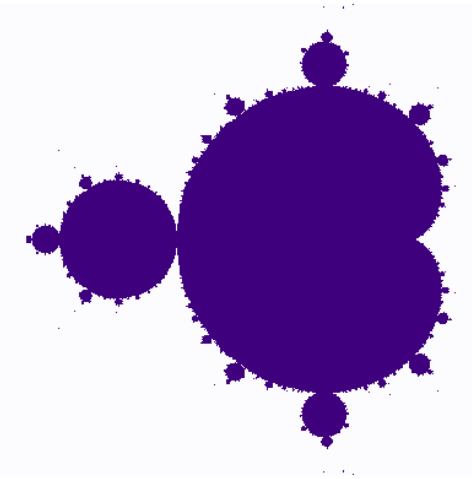
MAS212 The Mandelbrot set

Dr. Sam Dolan (s.dolan@sheffield.ac.uk)

1. Your friend is making a poster for an Outreach event. They have asked you to make a high-resolution (500×500) image of the **Mandelbrot set**.

The Mandelbrot set is the set of complex numbers c for which the function $f_c(z) = z^2 + c$ does not diverge when iterated from $z = 0$.

Here is the Mandelbrot set on the complex domain $-1.5 \leq \mathrm{Re}(c) \leq 0.5$, $-1 \leq \mathrm{Im}(c) \leq 1$:



2. You can use `numpy` arrays, together with universal functions, vectorization and broadcasting to help solve this problem quite efficiently.

(a) Use broadcasting to make a 2D array `c` out of two 1D arrays made with `np.linspace()`. The array `c` should have data type `dtype = np.complex128`. The array `c` should be evenly spaced across the domain $-1.5 \leq \mathrm{Re}(c) \leq 0.5$, $-1 \leq \mathrm{Im}(c) \leq 1$.

(b) Make a 2D array `z` of the same shape, initially set to zero values.

(c) Apply 100 iterations of the rule $z \to z^2 + c$. Use vectorization/ufuncs.

(d) Create a 2D boolean array, with value `True` if $|z_{ij}| \leq z_{\max}$, where z_{ij} is the corresponding element of the `z` array, and `False` otherwise.

(e) Plot an image of the 2D array, using `matplotlib.pyplot.imshow()`. Save and send to your friend.

```
# Python code to plot the Mandelbrot set, using
# numpy arrays, broadcasting, vectorization and universal functions.

import numpy as np
import matplotlib.pyplot as plt

# Part (a).
xmin = -1.5; xmax = 0.5; ymin = -1; ymax = 1.0; # the domain
npts = 501;  # number of points on each axis
xs = np.linspace(xmin,xmax,npts)  # real parts
ys = np.linspace(ymin,ymax,npts, dtype=np.complex128)*1j  # imag parts
 # Now make a 2D array by broadcasting two 1D arrays
cs = xs.reshape((1,npts)) + ys.reshape((npts,1))

# Part (b)
zs = np.zeros((npts,npts), dtype=np.complex128)

# Part (c).
nits = 100
for i in range(nits):
    zs = zs**2 + cs    # using a ufunc

# Part (d).
maxval = 100.0
mandelbrot = np.abs(zs) < maxval

# Part (e)
fig = plt.imshow(mandelbrot, cmap='Purples', origin='lower')
plt.axis('off')
plt.savefig("mandelbrot.png")
```