Coding Example: The Mandelbrot Set (Python approach)

In this lesson, we are going to look at the Python implementation of the Mandelbrot set case study.

We'll cover the following

- Python Implementation
- Output

Python Implementation

A pure python implementation is written as:

main.py

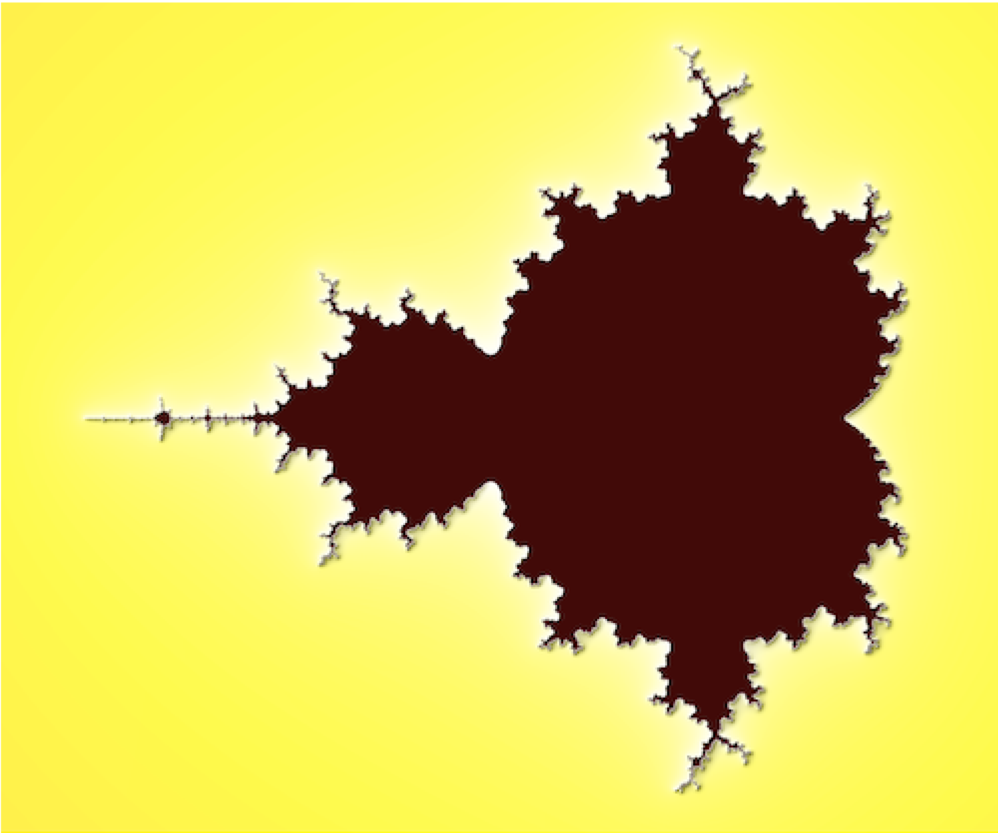
tools.py

```
# -----------------------------------------------------------
# From Numpy to Python
# Copyright (2017) Nicolas P. Rougier - BSD license
# More information at https://github.com/rougier/numpy-book
# -----------------------------------------------------------
import numpy as np

def mandelbrot_python(xmin, xmax, ymin, ymax, xn, yn, maxiter,
  def mandelbrot(z, maxiter):
    c = z
    for n in range(maxiter):
      if abs(z) > horizon:
        return n
      z = z*z + c
    return maxiter
  r1 = [xmin+i*(xmax-xmin)/xn for i in range(xn)]
  r2 = [ymin+i*(ymax-ymin)/yn for i in range(yn)]
  return [mandelbrot(complex(r, i),maxiter) for r in r1 for i i


def mandelbrot(xmin, xmax, ymin, ymax, xn, yn, itermax, horizon
    # Adapted from
```

```
    # https://thesamovar.wordpress.com/2009/03/22/fast-fractals
    Xi, Yi = np.mgrid[0:xn, 0:yn]
    Xi, Yi = Xi.astype(np.uint32), Yi.astype(np.uint32)
    X = np.linspace(xmin, xmax, xn, dtype=np.float32)[Xi]
    Y = np.linspace(ymin, ymax, yn, dtype=np.float32)[Yi]
    C = X + Y*1j
    N_ = np.zeros(C.shape, dtype=np.uint32)
    Z_ = np.zeros(C.shape, dtype=np.complex64)
    Xi.shape = Yi.shape = C.shape = xn*yn
```

Output

The output of the above code would look like this:



The interesting (and slow) part of this code is the mandelbrot function that actually computes the sequence $f_c(f_c(f_c...)))$. The vectorization of such code is not totally straightforward because the internal return implies a differential processing of the element. Once it has diverged, we don't need to iterate any more and we can safely return the iteration count at divergence. Now the problem is to do the same in NumPy.

In the next lesson, we'll solve this example using the numpy approach.