

Exercise 08: Performance Optimization

The goal of this exercise is to learn the basic procedure of performance analysis by runtime measurement and profiling. The numerical approximation of the [Mandelbrot set](#) (known fractal) serves as an example.

Task 1: Execution time measurement and profiling 01_task1.py

1. Insert the necessary instructions for time measurement before and after suitable places in the code (`time.time()`, ...), to determine the required runtime for one iteration and the total runtime of the function `create_fractal`.
2. Determine the dependence of the runtime on the resolution (`resx`, `resy`).
3. Create a *profile* of the script `task1.py` using the command (see doc slides): `python -m cProfile ...`.
4. Filter the output for the relevant data using module `pstats`.

Aufgabe 2: numba

1. Install the `numba` package, see doc slides (enable conda first if necessary).
2. Create `02_task2.py` as a copy of `01_task1.py` and extend the script in such a way that the numerically complex part is accelerated using `numba`.

Note: see `example-code/numba.py`.

3. Determine by what factor the execution speed has changed.

Task 3: Cython, see directory example-code

1. Install the `cython` package using `pip install ...` or `conda install ...` (see slides).
2. Get an overview of the three `mandel-cython*` files in the `example-code` directory.
3. Compile the numeric module `mandel-cython.pyx` using the command
`python3 mandel-cython-setup.py build_ext --inplace`
and run `mandel-cython-main.py`.
4. Measure the runtime of generating the data (without visualization). Compare the result with task 2.
5. Optional: create a [histogram](#) over the values in `dataarray` and adjust the color scaling.

Task 4 (optional add-on):

1. Visualize another fractal using `numba` or `cython`, see for example. https://en.wikipedia.org/wiki/Julia_set.