Python Exercises, Part 2

The following questions are to be prepared till the fifth session of the course. At the beginning of the session, a list is handed through, where each student will tick his/her individual achievements as a basis of grading. Students will be randomly picked from the present ones and have to present (defend) their solution. The student will be interviewed to his/her solution. In case there are clear doubts, that the student can not explain the presented work, no exercise of this sheet will be accounted towards the final grade. The solution will be further discussed with the plenum, for a common learning experience.

Ex. 6 Command-Line Tool Using `sys.argv`

- **Objective**: Create a Python script that acts as a command-line tool for basic text file manipulation. The script should support various operations like counting words, counting lines, and searching for specific words in a file. Use the `sys` package to read command-line arguments and implement these operations.
- **Requirements**:
  - The script should be run from the command line with the following arguments:
    - The first argument should be the filename.
    - The second argument should be the operation to perform: `count_words`, `count_lines`, or `search_word`.
    - If the operation is `search_word`, a third argument should specify the word to search for in the file.
  - The script should:
    - If the operation is `count_words`, count and print the number of words in the file.
    - If the operation is `count_lines`, count and print the number of lines in the file.
    - If the operation is `search_word`, print all lines that contain the specified word.
  - Handle errors if the file doesn't exist or incorrect arguments are provided.
- **Example Usage**:

```
python text_tool.py example.txt count_words
python text_tool.py example.txt count_lines
python text_tool.py example.txt search_word Python
```

Ex. 7 Advanced Command-Line Tool Using `argparse`

- **Objective**: Create a more advanced command-line tool for file management using the `argparse` package. The tool should support multiple operations such as copying, moving, and renaming files, and it should provide detailed help and error messages using `argparse`.
- **Requirements**:
  - The script should support the following subcommands:
    - `copy`: Copy a file from the source to the destination.
    - `move`: Move a file from the source to the destination.
    - `rename`: Rename a file to a new name.
  - Use `argparse` to:
    - Parse command-line arguments.
    - The source and destination file shall be passed using a corresponding argument (`--src` and `--dst`).
    - Handle missing arguments with clear error messages.
    - For copying files it shall be possible to pass one or more destinations (copy the file to all destinations).
    - Provide detailed help when requested using `-h` or `--help`.
    - The execution shall be invariant of the order of the arguments

  - Each operation should display an appropriate message (e.g., `File copied successfully`, `File moved successfully`, etc.).
  - To perform the file operations, you may use the `shutil` package.

- **Example Usage**:

```
python file_manager.py copy --src source.txt --dst
destination.txt
python file_manager.py move --dst destination.txt --src
source.txt
python file_manager.py rename --src old_name.txt --dst
new_name.txt
```