Conditional Statements: Takeaways

Syntax

- Using an if statement to control your code:

```
if True:

    print(1)

if 1 == 1:

    print(2)

    print(3)
```

- Combining multiple conditions:

```
if 3 > 1 and 'data == data':

    print('Both conditions are true!')

if 10 < 20 or 4 <= 5:

    print('At least one condition is true.')
```

- Building more complex if statements:

```
if (20 > 3 and 2 != 1) or 'Games' == 'Games':

    print('At least one condition is true.')
```

- Using the else clause:

```
if False:

    print(1)

else:

    print('The condition above was false.')
```

- Using the elif clause:

```
if False:

    print(1)

elif 30 > 5:

    print('The condition above was false.')
```

Concepts

- We can use an `if` **statement** to implement a condition in our code.
- An `elif` clause is executed if the preceding `if` statement (or the other preceding `elif` clauses) resolves to `False` *and* the condition specified after the `elif` keyword evaluates to `True` .
- `True` and `False` are **Boolean values**.
- `and` and `or` are **logical operators**, and they bridge two or more Booleans together.
- We can compare a value `A` to value `B` to determine whether:
    - `A` is **equal** to `B` and vice versa (`B` is equal to `A`) — `==` .
    - `A` is **not equal** to `B` and vice versa — `!=` .
    - `A` is **greater** than `B` or vice versa — `>` .
    - `A` is **greater than or equal to** `B` or vice versa — `>=` .
    - `A` is **less** than `B` or vice versa — `<` .
    - `A` is **less than or equal to** `B` or vice versa — `<=` .

Resources

- If Statements in Python