```
import numpy as np
import matplotlib.pylab as plt
%matplotlib inline
X0 = (0,0)
```

Problem 1

The Henon Map is found by obtaining the sequence of iterations

$$(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots$$

where

$$(x_{i+1}, y_{y+1}) = (a - \alpha x_i^2 + by_1, \gamma x_i)$$

Starting from $(x_0, y_0) = (0, 0)$,

(a) Write a function `HenonMap(x,y,a,b,alpha,gamma)` that returns $(x_i + 1, y_i + 1)$ given (x_i, y_i)

```
HenonMap = lambda X,a,b,alpha,gamma: np.array([a-alpha*X[0]*X[0]+b*X[1], gamma*X[0]])
```

(b) Write a function `HenonIterations(n)` that returns a list of the first n iterations of the Henon Map using $a = 1.4$, $b = 0.3$, $\alpha = 1$, $\gamma = 1$.

```
def HenonIterations(n,X0,a,b,alpha,gamma):

        iterations = np.zeros((n+1,2))

        for i in range(n):
                i += 1
                iterations[i] = HenonMap(iterations[i-1],a,b,alpha,gamma)

        return iterations
```

(c) Use Pylab and/or MatPlotLib to plot the Henon Attractor (a plot of the collection of dots you collected in the previous step). You should plot at least 10,000 points to get a good picture of the attractor.

```
n20000 = HenonIterations(20000,X0,1,1,1.4,0.3)
plt.plot(n20000[100:,0],n20000[100:,1],'.',color='green')
```

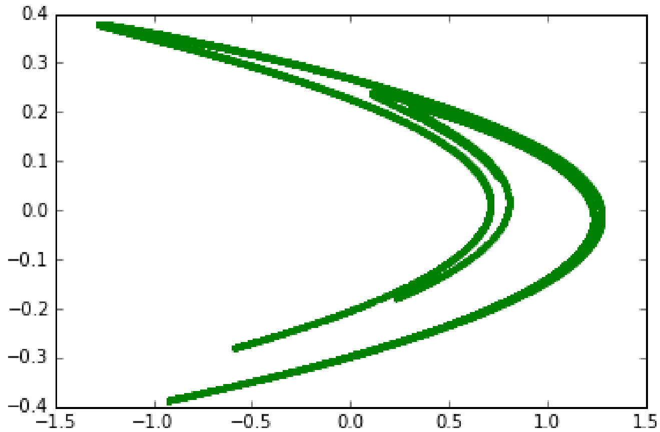


Figure 1: Henon Attractor, $\alpha = 1.4, \gamma = 0.3$

Problem 2

An fractal can be obtained from the Henon Map in the previous problem, as follows. Starting with the function HenonMap that you wrote in the first problem,

(a) Let `H(x,y,gamma)=HenonMap(x,y,1,1,0.2,gamma)`

```
H = lambda X,gamma: HenonMap(X,1,1,0.2,gamma)
```

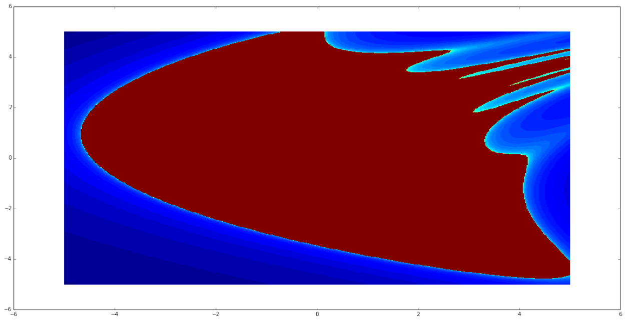
(b) Write a function `HenonIterate(x0, y0, n, gamma)` that iterates the on `H(x,y,gamma)` for n iterations and returns one of the following: a value of i (the number of iterations) if at any point during the iteration process $x_i^2 + y_i^2 > 100$ (stop and return as soon as this happens); or else if all n iterations are completed without ever having $x_i^2 + y_i^2 > 100$ for any i, return the value of n.

```
def HenonIterate(n,X0,gamma):
        x,y = X0
        for i in range(n):
                x,y = H((x,y),gamma)
                if ((x**2+y**2 > 100)): return i+1
        return n

HenonIterate(10,(-5,-5),1.03)
```

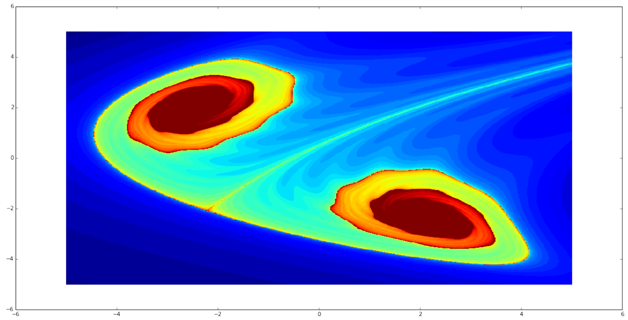
(c) Now write a function that will call `HenonIterate` for a range of values of (x_0, y_0) for on the square $[-5, 5] \times [-5, 5]$ and store the results in a square matrix (i.e., list of lists, where each list is a row of the matrix). Then use `pylab.figure` followed by `pylab.figure(2)` to plot the results. Use $\gamma = 1.03$, then see what happens if you vary gamma by a few percent. A good value to use for n is 100, but you may want to use a smaller value for debugging and testing. By varying your parameters you may get either a whirlpool or a sting-ray shaped fractal.

```
def HenonIterateMap(iterations,density,gamma):
        x = np.linspace(-5,5,density)
        y = np.linspace(-5,5,density)

        map = np.zeros((density,density))

        for i in range(density):
                for j in range(density):
                        val = HenonIterate(iterations,(x[i],y[j]),gamma)
                        map[i,j] = val
        plt.figure(figsize=(20,10))
        plt.pcolormesh(x,y,map)

        return map
```

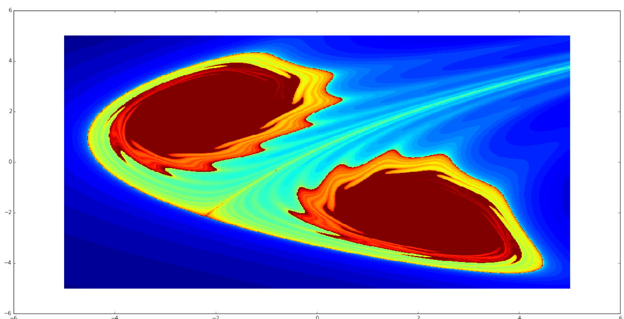
HenonIterateMap(50,500,.98)



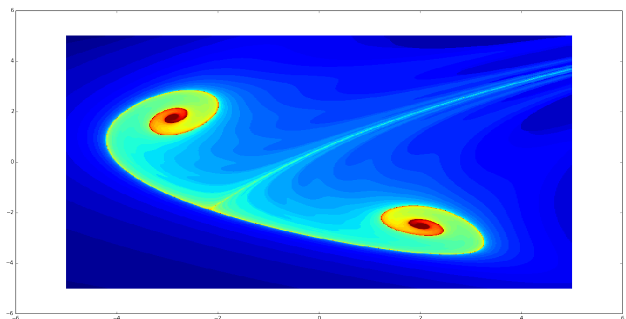
HenonIterateMap(50,500,1.11)



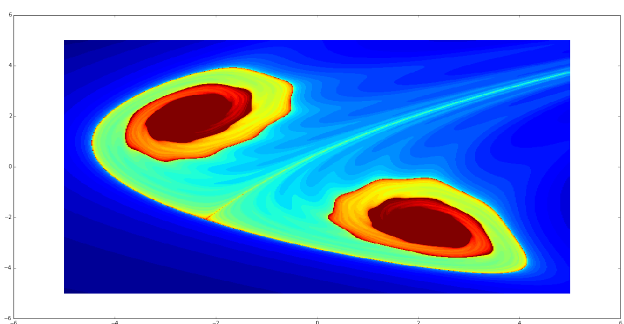
HenonIterateMap(50,500,1.03)



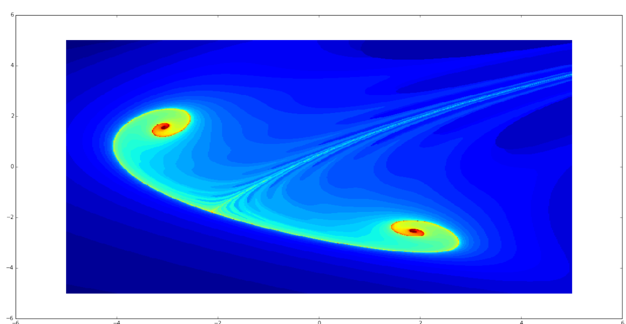
HenonIterateMap(50,500,1.15)



HenonIterateMap(50,500,1.06)



HenonIterateMap(50,500,1.2)



Problem 3

Write a python program to find the Julia set for $f(z) = z^2 + C$ where $C = 0.360284 + 0.100376j$

(a) Write a function to find $f(z)

```
C = 0.360283 + 0.100376j
julia = lambda z: z**2 + C
```

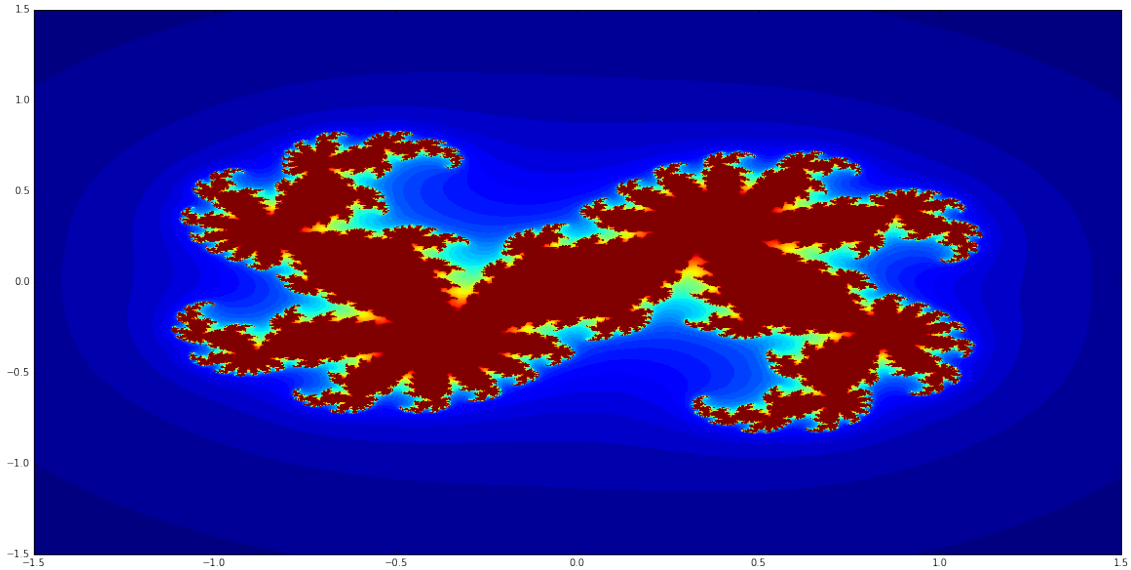
(b) Write a function `iterate(n, z0, zmax)` that iterates on f for n iterations starting at $z0$, using fixed point iteration, returning the number of iterations when $|z| >$ `zmax` for the first time, or returning n if the absolute value $|z|$ never exceeds `zmax`.

```
def juliaIterate(n,z0,zmax):
        for i in range(n):
                z0 = julia(z0)
                if (abs(z0) > zmax): return i+1
        return n
```

(c) Collect the values produced by the above iteration in a square $[-1.5, 1.5] \times [1.5, 1.5]$. Good values for final pictures are $n = 200$ and `zmax`$= 10$, but you should use a smaller n (say $n = 10$) for debugging.

```
def juliaIterateMap(iterations,density,zmax):
        x = np.linspace(-1.5,1.5,density)
        y = np.linspace(-1.5,1.5,density)

        map = np.zeros((density,density))

        for i in range(density):
                for j in range(density):
                     val = juliaIterate(iterations,(complex)(x[i],y[j]),zmax)
                     map[i,j] = val
        plt.figure(figsize=(20,10))
        plt.pcolormesh(x,y,map)

        return map
```

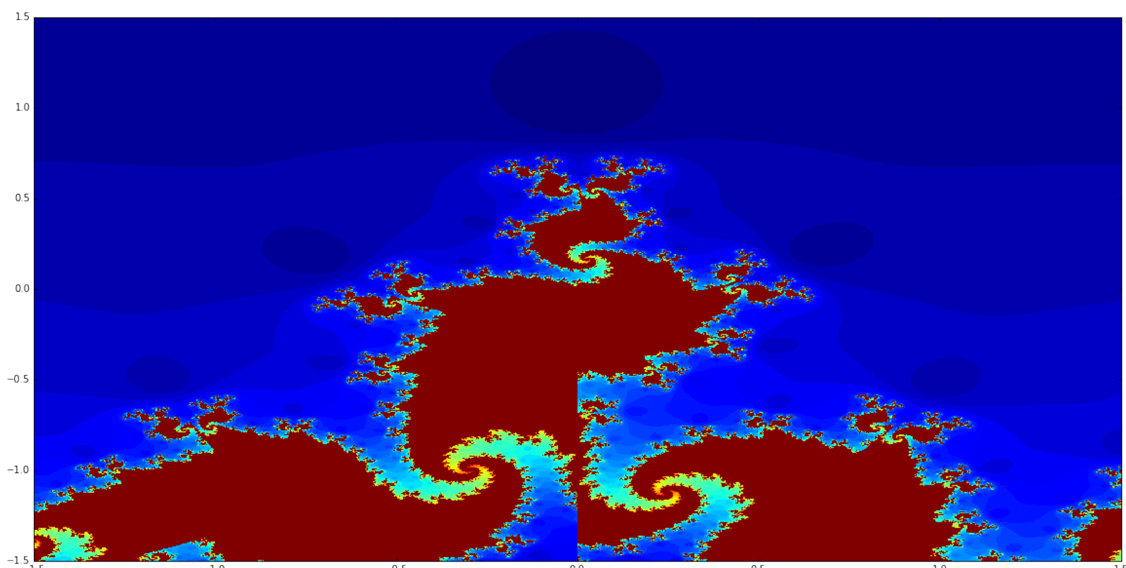
`juliaIterateMap(50,1000,10)`



(d) Look up the function for another fractal you like on the internet (or just experiment) and make a plot of its Julia set using the same algorithm (all you have to do is change the value of C in part (a)).

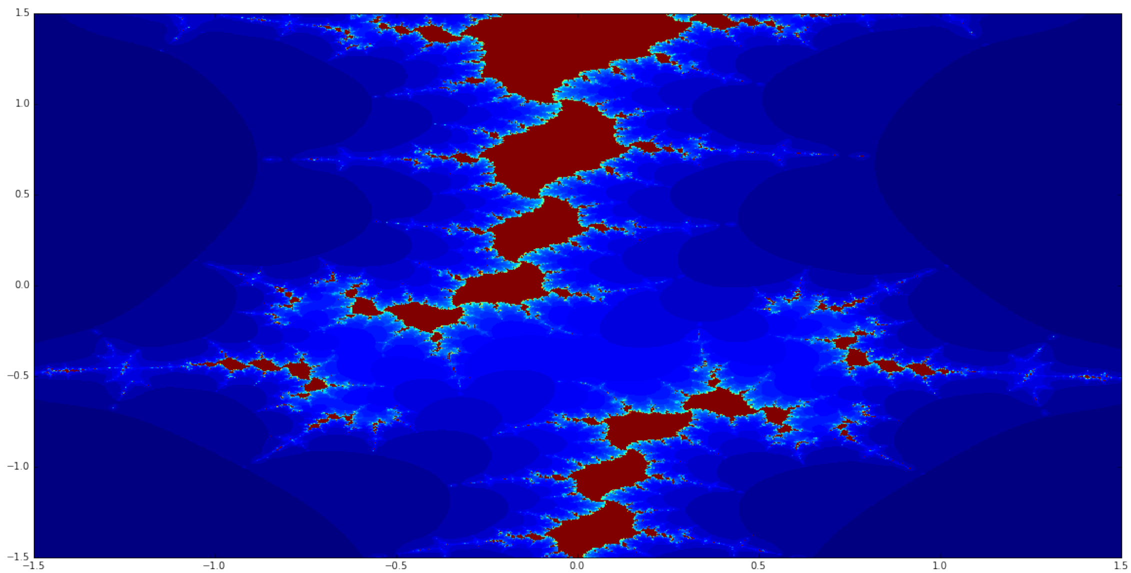
Variant 1, `juliaIterateMap(50,1000,10)`

```
julia = lambda z:  (z**2+z)/np.log(z) +0.268+0.060j
```



Variant 2, `juliaIterateMap(50,1000,10)`

```
julia = lambda z: z*np.sin(z)+z*np.cos(z) + C
```



Variant 3, `juliaIterateMap(50,1000,10)`

```
julia = lambda z: np.sinh(z)+np.cosh(z)
```