Scientific Programming
(Wissenschaftliches Programmieren)

Exercise 10

1. Input generator with arguments

- Change the script for random input generation (`geninput`), so that it reads the number of the variables and the number of the right-hand-side vectors from positional command line arguments.
- The script should also accept optional arguments for setting the lower and upper bounds of the generated numbers and for setting the directory in which the input file should be created.
- The help text (obtained with `./geninput -h`) should look similar to the example given at the end of the exercises.

2. Equation solver with arguments

- Change the script for solving the linear system of equations, so that it accepts an optional argument specifying the directory where the input for the equation can be found (and where the output will be written).
- The help text (obtained with `./linsolve -h`) should look similar to the example given at the end of the exercises.

3. Cleaning up the project

- Place your modules `solvers.py` and `io.py` into a package (`linsolver`) in order to avoid eventual name-conflicts with other Python-modules. Update the `import` statements accordingly.
- Check which entitites in your modules should be public. Make sure, all other entitites are marked as private (by using the "_" prefix).
- Make sure that each source file follows PEP 8.
- Make sure that each source file in your project has a high `pylint` score (at least 9.0).
- Make sure that your tests are working and are extensive enough (coverage).
- Make sure, the API-documentation generation via Sphinx works for both modules and produces the right documentation.
- Extend the README file, so that all information necessary to create an input for your program, to run your program and to iterprate the results are contained. (Alternatively, you could write a user manual in the Sphinx-documentation. In that case, refer to that documentation in the README file.)
- Commit the final stage and tag it as version 1.0.

Congratulation! Your scientific Python project has passed the first milestone of its development.

- Help page of geninput (obtained via `geninput -h`):

```
usage: geninput [-h] [-d DIRECTORY] [-l LOWER] [-u UPPER] NVARS NRHS

Input generator for the linsolve program.

positional arguments:
  NVARS                 Nr. of variables
  NRHS                  Nr. of rhs-vectors

optional arguments:
  -h, --help            show this help message and exit
  -d DIRECTORY, --directory DIRECTORY
                        Directory, where the input file should be created
                        (default .)
  -l LOWER, --lower LOWER
                        Lower bound for the random numbers (default: 0.0)
  -u UPPER, --upper UPPER
                        Upper bound for the random numbers (default: 1.0)
```

- Help page of linsolve (obtained via `linsolve -h`):

```
usage: linsolve [-h] [-d DIRECTORY]

Solves the linear sytem of equation AX=B. It requires an input file
"linsolve.in", containing the coefficient matrix A (each line of A should be
written into a separate line) followed by the right-hand-side vectors B. Each
vector in B (column vector in the equation) should be written into a separate
*row* (as row vector) in the input. The script writes a file "linsolve.out"
containing the solution vectors X as row vectors (one solution vector per
line) or an error message if the solver failed.

optional arguments:
  -h, --help            show this help message and exit
  -d DIRECTORY, --directory DIRECTORY
                        Directory, where input file is located and where
                        output should be written to (default: .)
```