Conditional and Branching

Estimated time needed: 10 minutes

Objective:-

By the end of this reading, you should be able to:

1. Comparison Operator
2. Branching
3. Logical Operator

1. Comparison Operations

Comparison operations are essential in programming. They help us 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 if it is.

inequality

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 if it is.

Greater Than and Less Than

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

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

```
1. if age>= 20:
2.     Print("Yes, the Age is greater than 20")
```

Copied!

Here, the code checks if the variable age is greater than equal to 30 and prints a message if it is.

2. Branching

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: 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, we use 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.

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
```

```
1. user_choice = "Withdraw Cash"
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 us 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 multiple conditions are all true. Like needing both keys to open a safe.

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

```
has_valid_id_card = True
has_matching_fingerprint = True
if has_valid_id_card and has_matching_fingerprint:
    open_high_security_door()
```

Copied!

The OR Operator

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
2
3
4
5
```

```
friend1_likes_comedy = True
friend2_likes_action = False
friend3_likes_drama = False
if friend1_likes_comedy or friend2_likes_action or friend3_likes_drama:
    choose a movie()
```

Copied!

Summary

In this reading, we 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

Akansha Yadav

Changelog

Date	Version	Changed by	Change Description
2023-11-03	1.0	Akansha Yadav	Created a reading file

Date	Version	Changed by	Change Description