UNI FREIBURG

Python Course

WT 24/25

Lecture: Marcus Rockel & Prof. Dr. Eva Lütkebohmert-Holtz, Tutorial: Felix Häschel
https://www.finance.uni-freiburg.de/

Tutorial 11

Exercise 1 (Carbon Emissions)**.**
In the `macro/Emissions_Data` directory inside the GitHub repository you will find an Excel file, `Carbon_Project_2023_Emissions_by_Nation.xlsx`, that contains a sheet named "Territorial Emissions" with CO_2 emissions data.
Your task is to:

- Load the data and rename columns from the first valid row, so each column represents a specific **country/region** (you can use year as index).
- (Optional) Multiply all emission values by a factor if needed (e.g., to convert units).
- Finally, **plot** a subset of regions (e.g. `World, Germany, USA, China, India`) on a line chart to illustrate their emissions over time.

Exercise 2 (World Bank WDI)**.**
A World Bank-style Excel file (`World_Bank_2022_WDI.xls`) has two unwanted header rows and possibly columns like `''World Development Indicators''`.
Perform the following steps:

- Drop the unneeded rows/columns.
- Transpose the table so that each row represents a **Year**, and each column a specific **country/region**.
- Convert the year labels to integers if they are strings.
- Use two CSV files:
  1. `income_levels.csv` mapping each country to "High," "Upper_Middle," "Lower_Middle," or "Low".
  2. `oecd_countries.csv` listing the countries in the OECD.
- Create **aggregated columns** for different region groups (e.g. "USA," "India," "OECD Europe," "Developing countries (=Upper_Middle),", "Low Income (=Low + Lower_Middle)").
- Generate one or more **plots** (e.g., line or bar charts) to illustrate the data by region or decade.

Exercise 3 (University System with GPA Tracking)**.**
Create a simple university system using Object-Oriented Programming principles. The system should include students, professors, and courses. Perform the following tasks:

- Define a `Person` class with attributes for `name` and `age`, and a method to display these details. Implement a `__str__` method for user-friendly print outputs.
- Extend the `Person` class to create a `Student` class:
  - Add attributes for `student_id`, `major`, and a `grades` dictionary to store course grades.
  - Use private attributes (e.g., `_grades`) where appropriate, and provide access through properties using the `@property` decorator.
  - Implement methods to add grades and calculate the **global GPA**.
- Extend the `Person` class to create a `Professor` class:
  - Add attributes for `professor_id` and `department`.
  - Add a method to calculate the average GPA of courses they teach.
- Create a `Course` class with:
  - Attributes for `course_name`, `professor`, and enrolled `students`.
  - Methods to add students, assign grades, and calculate the **course GPA**.
- Compare students based on their global GPA using comparison operators.
- Test your implementation with a professor, a course, and multiple students.