Lesson 2: Conditional Statements and Loops in Python

Marcin Kurzyna

Lecture Goals

This lecture introduces how to control the flow of execution in Python using **conditional statements** and **loops**. By the end, students should be able to:

- Use `if`, `elif`, and `else` statements to make decisions.
- Use `match` statements (Python 3.10+) for multi-branch selection.
- Use `for` and `while` loops to repeat actions.
- Understand basic flow control structures in Python.

1 Introduction to Control Flow

So far, Python programs have executed line by line in order. Control flow statements allow us to:

- Make **decisions** based on conditions.
- **Repeat** code using loops.

These features make programs more flexible and interactive.

2 Conditional Statements

Conditional statements check whether a condition is true and perform different actions accordingly.

2.1 The `if` Statement

The simplest form:

```
x = 10
if x > 0:
    print("x is positive")
```

Indentation is critical in Python — it defines which statements belong to the `if` block.

2.2 `if-else` and `elif`

We can provide alternative branches:

```
number = -3

if number > 0:
    print("Positive")
elif number < 0:
    print("Negative")
else:
    print("Zero")
```

Python executes only the first true branch and then skips the rest.

2.3 Nested `if` Statements

You can nest conditions inside one another:

```
age = 20

if age >= 18:
    if age >= 65:
        print("Senior citizen")
    else:
        print("Adult")
else:
    print("Minor")
```

2.4 Comparison and Logical Operators

Conditions often combine comparisons and logical operations.

Operator	Meaning
`==`	equal to
`!=`	not equal to
`>`	greater than
`<`	less than
`>=`	greater than or equal to
`<=`	less than or equal to
`and`	logical AND
`or`	logical OR
`not`	logical NOT

Example:

```
temperature = 25
if temperature > 15 and temperature < 30:
    print("Pleasant weather")
```

3 The `match` Statement (Python 3.10+)

The `match` statement provides a clean way to handle multiple conditions (similar to `switch` in other languages).

```
day = 3

match day:
    case 1:
        print("Monday")
    case 2:
        print("Tuesday")
    case 3:
        print("Wednesday")
    case _:
        print("Invalid day")
```

`_` acts as the default case, matching anything not listed above.

4 Loops

Loops are used to repeat a block of code multiple times.

4.1 The `while` Loop

The `while` loop runs as long as the condition is true.

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

Note: Always ensure that the condition will eventually become false, or you will create an infinite loop.

4.2 The `for` Loop

The `for` loop iterates over a sequence (like a list or range).

```
for i in range(1, 6):
    print("i =", i)
```

The `range(start, stop)` function generates a sequence of numbers from `start` up to (but not including) `stop`.

4.3 Loop Control Statements

- `break` — exits the loop immediately.
- `continue` — skips the rest of the loop body for the current iteration.

Example:

```python
for i in range(1, 6):
    if i == 3:
        continue  # skip number 3
    if i == 5:
        break     # stop the loop completely
    print(i)
```

4.4 Nested Loops

A loop can contain another loop inside it:

```python
for i in range(1, 4):
    for j in range(1, 3):
        print(f"i={i}, j={j}")
```

5 Example: Summing Numbers

Program to compute the sum of numbers from 1 to 10:

```python
total = 0
for i in range(1, 11):
    total += i

print("The sum is", total)
```

Summary

This lecture introduced:

- Conditional statements: `if`, `elif`, `else`, and `match`.
- Loop constructs: `while` and `for`.
- Basic loop control with `break` and `continue`.

6 Exercises

1. Write a Python program that checks if a given number is positive, negative, or zero.
2. Modify the above program to also print whether the number is even or odd.
3. Use a `match` statement to print the name of a month given its number (1–12).
4. Write a program that prints numbers from 1 to 10 using a `while` loop.
5. Write a program that prints even numbers between 1 and 20 using a `for` loop.
6. Create a program that calculates the factorial of a number (e.g. $5! = 120$) using a `for` loop.
7. Rewrite the factorial program using a `while` loop.

8. Write a program that repeatedly asks for a password until the user enters the correct one.

9. Use nested loops to print the following pattern:

```
****
****
****
```

10. Challenge: Write a program that prints the multiplication table (1–10) using nested `for` loops.