Python and SQL: intro / SQL platforms

Ewa Weychert

Class 1: Introduction to Python

Who am I? ;)

- mgr Ewa Weychert
- Research interests: demography, machine learning, NLP
- Collaboration with LabFam (Interdisciplinary Centre for Labour Market and Family Dynamics)
- Working at University of Florence
- Conducting courses: Advanced Programming in R, Webscraping and Social Media Scraping, Machine Learning 2: predictive models, deep learning, neural network, Python and SQL: intro / SQL platforms, Introduction to Data Science
- `e.weychert@uw.edu.pl`

Who are you? ;)

To better conduct the course I want to know you.
Please fill this anonymous Google Form:

Click here to open the form

Plan of the Course - Thursday group 3 (16:45) group 4 (18:30)

#	**Date**	**Topics**
1	Thu, Oct 2, 2025	Course intro; Why Python?; Installing Python, Jupyter, VS Code; Calculator use; Data types (int, float, str, bool); Libraries; Importing/saving files
2	Thu, Oct 9, 2025	If statements – conditions, comparisons, nesting
3	Thu, Oct 16, 2025	For loops – iteration over lists/ranges, nesting, break/continue
4	Thu, Oct 23, 2025	Functions – definitions, parameters, return values, scope
5	Thu, Oct 30, 2025	Pandas – reading/writing, DataFrames, indexing, filtering, aggregation
6	Thu, Nov 6, 2025	NumPy – arrays, vectors, matrices, broadcasting, map
7	Thu, Nov 13, 2025	OOP – classes, objects, methods, attributes, inheritance
8	Thu, Nov 20, 2025	Visualisation – Seaborn and Matplotlib plots, customization
9	Thu, Nov 27, 2025	Streamlit – dashboards, interactive apps, deployment
10	Thu, Dec 4, 2025	SQL – queries, filtering, joins, aggregations
11	Thu, Dec 11, 2025	SQL + Python – connecting DBs, queries in Pandas
12	Thu, Dec 18, 2025	Django – models, views, templates, simple web apps
13	Thu, Jan 8, 2026	Presentations – student projects + feedback
14	Thu, Jan 15, 2026	Presentations – student projects + feedback
15	Thu, Jan 22, 2026	Presentations – student projects + feedback

Plan of the Course - Monday group 1 (11:30) group 2 (13:15)

#	**Date**	**Topics**
1	Thu, Oct 2, 2025	Course intro; Why Python?; Installing Python, Jupyter, VS Code; Calculator use; Data types (int, float, str, bool); Libraries; Importing/saving files
2	Mon, Oct 6, 2025	If statements – conditions, comparisons, nesting
3	Mon, Oct 13, 2025	For loops – iteration over lists/ranges, nesting, break/continue
4	Mon, Oct 20, 2025	Functions – definitions, parameters, return values, scope
5	Mon, Oct 27, 2025	Pandas – reading/writing, DataFrames, indexing, filtering, aggregation
6	Mon, Nov 3, 2025	NumPy – arrays, vectors, matrices, broadcasting, map
7	Thu, Nov 13, 2025	OOP – classes, objects, methods, attributes, inheritance
8	Mon, Nov 17, 2025	Visualisation – Seaborn and Matplotlib plots, customization
9	Mon, Nov 24, 2025	Streamlit – dashboards, interactive apps, deployment
10	Mon, Dec 1, 2025	SQL – queries, filtering, joins, aggregations
11	Mon, Dec 8, 2025	SQL + Python – connecting DBs, queries in Pandas
12	Mon, Dec 15, 2025	Django – models, views, templates, simple web apps
13	Mon, Jan 12, 2026	Presentations – student projects + feedback
14	Mon, Jan 19, 2026	Presentations – student projects + feedback
15	Fri, Jan 23, 2026	Presentations – student projects + feedback

Final Grade

- Final Exam: Multiple-choice questions (MCQs) (0.5)
- Final Individual Project presentation (0.5)

Final Exam

- Format: Multiple-choice questions (MCQs) (A multiple-choice question is a type of objective assessment in which a question has zero or more possible answers)
- Number of questions: 25
- Time limit: 90 minutes
- Date of exam – during the examination session: 26 January 2026 – 8 February 2026 (to be determined later by the administration office)

Final Presentation

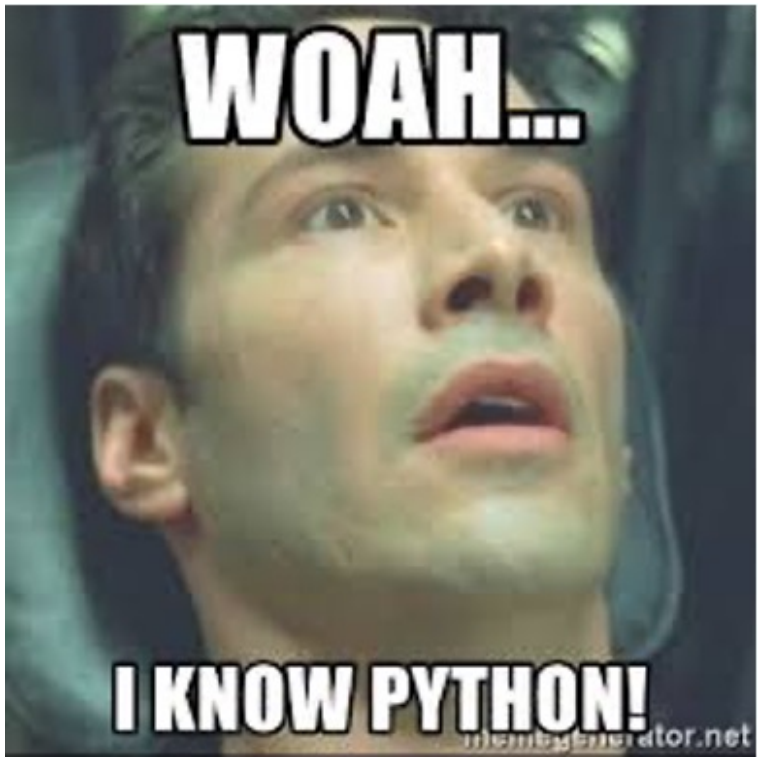
- Project must be done **individually**
- Deadlines: Presentation + code submission **07.01.2026 via Moodle**
- Final project must include:
  - **Problem/Dataset:** choose dataset (Kaggle, open data, or self-collected); define clear research question/goal.
  - **Data Wrangling (Pandas + NumPy):** import, clean (missing values, duplicates), exploratory analysis (summaries, aggregations).
  - **Logic & Functions:** Python functions, if/loops, simple algorithms (e.g. similarity, rules).
  - **Visualization:** Seaborn/Matplotlib; 5–10 plots (trends, comparisons, distributions).
  - **SQL:** queries, joins, filtering, aggregations.
  - **Presentation Layer:** Streamlit (interactive charts, filters) or Django (web app with models/views/templates)

When to present Final Project?

- Final project presentations will take place during the last three classes in January:
  - **08 January 2026**
  - **15 January 2026**
  - **22 January 2026**
- You will be able to **choose your presentation date** via a Doodle form.
- The link to the Doodle form will be distributed **next Thursday at 09:00, 09.10.2025.** through **USOS email**.
- Each presentation will last **10 minutes**
- the way you present and the originality of the idea matter for the final grade

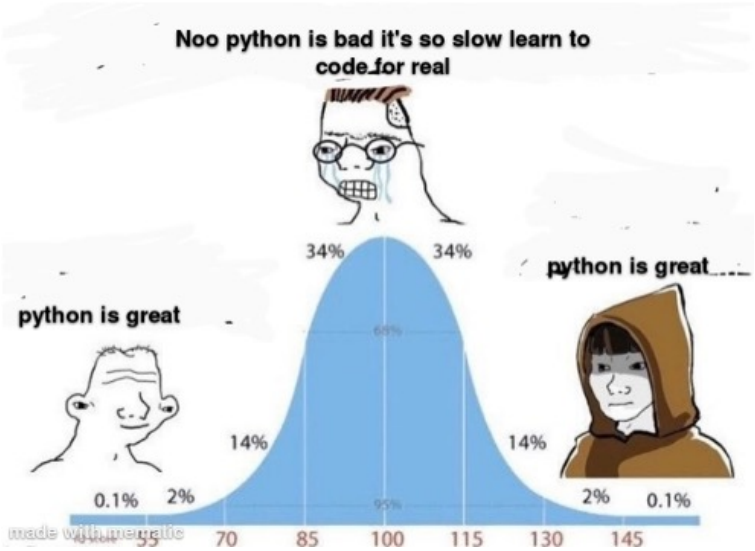
What is the aim for me of this class? (1/2)

- **Code automation in Python** – automation reduces manual work, increases efficiency, and eliminates human errors. In Python, we can automate various tasks, such as file handling, data processing, and reporting.
- **Reusability and reproducibility of code** – writing code that can be reused and easily reproduced is key to effective programming.
- **Understanding the difference between functional and object-oriented programming** – both paradigms have different advantages, and recognizing when to use each is important.

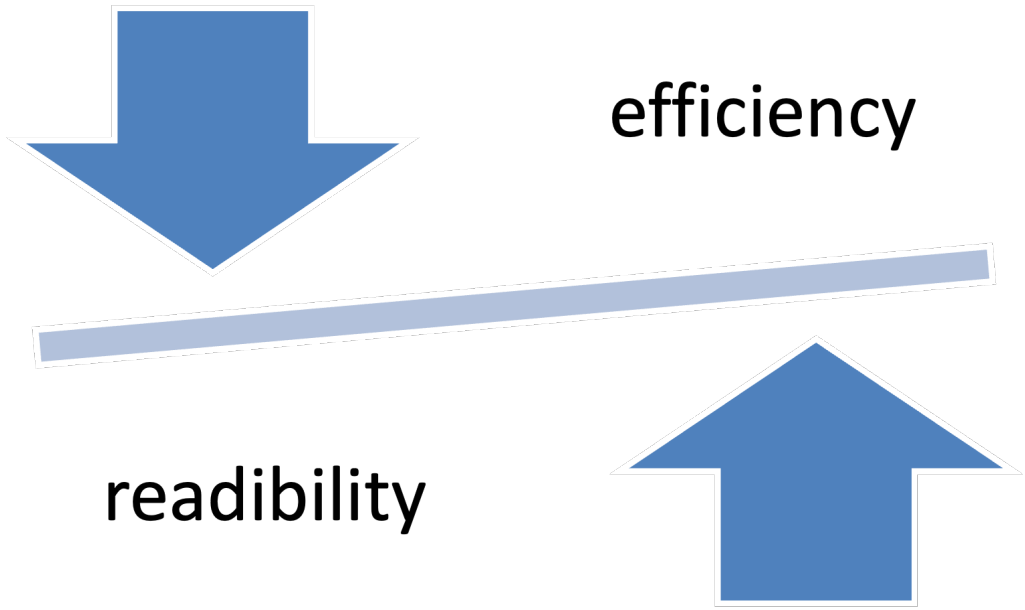


What is the aim for me of this class? (2/2)

- **Understanding time efficiency of code** – optimization allows applications to run faster and use fewer resources.
- **Writing clean and optimized code** – code should be readable, easy to understand, and well-optimized, which is essential for every programmer.



Efficiency Readability Trade-off



Plan of today's class

1. What is Python?
2. What is python and why so useful?
3. How to install Python?
4. Jupyter Notebook
5. Visual Studio Code
6. Python - as calculator
7. Types of data: Variables and data types (int, float, str, bool)
8. Concept of libraries in python
9. Importing and saving files to python

What is Python?

1. Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation
2. The name Python derives from the British comedy series Monty Python's Flying Circus
3. As of December 2022, Python was the most popular programming language.
4. Python was selected as *Programming Language of the Year* (for "the highest rise in ratings in a year") in 2007, 2010, 2018, 2020, 2021, and 2024 — the only language to have done so six times as of 2025

Why so useful?

- **Easy to learn and read** – syntax is simple and close to English.
- **Versatile** – can be used across multiple domains.
- **Large standard library** – many built-in modules (e.g., `math`, `datetime`, `os`).
- **Extensive ecosystem of libraries and frameworks**:
  - Data Science: `pandas`, `numpy`, `matplotlib`, `scikit-learn`
  - Web: `Django`, `Flask`, `FastAPI`
  - AI/ML: `TensorFlow`, `PyTorch`
- **Cross-platform** – works on Windows, macOS, Linux, and embedded systems.
- **Strong community support** – large global community and abundant tutorials.
- **Integration** – connects easily with databases, APIs, and other languages.
- **Common applications**:
  - Scripting for web applications
  - Scientific computing
  - Artificial intelligence and machine learning projects
  - Embedded scripting in software and hardware products

How to install Python?

1. Go to the official website: python.org/downloads
2. Download the latest stable version (Python 3.x).
3. Run the installer:
   - On Windows: check **"Add Python to PATH"** before installing.
   - On macOS: open the downloaded `.pkg` file.
4. Verify installation by opening a terminal/command prompt and running:
   `python -version|`

Installing Anaconda vs Vanilla Python

- **Vanilla Python**
  - Downloaded from `python.org`
  - Lightweight, just the Python interpreter + standard library
  - Requires manual installation of packages with `pip`
  - Good for small scripts and when you want full control over dependencies
- **Anaconda Distribution**
  - Includes Python + many pre-installed packages (`numpy`, `pandas`, `matplotlib`, etc.)
  - Comes with `conda` package/environment manager
  - Provides **Jupyter Notebook**, **Spyder**, and other tools out-of-the-box
  - Convenient for data science and machine learning beginners
- **Recommendation:**
  - I recommend using **Vanilla Python** because it is lightweight, flexible, and teaches you how to manage your own environments and packages.
  - Use **Anaconda** if you want an "all-in-one" solution with many libraries pre-installed.

Jupyter Notebook

- **What is Jupyter Notebook?** An interactive development environment for Python and data science. Runs in the browser and allows mixing **code**, **text**, **math**, and **visualizations**.
- **Why is it useful?**
  - Great for experiments, teaching, and presentations
  - Easy to run code step by step in "cells"
  - Supports rich output (tables, plots, images)
  - Widely used in machine learning and data science
- **Basic structure:**
  - **Code cells** – where Python code is executed
  - **Markdown cells** – for text, notes, equations (LaTeX)

Example

```
print("Hello Jupyter!")
```

Most Common Jupyter Notebook Shortcuts

- **Command Mode (press `Esc`)**
  - `A` – Insert new cell **above**
  - `B` – Insert new cell **below**
  - `D, D` – Delete selected cell
  - `M` – Change cell to **Markdown**
  - `Y` – Change cell to **Code**
- **Edit Mode (press `Enter`)**
  - `Shift + Enter` – Run cell and go to next
  - `Ctrl + Enter` – Run cell and stay in it
  - `Alt + Enter` – Run cell and insert new one below
- **General**
  - `Ctrl + S` – Save notebook
  - `H` – Show all shortcuts help menu

Visual Studio Code

- **What is VS Code?** A lightweight, open-source code editor developed by Microsoft. Popular among Python developers for its speed, flexibility, and extensions.
- **Key Features:**
  - Syntax highlighting and IntelliSense (auto-completion, hints)
  - Integrated terminal for running Python directly
  - Debugging tools built in
  - Rich ecosystem of extensions (e.g., Python, Jupyter, Git, Docker)
  - Works on Windows, macOS, and Linux
- **Why use it for Python?**
  - Good for writing full Python scripts and applications
  - More flexible for larger projects than Jupyter Notebook
  - Supports version control (Git/GitHub) integration

First Python Script in Visual Studio Code

Step 1: Create a file `hello.py`

hello.py

```
print("Hello, Python!")
```

Step 2: Run it from the terminal

Command line

```
$ python hello.py
Hello, Python!
```

Notes:

- Use `python3` instead of `python` on some systems.
- Python scripts must be saved with the extension `.py`.
- This is the simplest way to start running Python outside of Jupyter.

Jupyter vs VS Code: When to Use Which

Jupyter Notebook

- Interactive, cell-by-cell execution
- Mix code, text (Markdown), equations, plots
- Great for exploration, EDA, teaching, demos
- Rich outputs (tables, images) inline
- Easy to share as `.ipynb` or HTML

Use Jupyter when:

- Prototyping / data exploration
- Building step-by-step narratives
- Trying out ML models quickly

Visual Studio Code

- Full editor: folders, projects, refactors
- Integrated terminal, debugger, testing
- Git/GitHub integration, extensions
- Better for scripts, packages, apps
- Works with notebooks *and* `.py` files

Use VS Code when:

- Building production-ready code
- Structuring larger projects/modules
- Need debugging, linting, CI, version control

Jupyter vs VS Code: When to Use Which

Rule of Thumb

Start in **Jupyter** to explore and explain; move to **VS Code** to engineer, test, and ship.

Python as a Calculator

- **Why use Python as a calculator?**
  - Quick way to test ideas and verify calculations
  - Handles integers, floats, and very large numbers easily
  - Supports scientific notation (`1e6 = 1,000,000`)
  - Foundation for advanced libraries in data science (`numpy`, `pandas`, `scipy`)
- **Examples:**
  - `2 + 3 * 4  # 14`
  - `10 // 3  # 3 (integer division)`
  - `10 % 3  # 1 (remainder)`
  - `2 ** 10  # 1024`

Mini Exercise

Compute Body Mass Index (BMI): `weight = 70; height = 1.75; weight / (height ** 2)`

Types of data: Variables and data types (int, float, str, bool)

- **int** – integer numbers `5`, `-12`, `1000`
- **float** – decimal numbers `3.14`, `-0.5`, `2.0`
- **str** – text or string values `"Hello"`, `'Python'`
- **bool** – logical values `True`, `False`

Example in Python:

```
x = 10          # int
y = 3.14        # float
name = "Alice"  # str
is_student = True  # bool
```

Type Casting in Python

- **What is type casting?** Converting one data type into another (explicit conversion).
- **Why is it useful?**
  - Ensures compatibility between variables
  - Common when working with user input (strings) and numbers
- **Examples:**

```
int("5")          # 5 (string → integer)
float(3)         # 3.0 (int → float)
str(123)        # "123"(int → string)
bool(0)         # False (int → boolean)
bool(42)       # True (nonzero int → boolean)
```

Key Idea

Python is dynamically typed, but explicit casting is often required when combining different types (e.g., strings and numbers).

Variables in Detail

- **Naming rules:**
  - Must start with a letter or underscore (_)
  - Cannot start with a number
  - Can contain letters, numbers, underscores
  - Case-sensitive (`name` ≠ `Name`)
- **Reserved words:** Certain words are reserved by Python and cannot be used as variable names (e.g., `for`, `while`, `if`, `class`, `True`, `None`).
- **Dynamic typing:** In Python, a variable's type is determined at runtime and can change.

Example

```
x = 5        # int
x = "Hello"  # str
x = 3.14     # float
```

Concept of Libraries in Python

- **What are libraries?** Collections of pre-written code that provide ready-to-use functions and tools.
- **Why use libraries?** Save time, avoid rewriting code, and access advanced functionality.
- **Types of libraries:**
  - Built-in (included with Python) – e.g., `math`, `datetime`, `os`
  - External (installed separately) – e.g., `numpy`, `pandas`, `matplotlib`

How to use them in Python

```
import math
math.sqrt(16)     # returns 4.0
```

Importing and Saving Files in Python (1/2)

- **CSV files** (Comma-Separated Values) - Common for tabular data - Easy to read/write with `pandas`

Example

```
import pandas as pd
df = pd.read_csv("data.csv")
df.to_csv("output.csv", index=False)
```

- **TXT files** (Plain text) - Flexible but less structured - Use Python's built-in `open()`

Example

```
with open("file.txt", "w") as f:
   f.write("Hello Python!")
```

Importing and Saving Files in Python (2/2)

- **Excel files** (.xlsx) - Widely used in business and research - Handled via `pandas` or `openpyxl`

Example

```
df = pd.read_excel("data.xlsx")
df.to_excel("output.xlsx", index=False)
```

- **JSON files** (JavaScript Object Notation) - Common for web data and APIs - Supports nested structures (dictionaries/lists)

Example

```
import json
with open("data.json") as f:
   data = json.load(f)
with open("out.json", "w") as f:
   json.dump(data, f)
```

Indentation in Python – Why Whitespace Matters

- **Indentation = code structure in Python** Unlike many languages (C, Java) that use {braces}, Python uses spaces/tabs to define blocks of code.
- **Rules:**
  - Consistent indentation is required (default: 4 spaces).
  - Mixing tabs and spaces leads to **IndentationError**.
  - Every block after : must be indented.
- **Example – Correct:**

```
if x > 0:
  print("Positive")
```

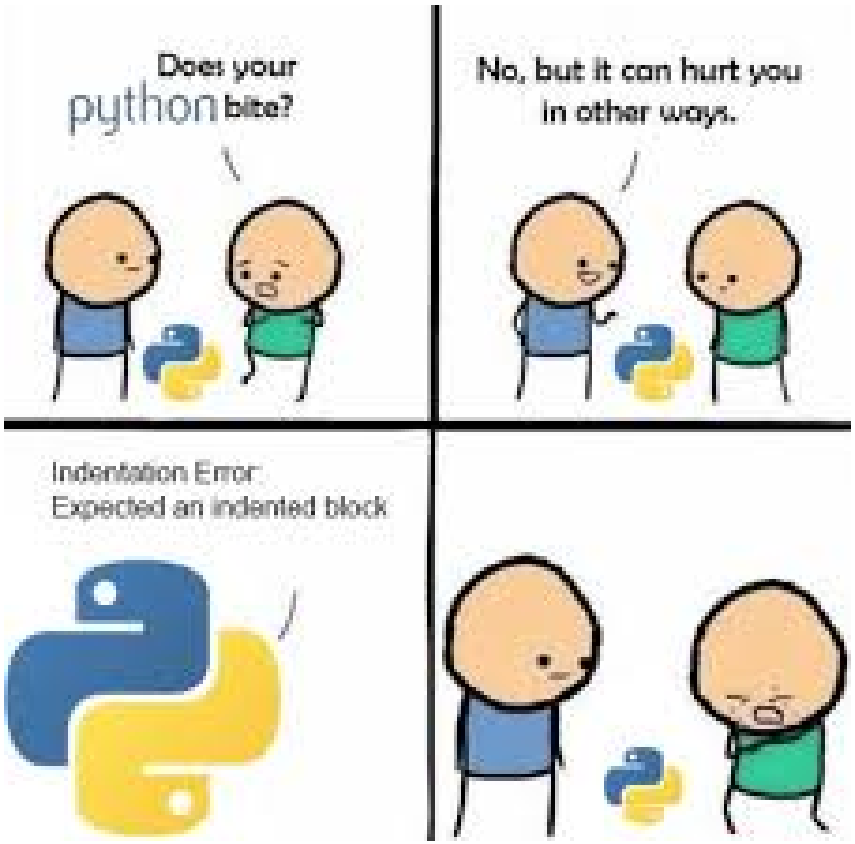
- **Example – Incorrect:**

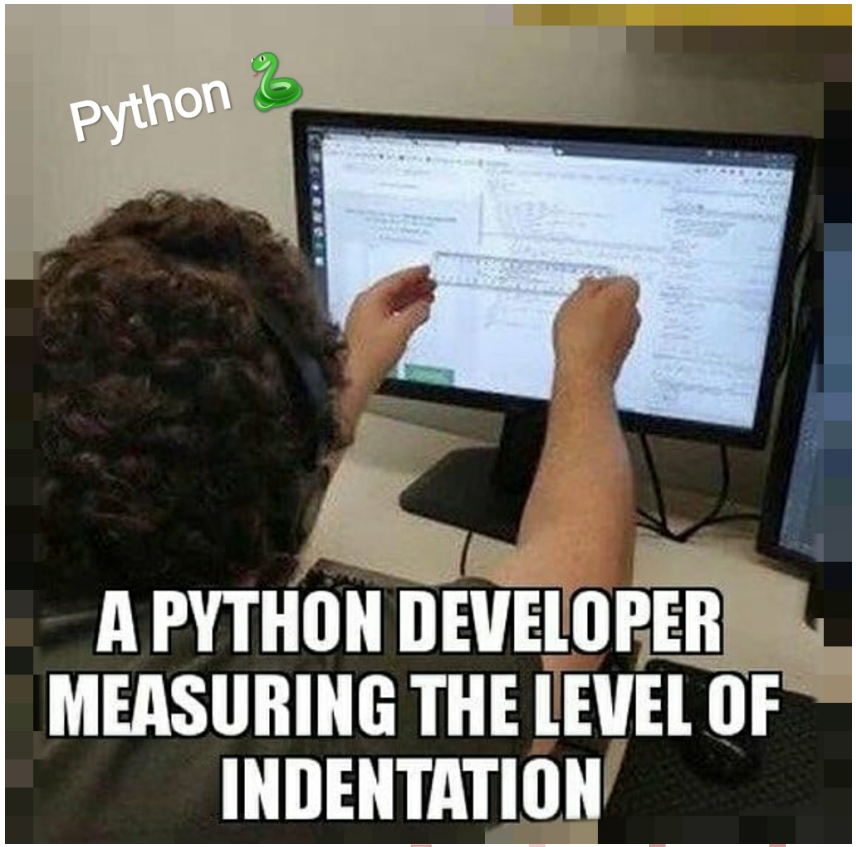
```
if x > 0:
  print("Positive")
 print("Still inside")     # inconsistent spacing
```

Indentation in Python – Why Whitespace Matters

Key Idea

Whitespace is not just for readability in Python — it defines the program's logic!





Errors in Python – Syntax vs Runtime

- **Syntax Errors**
  - Happen when Python code violates language rules.
  - Detected before execution.

Example – SyntaxError

```
if x > 0
  print("Positive")
```

- **Runtime Errors (Exceptions)**
  - Code is syntactically correct but fails during execution.
  - Usually caused by invalid operations.

Example – ZeroDivisionError

```
x = 5 / 0
# Error:  division by zero
```

Key Idea

Syntax errors stop the program before it runs. Runtime errors stop

Errors in Python – Syntax vs Runtime

- **Syntax Errors**
  - Happen when Python code violates language rules.
  - Detected before execution.

Example – SyntaxError

```
if x > 0
  print("Positive")
```

- **Runtime Errors (Exceptions)**
  - Code is syntactically correct but fails during execution.
  - Usually caused by invalid operations.

Example – ZeroDivisionError

```
x = 5 / 0
# Error:  division by zero
```

Key Idea

Syntax errors stop the program before it runs. Runtime errors stop execution while the program is running.

Python Comments

- **Why use comments?**
  - Explain code logic for yourself and others
  - Make code more readable and maintainable
  - Helpful for debugging (temporarily disable code)
- **Types of comments in Python:**
  - **Single-line comment:** starts with #

Example

```
# This is a single-line comment
print("Hello") # Prints Hello
```

  - **Multi-line (docstring-style) comment:** use triple quotes """..."""

Example

```
"This is a multi-line comment.  It can span several lines.  "
```

Wrap-up and Next Steps

Key Points from Today:

- Python is simple, versatile, and widely used in data science, AI, and web development.
- Development tools: Jupyter Notebook for exploration, VS Code for full projects.
- Python basics: calculator, variables, data types, libraries, file handling.
- Importance of writing clean, reusable, and efficient code.

Suggested Resources:

- W3Schools Python Tutorial – beginner-friendly reference.
- RealPython – tutorials and hands-on guides.
- Official Python Documentation – complete and authoritative.

Q1

What is the main reason Python requires indentation (whitespace) in code blocks?

A) To improve performance

B) To increase code readability only

C) **To define the program's structure and logic**

D) To separate comments from code

Q2

Which of the following data types are built-in to Python?

A) **int**

B) **float**

C) **str**

D) decimal

Q3

Which of the following statements about Python is correct?

A) Python does not support floating-point numbers.

B) Python requires curly braces {} to define code blocks.

C) Jupyter Notebook cannot display plots or images.

D) **None of the above**

Thank you!

See you next week

e.weychert@uw.edu.pl