Errors

Week 9

What is an error?

- **Errors**, or "bugs," are a common occurrence in programming.
- An error is **any issue in the code** that prevents it from functioning as intended.
- Errors are an **unavoidable** part of programming.
- Learning to identify and fix them is a crucial skill for any programmer.
- Errors can be caused by various factors, including incorrect syntax, logical errors, and runtime errors.
- They can range from simple typos to complex logical mistakes.

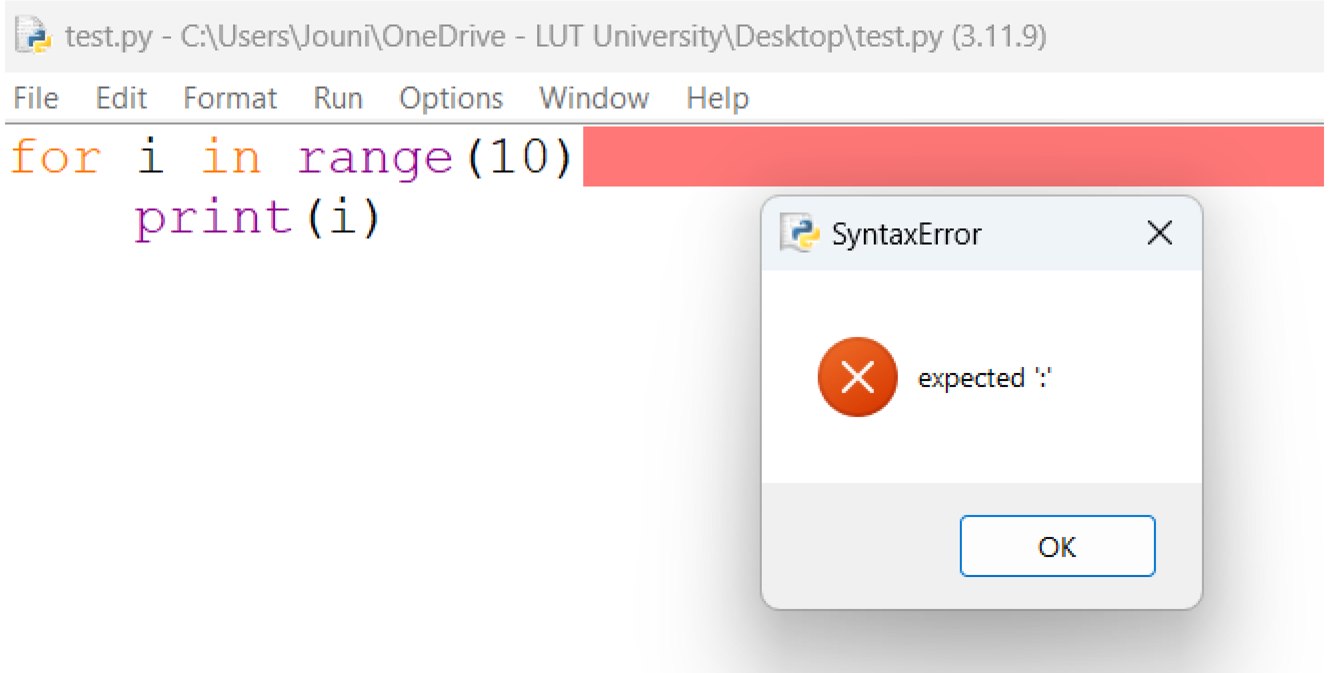
Syntax errors

The code violates the specific grammar rules of the programming language:

- Missing colon
- Indentation error
- Mismatched parentheses
- Incorrect variable assignment: `10 = x`
- Using reserved keywords: `def = "house"`

Usually caught by the interpreter (or compiler).

Syntax error



test.py - C:\Users\Jouni\OneDrive - LUT University\Desktop\test.py (3.11.9)
File
Edit
Format
Run
Options
Window
Help

```
for i in range(10)
    print(i)
```

SyntaxError

expected ':'

Runtime errors

These errors arise during the execution of a program, often due to unexpected conditions:

- Division by zero: `10 / 0`
- Type error: `"hello" + 5`
- Name error: **variable** is not (yet) defined
- Index error

Runtime errors can cause the program to **crash** or produce **incorrect** results.

Logical errors

These types of errors are **harder to detect** because the code runs without crashing, but the results are not what the programmer intended.

- **Incorrect conditions:** Writing a condition that is never `True` or does not match the intended purpose.
- **Error in loops:** Miscounting iterations by one in loops can cause missing or extra iterations.
- **Incorrect variable initialization:** Setting a variable to an incorrect starting value.
- **Incorrect operator usage:** Using the wrong operator, for example `*` instead of `+`, in calculations.

Debugging

To identify and fix errors, programmers employ various debugging techniques:

- Insert **print statements** to display variable values at specific points in the code can help trace the program's execution flow.
- **Debuggers** are specialized tools that allow you to **step** through code line by line, **inspect variables**, and set **breakpoints**.
- Have **other programmers** review your code can help identify potential errors and improve code quality. You can also ask AI.
- Use **unit testing** by writing small tests for individual code components can help isolate and fix errors