The Python script (program) is named `AA_OD_Analyzer` and has the following structure:

```
├── aa_od_analyzer.py
├── tests/
│   ├── test_aa_od_analyzer.py
├── example_data/
│   ├── input_data.csv
├── README.md
└── __init__.py
```

During its development, several fundamental principles of software engineering, such as modularity (separation of concerns), readability (including comments), robustness (fault-tolerance), and maintainability (e.g., ToDo tasks for future improvements), were considered. However, it is essential to note that this project is still a work in progress.

For example, in terms of modularity, separate functions have been created to deal with:

1. **Input Parsing:**
    - Reads the CSV file and parses the data into a data structure (e.g., Pandas DataFrame) for further analysis.
2. **Data Analysis:**
    - Calculates the histogram of optical densities using the data from the input file.
    - Group the data by amino acid sequences and calculate the minimum, maximum, and mean optical densities for each group.
3. **Output Generation:**
    - Creates a histogram plot using a data visualization library (e.g., Matplotlib) and saves it as a PNG file with clear annotations.
    - Generates a CSV file with the statistics (min, max, mean) for each amino acid group.

Other SE and development aspects include:

- **Error Handling:**
    - Implements some essential error handling functionality to gracefully handle various scenarios, such as missing files or invalid data (can be always expanded)
- **Logging and Debugging:**
    - Includes an optional logging statement to help troubleshoot issues and provide useful information during runtime. This can be also used as a part that demonstrates that the security has been considered in mind, for example, to warn about handling sensitive data (although this is not the case in this program).

- **Unit Tests:**
  - To demonstrate the TDD approach and robustness, the unit tests for the core functions of the program have been created to ensure their correctness and robustness.
  - Tests the input parsing, data analysis, and output generation functions with various test cases, including edge cases.
- **Documentation:**
  - It provides relatively comprehensive documentation for the program in the form of a README.md file, which includes the purpose, usage, input/output format, and dependencies.
  - Include information on how to run the program, ….
  -

The program already contains an example data (and other data can be added /used):

- example data in the `example_data` directory exist for running the model or test cases and validating the program's functionality.

What is missing as a concrete example in the current version is:

Deployment Strategy:

- Version control (Git) has been used to demonstrate this important aspect of development and deployment, enabling change tracking and facilitating collaborative development.

- However, the program has not been packaged as a Python package, which would facilitate easy distribution and installation, for example, using pip directly. This can be implemented later to showcase the skills and capabilities for packaging and distribution.

- Additionally, Continuous Integration (CI) and Continuous Deployment (CD) pipelines to automate testing and deployment processes have not been set up yet. Implementing these pipelines, for instance, using *GitHub Actions* can be done later to showcase the skills and capabilities for automating development workflows.