
4. else:
5.     print("Sorry, you cannot enter.")
```

Copied!

Here, you are using the `if` statement to make a decision based on the age variable.

The ELIF Statement

Sometimes, there are multiple conditions to check. For example, if you're not old enough for the bar, you can go to a movie instead.

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
```

```
1. if age >= 21:
2.     print("You can enter the bar.")
3. elif age >= 18:
4.     print("You can watch a movie.")
5. else:
6.     print("Sorry, you cannot do either.")
```

Copied!

Real-life example: Automated Teller Machine (ATM)

When a user interacts with an ATM, the software in the ATM can use branching to make decisions based on the user's input. For example, if the user selects "Withdraw Cash" the ATM can branch into different denominations of bills to dispense based on the amount requested.

```
2. if user_choice == "Withdraw Cash":
3.     amount = input("Enter the amount to withdraw: ")
4.     if amount % 10 == 0:
5.         dispense_cash(amount)
6.     else:
7.         print("Please enter a multiple of 10.")
8. else:
9.     print("Thank you for using the ATM.")
```

Copied!

3. Logical operators

Logical operators help combine and manipulate conditions.

The NOT operator

Real-life example: Notification settings

In a smartphone's notification settings, you can use the NOT operator to control when to send notifications. For example, you might only want to receive notifications when your phone is not in "Do Not Disturb" mode.

The not operator negates a condition.

```
1. 1
2. 2
3. 3
```

```
1. is_do_not_disturb = True
2. if not is_do_not_disturb:
3.     send_notification("New message received")
```

Copied!

The AND operator

Real-life example: Access control

In a secure facility, you can use the AND operator to check multiple conditions for access. To open a high-security door, a person might need both a valid ID card and a matching fingerprint.

The AND operator checks if all required conditions are true, like needing both keys to open a safe.

```
1. 1
2. 2
3. 3
4. 4
```

Real-life example: Movie night decision

When planning a movie night with friends, you can use the OR operator to decide on a movie genre. You'll choose a movie if at least one person is interested.

The OR operator checks if at least one condition is true. It's like choosing between different movies to watch.

```
1. 1
2. 2
3. 3
4. 4
5. 5
```

```
1. friend1_likes_comedy = True
2. friend2_likes_action = False
3. friend3_likes_drama = False
4. if friend1_likes_comedy or friend2_likes_action or friend3_likes_drama:
5.     choose a movie()
```

Copied!

Summary

In this reading, you delved into the most frequently used operator and the concept of conditional branching, which encompasses the utilization of `if` statements and `if-else` statements.

Author

Muhammad Faheem Iqbal