Efficient Code with Cython

Why?

- Your code is slow due to calculations
- You don’t want to write C/C++ code and then bindings for that library
- You want your code to stay readable

Cython

Cython is a programming language that makes writing C extensions for the Python language **as easy as Python** itself. It aims to become a **superset** of the [Python] language which gives it high-level, object-oriented, functional, and dynamic programming. Its main feature on top of these is support for **optional static type declarations** as part of the language. The source code gets translated into **optimized C/C++ code** and compiled as Python extension modules. This allows for both very fast program execution and tight integration with external C libraries, while keeping up the high programmer productivity for which the Python language is well known.

https://cython.readthedocs.io/en/latest/src/quickstart/overview.html

Simple example

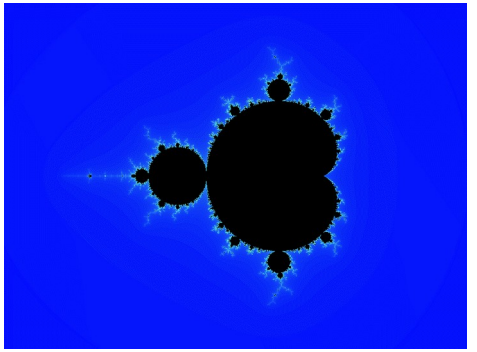
- Based on
  https://cython.readthedocs.io/en/latest/src/tutorial/cython_tutorial.html
- Create virtual env with “`Cython`”
- Create “`helloworld.pyx`” with “`print('hello world')`”
- Create “`setup.py`”

```
from setuptools import setup
from Cython.Build import cythonize
setup(ext_modules = cythonize("helloworld.pyx"))
```

- Compile with “`python setup.py build_ext --inplace`”
- Run with “`python -c 'import helloworld'`”

Mandelbrot set

- Mandelbrot set requires a fair bit of computation
  - $z_{n+1} = z_n^2 + c$, for every *c* in complex plane
  - paint black if bounded for max num of iterations
  - otherwise use exit iterations to determine color



Mandelbrot set (pure Python)

- Complex plane
  - X: [-2.5; 1.5]
  - Y: [-1.5; 1.5]
- Image size: 800x600
- Max iterations: 1000
- Palette: black and 255 colors (white to blue)
- Module: `mandelbrot.py`
- Run: `python -c "import mandelbrot.py"`
- Total run time (i7-9700KF): ~1.6s

Mandelbrot set (Cython)

- Rename `.py` to `.pyx`
- Add `setup.py`:

```
from setuptools import setup
from Cython.Build import cythonize
setup(
    install_requires=["numpy", "pillow", "colour", "cython"],
    ext_modules = cythonize("mandelbrot.pyx"))
```

- Compile: `python setup.py build_ext --inplace`
- Run: `python -c "import mandelbrot.py"`
- Total run time (i7-9700KF): ~1.1s

Mandelbrot set (Cython - optimized)

```
def calculate(cx, cy, max_iter):
    zx = 0
    zy = 0
    i = 0
    while (i < max_iter):
        i += 1
        zx_new = zx*zx - zy*zy + cx
        zy_new = 2*zx*zy + cy
        zx = zx_new
        zy = zy_new
        if (zx*zx + zy*zy > max_value):
            break
    return i
```

```
def calculate(float cx, float cy, int max_iter):
    cdef float zx, zy, zx_new, zy_new
    cdef int i
    zx = 0
    zy = 0
    i = 0
    while (i < max_iter):
        i += 1
        zx_new = zx*zx - zy*zy + cx
        zy_new = 2*zx*zy + cy
        zx = zx_new
        zy = zy_new
        if (zx*zx + zy*zy > max_value):
            break
    return i
```

Total run time (i7-9700KF): ~0.34s

Closing thoughts

- Does not optimize other libraries, only your modules that you specify
- Requires Cython
- You have to annotate your code
- People using your library require a build environment (wheels?)