Control Flow and Loops

If-Else Conditional Statements

What are Conditional Statements?

- Conditional statements allow you to execute code based on certain conditions.
- Python uses `if`, `elif`, and `else` for decision-making.

Syntax:

```
if condition1:
    # Code to execute if condition1 is True
elif condition2:
    # Code to execute if condition2 is True
else:
    # Code to execute if all conditions are False
```

Example:

```
age = 18

if age < 18:
    print("You are a minor.")
elif age == 18:
    print("You just became an adult!")
else:
    print("You are an adult.")
```

Match Case Statements in Python (Python 3.10+)

What is Match-Case?

- Match-case is a new feature introduced in Python 3.10 for pattern matching.
- It simplifies complex conditional logic.

Syntax:

```
match value:
    case pattern1:
        # Code to execute if value matches pattern1
    case pattern2:
        # Code to execute if value matches pattern2
    case _:
        # Default case (if no patterns match)
```

Example:

```
status = 404

match status:
    case 200:
        print("Success!")
    case 404:
        print("Not Found")
    case _:
        print("Unknown Status")
```

For Loops in Python

What are For Loops?

- For loops are used to iterate over a sequence (e.g., list, string, range).
- They execute a block of code repeatedly for each item in the sequence.

Syntax:

```
for item in sequence:
    # Code to execute for each item
```

Example:

```
fruits = ["apple", "banana", "cherry"]

for fruit in fruits:
    print(fruit)
```

Using `range()`:

- The `range()` function generates a sequence of numbers.
- Example:

```
for i in range(5):
    print(i)  # Output: 0, 1, 2, 3, 4
```

While Loops in Python

What are While Loops?

- While loops execute a block of code as long as a condition is `True`.
- They are useful when the number of iterations is not known in advance.

Syntax:

```
while condition:
    # Code to execute while condition is True
```

Example:

```
count = 0

while count < 5:
    print(count)
    count += 1
```

Infinite Loops:

- Be careful to avoid infinite loops by ensuring the condition eventually becomes `False`.
- Example of an infinite loop:

```
while True:
    print("This will run forever!")
```

Break, Continue, and Pass Statements

Break

- The `break` statement is used to exit a loop prematurely.
- Example:

```
for i in range(10):
    if i == 5:
        break
    print(i)  # Output: 0, 1, 2, 3, 4
```

Continue

- The `continue` statement skips the rest of the code in the current iteration and moves to the next iteration.

- Example:

```
for i in range(5):
    if i == 2:
        continue
    print(i)  # Output: 0, 1, 3, 4
```

Pass

- The `pass` statement is a placeholder that does nothing. It is used when syntax requires a statement but no action is needed.
- Example:

```
for i in range(5):
    if i == 3:
        pass  # Do nothing
    print(i)  # Output: 0, 1, 2, 3, 4
```

Summary

- Use `if`, `elif`, and `else` for decision-making.
- Use `match-case` for pattern matching (Python 3.10+).
- Use `for` loops to iterate over sequences and `while` loops for repeated execution based on a condition.
- Control loop execution with `break`, `continue`, and `pass`.