EPFL CIVIL-127, Lab 1

February 17, 2026

1.1

1. Install the following tools. If you have trouble installing any of these tools, please ask a TA for help:
    - uv, a Python version and package manager
    - Visual Studio Code
    - The following VS Code extensions:
        - Ruff, a code formatter and linter
        - Pylance, type checker
        - Python language support
        - Python debugger

    To install VSCode extensions, launch VSCode. Click on this icon or use the menu items "View" > "Extensions":

    You can then search for specific extensions. Make sure you are installing extensions from trusted sources.

    After installing extensions, you might need to restart VSCode.

    Note: installing software on Windows might require you to start a PowerShell as an administrator and run `set-executionpolicy remotesigned`. You might also have to disable S Mode.
2. Create a folder `CIVIL-127`. You can organize this folder and place it inside any other parent folder. This folder is going to be your workspace. Make sure this folder is included in your regular data backups since it's going to contain all your work this semester.
3. Open the `CIVIL-127` folder in VSCode ("File" > "Open folder…" menu item).

4. Open “Terminal” > “New Terminal” from within VSCode. Run the following command. You should see a .venv folder get created.

   ```
   uv venv --python 3.14.2
   ```

   Venvs (virtual envs) enables everyone to run the same version of Python (and packages when we start using them later). It reduces TA chaos and increases the chances that if the code runs on your computer, it will also run on other people’s computers.

5. Create a subfolder called `lab1`
6. Download and add check_python_version.py to `lab1/`. Run the code. If you are asked to pick an interpreter, make sure you are picking the one that’s inside the newly created .venv folder.

   If you see the message `Correct Python version`, your setup works. You are ready to write real programs!

1.2

1. Download and run exercise_1_2.py.
2. Can you explain why the `while` loop in exercise_1_2.py never exits?
3. If you can’t explain it, try debugging the program. Add print statements or set debugger breakpoints.

1.3

- Implement a two-player, text mode number guessing game (the same game as seen in class). Build your program incrementally. First make it accept a number. Then add the loop. Then add comparisons.
- Player 1 picks a number and player 2 keeps taking guesses. The program must tell player 2 if their guess is correct, too low, or too high.
- Only use coding patterns and constructs which were seen in class today. The Python built-in functions are documented at https://docs.python.org/3/library/functions.html.
- Optionally: you can use `print("\033c")` to clear the screen and make your game more playable.

  Example game, where blue is user input:

```
player 1, pick a number between 1 and 100: 67
<screen clears>
player 2, enter your guess:  42
Too small, try again
player 2, enter your guess:  88
Too big, try again
player 2, enter your guess: 67
You got it!
<program exits>
```

1.4 (optional)

- Improve your game by counting how many attempts were needed. Display the attempts count at the end of the game, when player 2 finds the correct number.
- Can you think of ways in which the first player can be naughty and make the second player take a lot of guesses?