Práctica 2: Fractales de Newton

Ingrediente 1: Funciones que admiten otras funciones como parámetros

Al definir una función, algunos de sus parámetros pueden ser funciones que se llaman dentro de nuestra función

In [4]:
```
def asigna(f, n):
    '''
    Esta función devuelve una lista de tamaño n
    que consiste en aplicar el valor de f a los
    valores entre 0 y n-1
    '''
    lista = []
    i = 0
    while i< n:
        lista.append(f(i))
        i += 1
    return lista
```

In [5]:
```
def f(x):
    return x + 1
def g(x):
    return x**2
```

In [6]:
```
asigna(f,10)
```

Out[6]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [7]:
```
asigna(g,10)
```

Out[7]:
```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [8]:
```
def compone(f,g,x):
    '''
    compone las funciones f y g
    '''
    return f(g(x))
```

In [9]:
```
compone(f,g,10)
```

Out[9]:
```
101
```

In [10]:
```
compone(g,f,10)
```

Out[10]:
```
121
```

Ingrediente 2: Números complejos

Manejar números complejos con Python es sencillo. Basta utilizar la función `complex`

In [11]:
```
z = complex(2,3)
z
```

Out[11]: (2+3j)

Observa que la tradicional "i" se escribe como "j".

Podemos recuperar la parte real y la parte imaginaria.

In [12]:
```
print z.real, z.imag
```

```
2.0 3.0
```

In [13]:
```
z == z.real + (z.imag)*1j
```

Out[13]: True

In [14]:
```
#Calcular el módulo de un número complejo
abs(z)
```

Out[14]: 3.605551275463989

Las operaciones que hemos efectuado hasta ahora con números reales (flotantes) las podemos ejecutar ahora con números complejos

In [15]:
```
y = complex(0,-2)
y
```

Out[15]: -2j

In [16]:
```
z + y
```

Out[16]: (2+1j)

In [17]:
```
z - y
```

Out[17]: (2+5j)

In [18]:
```
z*y
```

Out[18]: (6-4j)

In [19]:
```
z/y
```

Out[19]: (-1.5+1j)

In [20]:
```
y**2
```

Out[20]: (-4-0j)

In [21]:
```
#podemos utilizar las funciones anteriormente definidas
compone(f,g,y)
```

Out[21]:
```
(-3+0j)
```

Ingrediente 3: el módulo PIL

Ver http://www.pythonware.com/products/pil/

Con la librería PIL podemos manejar imágenes en Python. Para nuestro propósito, basta utilizar unas pocas instrucciones, que se entienden bien a través de ejemplos.

In [23]:
```
#Ejemplo 1
from PIL import Image
red = (255,0,0)
green = (0,255,0)
blue = (0,0,255)
width = 300
height = 200
margin = 10
box_width = width + margin
box_height = height + margin
k=50
i = Image.new("RGB", (width,height), "red")
for x in range(height):
    i.putpixel((x,x), green)
    i.putpixel((x+k,x), blue)
i.show()
```

In [25]:
```
im = Image.new("RGB", (box_width,box_height), "white")
#represento la grafica de x^2 entre -1 y 1
for x in xrange(width):
    #transformo el punto en un número entre -1 y 1
    a = (2.0/width)*x - 1
    b = a**2
    #transformo b, situado entre 0 y 1, en un pixel entre 0 y height
    y = int(height*(1-b))
    #print a, b, x, y
    im.putpixel((x,y), green)
im.show()
```

In [ ]:
```

```