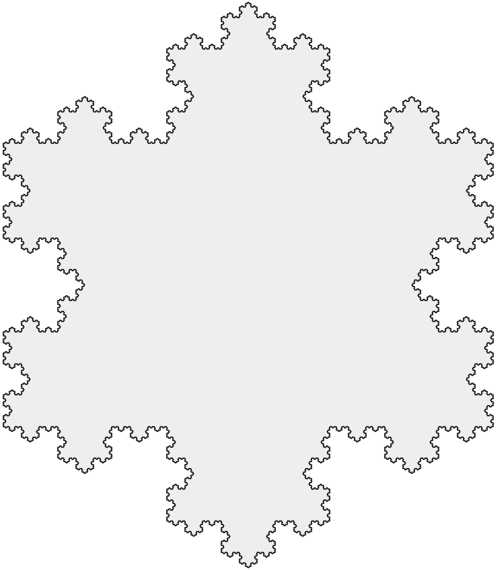
Fractal animation

Grégory Châtel

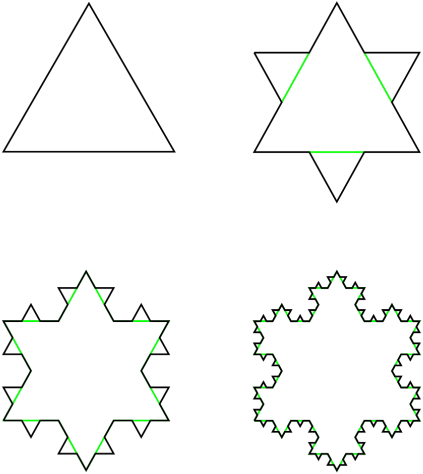
@rodgzilla
github.com/rodgzilla

October 18th, 2018

What are fractals?

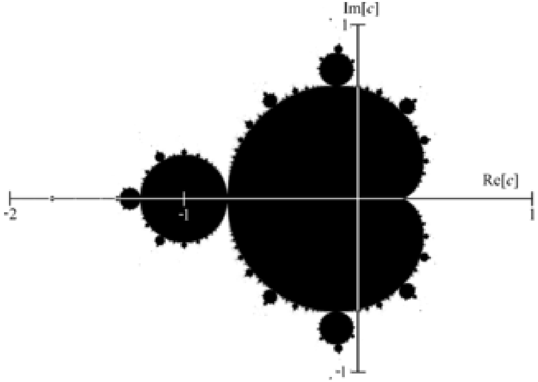


Koch snowflake



Pythagora's tree demo

Mandelbrot set



Mandelbrot set

Complex numbers and complex plane

$$c = ai + b$$
$$i^2 = -1$$

The *modulus* of a complex number is its euclidiean distance to 0:

$$c = ai + b$$
$$|c| = \sqrt{a^2 + b^2}$$

```
In [1]: c = complex(1, 2)

In [2]: c
Out[2]: (1+2j)

In [3]: c + complex(0,3)
Out[3]: (1+5j)

In [4]: c * c
Out[4]: (-3+4j)
```

Mandelbrot set

Function to iterate

The formula that generates everything is the following one:

$$z_0 = c$$
$$z_{n+1} = z_n^2 + c$$

For each pixel (x, y) of the screen, we compute the corresponding complex number $c_{x,y}$.

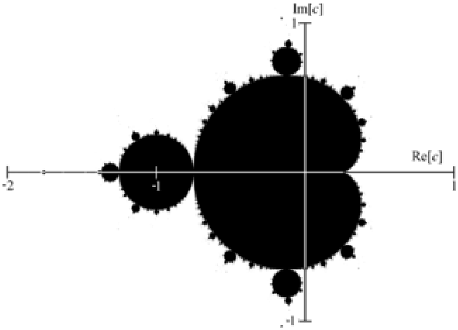
Now we compute a fixed number of terms of the sequence above starting with $z_0 = c_{x,y}$.

$$\begin{aligned}
z_0 &= c_{x,y} \\
z_1 &= c_{x,y}^2 + c_{x,y} \\
z_2 &= z_1^2 + c_{x,y} = (c_{x,y}^2 + c_{x,y})^2 + c_{x,y} \\
\dots \\
z_{100} &= z_{99}^2 + c_{x,y}
\end{aligned}$$

In Python:

```
mandelbrot_function = lambda z, c: z ** 2 + c
```

While we are iterating the formula for a pixel (x, y), if we find a step i for which $|z_i| > 2$ then we draw the (x, y) pixel white, otherwise we draw it black.



Mandelbrot set

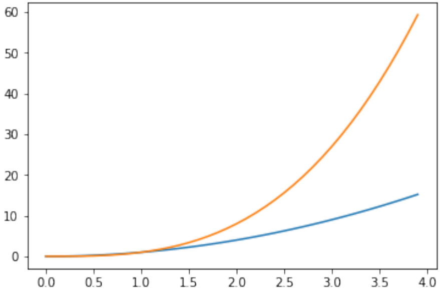
Source code

```
def draw_mandelbrot(f, max_iter, width, height):
    for y in range(height):
        for x in range(width):
            real = 3 * x / width - 2
            imag = 2 * y / height - 1
            c = complex(real, imag)
            z = c
            for i in range(max_iter):
                z = f(z, c)
                if z.real ** 2 + z.imag ** 2 > 4:
                    pixel(x, y, 'white')
                    break
            else:
                pixel(x, y, 'black')
```

Mandelbrot set zoom animation

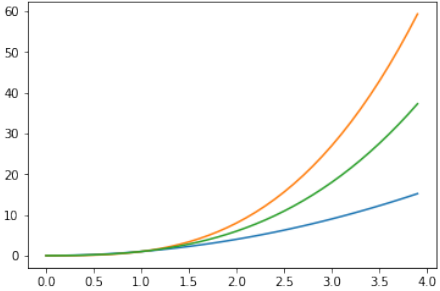
Function interpolation with Python

```
f_square = lambda x: x ** 2
f_cube   = lambda x: x ** 3
```



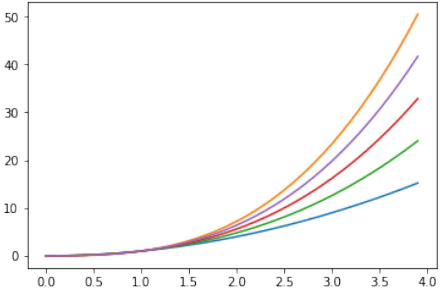
Function interpolation with Python

```
f_square = lambda x: x ** 2
f_cube   = lambda x: x ** 3
f_inter  = lambda x: (f_square(x) + f_cube(x)) / 2
```



Function interpolation with Python

```
n_int    = 5
int_fns = []
for i in range(n_int):
    new_fn = lambda x, i = i, n_int = n_int: (
        (n_int - i) * f_square(x) +
        i * f_cube(x)
    ) / n_int
    int_fns.append(new_fn)
```



Mandelbrot set formula animations