Scientific Computing Using Python

Debugging

13th of June, 2023

Thomas Arildsen
tari@its.aau.dk

CLAAUDIA

Aalborg University

Edit-run and debugging

- Learning objectives for this lecture:
    - be able to use a sensible debugging workflow
    - be able to use a debugger

Edit-run and debugging

- (Write-)Edit-run and debugging.
  - Used in small projects of no more than a few hours of work
  - or smaller subtasks in a larger project.

Edit-run and debugging

The following is partly from [Lan16].

- Understand the problem (and the algorithm).

Edit-run and debugging

- Use pen and paper to sketch part of the code/algorithm.

Edit-run and debugging

- Work out some examples of input/output

Edit-run and debugging

- Consult manuals, book, Internet, help functions etc.

Edit-run and debugging

- Write the program

Edit-run and debugging

- Make use of error messages: `SyntaxError`, `NameError`, `TypeError`, `ValueError`, `IndexError`.

Edit-run and debugging

- Verify the program with simple input where you know the solution.

Edit-run and debugging

- Examine intermediate results and understand the program (possibly via a debugger).

Edit-run and debugging

Personal suggestions for really persistent bugs:

- Try to explain it to a colleague. Describing the problem to another may change your approach to the problem.
- Put it aside. Suddenly tonight you will see the light.
- Go for a walk.

Debugger

- A debugger can be useful to locate and remove bugs.
- Possible to use `pdb` (python standard library) from an IDE, ipython (`-d`) or command line (`-m pdb`)
- Useful commands:
  - `b` or `break`: insert breakpoint at `<file name>`:`<line>`/<function name>.
  - `c` or `continue`: continue to next breakpoint.
  - `s` or `step`: step to the next line of code.
  - `n` or `next`: step to the next line but does not enter a new function.
  - `h` or `help`: list of commands. `h` <command> gives details.
- Normal usage: walk through critical part of the code using `(p)rint`, `type()` etc. when in debug mode to investigate the behaviour of the program.
- IDE: investigate variable explorer. Breakpoints usually marked in the editor (spyder: F12 toggle breakpoints).
- As with other things in programming: Try it out!

Exercise I

- Try and have a look at `debugging_logical_error.py`.
- We will now try to use the debugger.

Debugger option — II

- Post-mortem debugging:
  - run in IPython and use “%debug” after error has been raised (can e.g. be used in the IDE spyder).
  - or run with `python -m pdb <filename>`.

Exercise II

- Try and have a look at `debugging_pm.py`.
- We will try to use post-mortem debugging.

Literature I

[Lan16] Hans Petter Langtangen. *A Primer on Scientific Programming with Python*. 5th ed. Springer, 2016. ISBN: 978-3-662-49887-3. DOI: 10.1007/978-3-662-49887-3.