TP 2

> *Presented by: Omar Allouch*

Ollama copilot and debugging

In the source codes you will find a file named `factorial.py`, it contains the file on which I used the copilot and debugging tools. However that file is the end result, so here's the initial content:

```python
def factorial(n):
    """
    Function to calculate the factorial of a number
    """
    return n * factorial (n - 1)

if __name__ == "__main__":
    number = int(input("Enter a number: "))
    result = factorial(number)
    print(f"The factorial of {number} is {result}")
```

In the video I slowly built the final version of the file, which can find the factorial of a number, and I used this ollama copilot to help me with the code completion and the debugging tool to help me with the errors.

File content description

package.json

This file is a `package.json` file, commonly used in JavaScript and Node.js projects. It contains the following sections:

Basic Information: Like the name, version, and description of the package. Other fields include the name of the author, license, and keywords. This helps identify the package and its purpose.

Exports: This section defines how the package can be imported or required:

- `"."`: The root export:
  - `"require"`: Points to the CommonJS version of the main file.
  - `"import"`: Points to the ES module version of the main file.
  - `"types"`: Points to the TypeScript declaration for the main file.
- `"./browser"`: Provides browser-specific builds for CommonJS and ES modules along with TypeScript types.
- `"./*"`: A fallback that allows direct access to any file under the package directory.

Scripts: This section defines common tasks that can be run via `npm run <script>`. For example, here we have scripts for building, testing, and linting the project.

Homepage and Repository: Indicate the project's homepage (website) and repository (GitHub URL).

Dependencies: Defines the dependencies required for the project. Here we only have one dependency (`whatwg-fetch`), which is a polyfill for the Fetch API.

DevDependencies: These packages are only used for development:

Purpose: This file is used for managing the `ollama` JavaScript library, a library intended to be used both in Node.js and browser environments. It includes build and test tools, along with configurations for exporting different module formats (CommonJS, ES modules).

pyproject.toml

This configuration file is written in TOML format and is used by Poetry, a tool for managing Python dependencies and packaging. It contains the following sections:

[tool.poetry] This section contains general information about the project, such as its name, version, description, and authors.

[tool.poetry.dependencies] This section lists the project's runtime dependencies, in this case: python and httpx.

[tool.poetry.group.dev.dependencies] This section defines the development dependencies—packages used during development and testing, just like the `devDependencies` section in a `package.json` file.

[tool.ruff] This section configures `ruff`, a rust-based python linter and formatter.

[tool.ruff.format] The rules to be applied by the formatter.

[tool.ruff.lint] The rules to be applied by the linter.

[tool.pytest.ini_options] A configuration section for `pytest`.

Purpose: This file is used to configure and manage the Python project for the "ollama" Python client. It defines dependencies, development tools, testing frameworks, code formatting, and linting rules, making it easier to develop, test, and maintain the project in a standardized way.

pom.xml

This XML file is a Maven `pom.xml` (Project Object Model) file, used to define the configuration for building, testing, and packaging a Java project. It contains the following sections:

Project Metadata This section includes basic information about the project, such as the group ID, artifact ID, version, and name.

Properties Defines project-specific properties that can be referenced elsewhere in the file. For example, the Java version, project build directory, and Maven compiler plugin version.

Developers Lists the main developers of the project.

Licenses Specifies the license under which the project is distributed.

Source Code Management (SCM) Defines the Git repository where the project's code is hosted.

Build Section Contains configuration for various Maven plugins.

Dependencies This section lists the libraries that the project depends on.

Profiles Defines different build profiles that activate specific configurations based on context.

Distribution Management This section configures how the project artifacts are distributed, including the Maven repository where the artifacts are deployed.

Purpose: This `pom.xml` configures the **Ollama4j** project to be built, tested, and packaged with Maven. It ensures the project uses the necessary plugins for source code management, documentation, testing (unit and integration), and artifact signing.