Mandelbrot Set

Here is a real world program written in NumPy and translated to Fortran.

Python

```python
import numpy as np

ITERATIONS = 100
DENSITY = 1000
x_min, x_max = -2.68, 1.32
y_min, y_max = -1.5, 1.5

x, y = np.meshgrid(np.linspace(x_min, x_max, DENSITY),
                   np.linspace(y_min, y_max, DENSITY))
c = x + 1j*y
z = c.copy()
fractal = np.zeros(z.shape, dtype=np.uint8) + 255

for n in range(ITERATIONS):
    print "Iteration %d" % n
    mask = abs(z) <= 10
    z[mask] *= z[mask]
    z[mask] += c[mask]
    fractal[(fractal == 255) & (-mask)] = 254. * n / ITERATIONS

print "Saving..."
np.savetxt("fractal.dat", np.log(fractal))
np.savetxt("coord.dat", [x_min, x_max, y_min, y_max])
```

Fortran

```fortran
program Mandelbrot
use types, only: dp
use constants, only: I
use utils, only: savetxt, linspace, meshgrid
implicit none

integer, parameter :: ITERATIONS = 100
integer, parameter :: DENSITY = 1000
real(dp) :: x_min, x_max, y_min, y_max
real(dp), dimension(DENSITY, DENSITY) :: x, y
complex(dp), dimension(DENSITY, DENSITY) :: c, z
integer, dimension(DENSITY, DENSITY) :: fractal
integer :: n
x_min = -2.68_dp
x_max = 1.32_dp
y_min = -1.5_dp
y_max = 1.5_dp

call meshgrid(linspace(x_min, x_max, DENSITY), &
    linspace(y_min, y_max, DENSITY), x, y)
c = x + I*y
z = c
fractal = 255

do n = 1, ITERATIONS
    print "('Iteration ', i0)", n
    where (abs(z) <= 10) z = z**2 + c
    where (fractal == 255 .and. abs(z) > 10) fractal = 254 * (n-1) / ITERATIONS
end do

print *, "Saving..."
call savetxt("fractal.dat", log(real(fractal, dp)))
call savetxt("coord.dat", reshape([x_min, x_max, y_min, y_max], [4, 1]))
end program
```

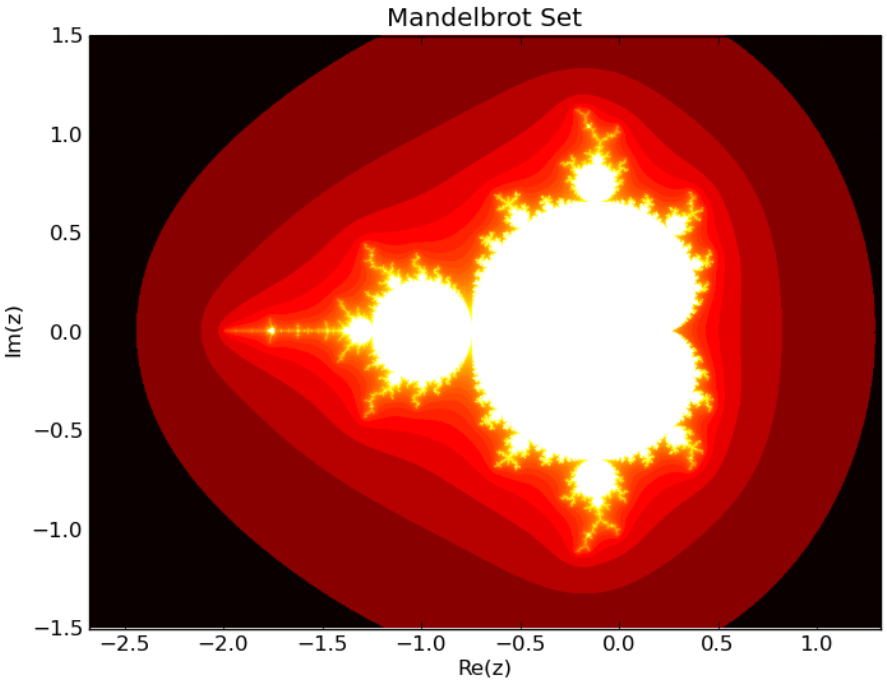
To run the Python version, you need Python and NumPy. To run the Fortran version, you need `types.f90`, `constants.f90` and `utils.f90` from the fortran-utils package. Both versions generate equivalent `fractal.dat` and `coord.dat` files.

The generated fractal can be viewed by (you need matplotlib):

```python
from numpy import loadtxt
import matplotlib.pyplot as plt

fractal = loadtxt("fractal.dat")
x_min, x_max, y_min, y_max = loadtxt("coord.dat")

plt.imshow(fractal, cmap=plt.cm.hot,
           extent=(x_min, x_max, y_min, y_max))
plt.title('Mandelbrot Set')
plt.xlabel('Re(z)')
plt.ylabel('Im(z)')
plt.savefig("mandelbrot.png")
```



Timings on Acer 1830T with gfortran 4.6.1 are:

	Python	Fortran	Speedup
Calculation	12.749	00.784	16.3x
Saving	01.904	01.456	1.3x
Total	14.653	02.240	6.5x