

1 Functional graphics

```
1 import numpy as np
2 from matplotlib import pyplot as plt
3 from functools import partial, reduce
4 from operator import add, mul

1 I = lambda t: t
2 const = lambda c: lambda _: c
3 compose = lambda *fs: reduce(lambda f, g: lambda *x: f(g(*x)), fs)
4 pure = lambda t: [t]
5 iterate = lambda n: lambda f: lambda x: iterate(n-1)(f)(f(x)) if n > 0 else x
6 mcompose = lambda *fls: reduce(lambda fl, gl: [compose(f, g) for f in fl for g in gl], fls)

1 def apply(f, *args, **kwargs):
2     return f(*args, **kwargs)

1 lapply = lambda *fs: lambda t: np.array([f(t) for f in fs])

1 origin = lapply(const(0), const(0))

1 line = lapply(lambda t: 2*t - 1, const(0))

1 ltrans = lambda T: lambda x: T @ x
2 rot = lambda theta: ltrans(np.array([[np.cos(theta), np.sin(theta)], [-np.sin(theta), np.cos(theta)]]))
3 rotcw = rot(np.pi / 2)
4 rotccw = rot(-np.pi / 2)

1 rotccw(np.array([-1, 1]))

array([-1., -1.])

1 split = lambda *fs: lambda t: fs[len(fs) - 1 if t == 1 else int(t * len(fs))](t*len(fs) % 1)

1 curve = split(compose(rot(1), line), line)

1 plt.figure(figsize=(5, 5))
2 plt.plot(*zip(*map(curve, np.linspace(0, 1))), 'wo')
3 plt.axis('off');
```

1.1 2D regions

```
1 pair = lambda *fs: lambda a: np.array([f(*a) for f in fs])
2 constv = lambda a, b: pair(const(a), const(b))
3 square = pair(lambda u, v: 2*u - 1, lambda u, v: 2*v - 1)
4 circle = pair(lambda u, v: v * np.cos(2*np.pi*u), lambda u, v: v * np.sin(2*np.pi*u))
5 rapply = lambda g: lambda f: lambda *args: g(*f(*args))

1 trans = lambda *cs: partial(add, np.array(cs))
2 resize = lambda *cs: partial(mul, np.array(cs))
3 scale = lambda c: resize(c, c)

1 transforms = [
2     circle,
3     scale(0.5),
4     trans(1/2, 1/2),
5 ] * 2
6 transforms.reverse()
7 shape = compose(*transforms)
8 shape([1/2, 1/2])

array([0.5 , 0.75])

1 n = 100
2 uniform = np.linspace(0, 1, n)
```

Grid

```
1 us, vs = np.meshgrid(uniform, uniform)
```

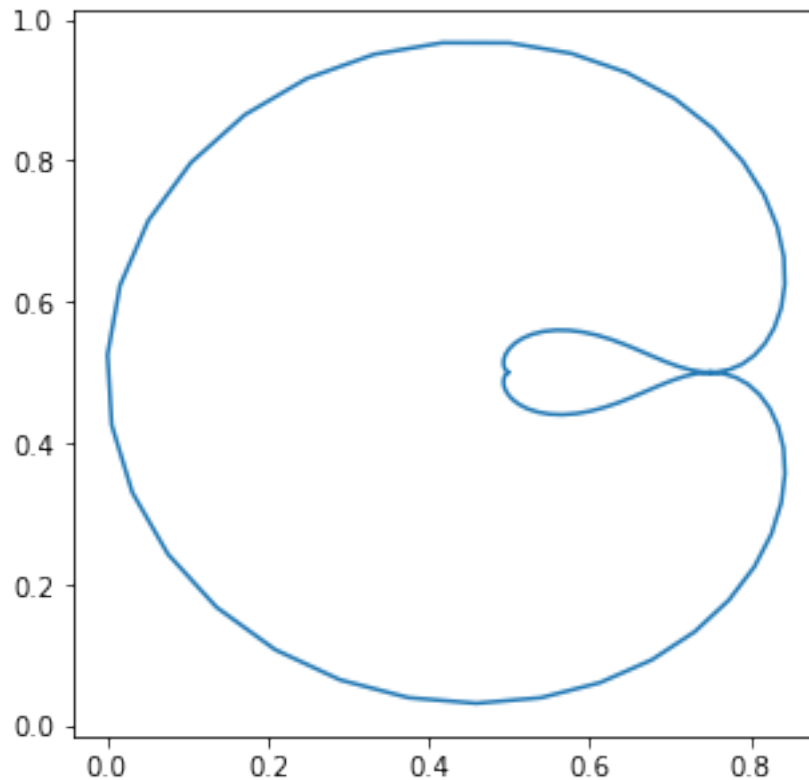
Boundary

```
1 us = np.hstack([uniform, np.ones(n), 1 - uniform, np.zeros(n)])  
2 vs = np.hstack([np.zeros(n), uniform, np.ones(n), 1 - uniform])
```

```
1 us, vs = uniform, np.ones(n)
```

Draw the shape

```
1 xs, ys = zip(*map(shape, zip(us.flat, vs.flat)))  
2 plt.figure(figsize=(5, 5))  
3 plt.plot(xs, ys, '-');  
4 # plt.axis('off');
```



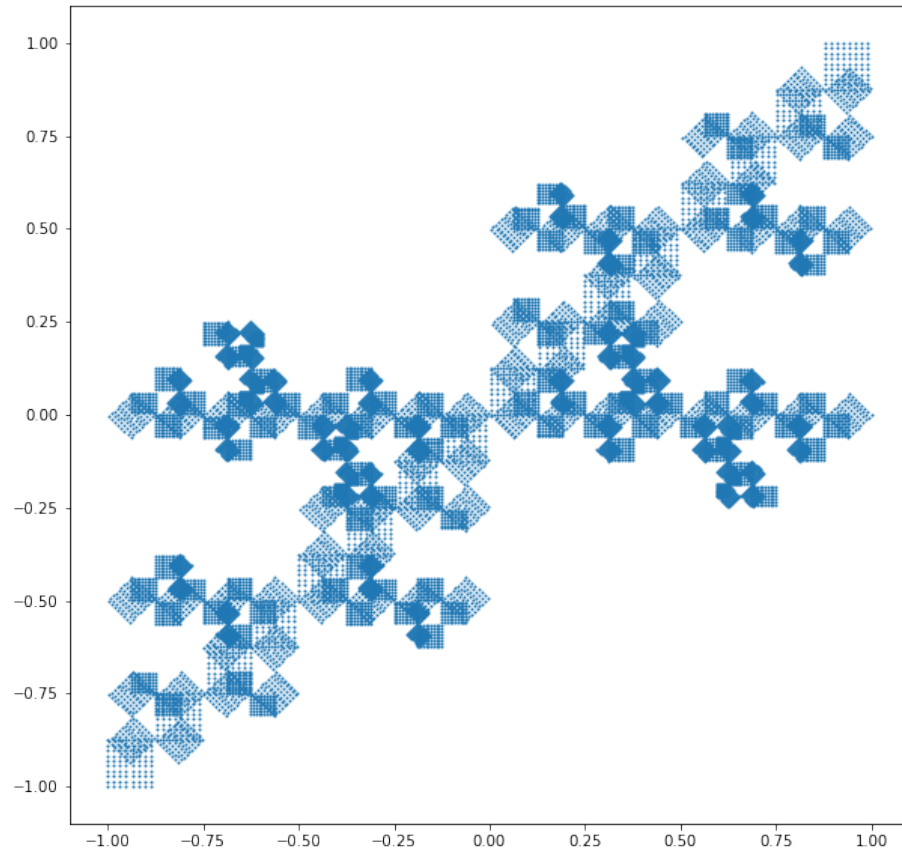
Now fractals

```
1 double = lambda f, g: lambda a: compose(f, resize(2, 1))(a) if a[0] < 1/2 else compose(g, resize(2, 1),  
↪ trans(-1/2, 0))(a)  
  
1 frac = lambda f: (lambda g: compose(scale(1/2), double(g, compose(rot(np.pi/4), scale(1/np.sqrt(2))),  
↪ g))))(double(compose(trans(-1, -1), f), compose(trans(1, 1), f)))  
2 shape = iterate(4)(frac)(square)
```

```

1 us, vs = np.meshgrid(np.linspace(0, 1, 2000), np.linspace(0, 1, 10))
2 xs, ys = zip(*map(shape, zip(us.flat, vs.flat)))
3 plt.figure(figsize=(10, 10))
4 plt.plot(xs, ys, 'o', markersize=1);

```



Now use a graphics library like a regular person.

```

1 import cairocffi as cairo
2 from PIL import Image

1 def draw_point(ctx, p, r=0.005):
2     ctx.arc(*p, r, 0, 2*np.pi)
3     ctx.fill()
4     ctx.stroke()
5
6 def draw_line(ctx, line):
7     p1, p2 = line
8     ctx.move_to(*p1)
9     ctx.line_to(*p2)
10    ctx.stroke()

1 frac = lambda geom: lambda *fs: lambda ls: [geom(f, l) for l in ls for f in fs]
2 linegeom = lambda f, line: (f(line[0]), f(line[1]))
3 pointgeom = lambda f, point: f(point)

```

Draw a mandala-like fractal

```
1 # drawgeom, mapgeom, initgeom = draw_line, linegeom, [[-1, 0], [1, 0]]
2 drawgeom, mapgeom, initgeom = draw_point, pointgeom, [[0, 0]]
3
4 langle = np.pi / 8
5 n = 4
6 geoms = iterate(n)(frac(mapgeom)(*mcompose(
7     [I, rot(np.pi / 2)],
8     [I, resize(-1, 1)],
9     [trans(1, 0)],
10    [I, resize(-0.5, 1)],
11    [I, resize(1, -1)],
12    [compose(trans(0, np.tan(langle)), rot(-langle), scale(1 / (2*np.cos(langle))), trans(-1, 0))]
13 ))) (initgeom)
14
15 width, height = 300, 500
16 surface = cairo.PDFSurface('mandala.pdf', width, height)
17 ctx = cairo.Context(surface)
18 ctx.translate(width / 2, height / 2)
19 side = 0.5 * min(width, height)
20 ctx.scale(side, side)
21 ctx.move_to(0, 0)
22
23 ctx.set_source_rgba(0, 0, 1, 0.25)
24 ctx.set_line_width(1e-4)
25 ctx.scale(0.5, 0.5)
26 for geom in geoms:
27     drawgeom(ctx, geom)
28
29 ctx.set_source_rgba(1, 0, 0)
30 ctx.arc(0, 0, 0.01, 0, 2*np.pi)
31 ctx.fill()
32 ctx.stroke()
33
34 surface.finish()
```

Distort a region

```
1 def draw_poly(ctx, poly):
2     ctx.move_to(*poly[0])
3     for p in poly[1:]:
4         ctx.line_to(*p)
5     ctx.set_source_rgba(1, 0, 1, 0.5)
6     # ctx.fill()
7     ctx.fill_preserve()
8     ctx.set_source_rgba(0, 1, 0, 0.5)
9     ctx.stroke()
10
11 polygeom = lambda f, poly: [f(p) for p in poly]
12
13 drawgeom, mapgeom, initgeom = (
14     draw_poly,
15     polygeom,
16     # [[circle([a, 1]) for a in np.linspace(0, 1, 4)]]
17     [
18         [[0,0], [1,0], [1/2,1]],
19     ]
20 )
```

```

7         [[1,1], [1,0], [1/2,1]],
8         [[0,1], [1/2,1], [0,0]]
9     ]
10 )
11
12 geoms = iterate(8)(frac(mapgeom)(*mcompose(
13     [resize(1, 1/2), compose(scale(1/2), trans(0, 1)), compose(scale(1/2), trans(1, 1))]
14 ))) (initgeom)
15 geoms = iterate(2)(frac(mapgeom)(*mcompose(
16     [compose(scale(0.5), trans(1, 1), circle)]
17 ))) (geoms)
18 # geoms = frac(mapgeom)(circle)(geoms)
19 # geoms = iterate(3)(frac(mapgeom)(*mcompose(
20 #     [I, rot(np.pi / 2)],
21 #     [I, resize(-1, 1)],
22 #     [trans(1, 0)],
23 #     [I, resize(-0.5, 1)],
24 #     [I, resize(1, -1)],
25 #     [compose(trans(0, np.tan(langle)), rot(-langle), scale(1 / (2*np.cos(langle))), trans(-1, 0))],
26 # ))) (geoms)
27
28 width, height = 300, 500
29 surface = cairo.PDFSurface('distort.pdf', width, height)
30 ctx = cairo.Context(surface)
31 ctx.translate(width / 2, height / 2)
32 side = 0.5 * min(width, height)
33 ctx.scale(side, -side)
34
35 ctx.scale(0.5, 0.5)
36
37 # ctx.set_source_rgba(0, 0, 0, 0.02)
38 # draw_poly(ctx, [[0,0],[0,1],[1,1],[1,0]])
39 # ctx.set_source_rgba(0, 1, 0, 0.5)
40 # ctx.arc(1/2, 1/2, 0.01, 0, 2*np.pi)
41 # ctx.fill()
42
43 ctx.set_line_width(1e-4)
44 ctx.set_source_rgba(0, 0, 1, 0.25)
45 for geom in geoms:
46     drawgeom(ctx, geom)
47
48 ctx.set_source_rgb(1, 0, 0)
49 ctx.arc(0, 0, 0.01, 0, 2*np.pi)
50 ctx.fill()
51
52 surface.finish()

```

A line fractal

```

1 drawgeom, mapgeom, initgeom = (
2     draw_line,
3     linegeom,
4     [([-1, 0], [1, 0])]
5 )
6 m = 3
7 geoms = iterate(4)(frac(mapgeom)(*mcompose(
8     [I, *mcompose(
9         [compose(scale(1), rot(np.pi/m))],

```

```

10         [rot(2*np.pi*i/m) for i in range(m)],
11         [compose(scale(-0.5), trans(1, 0))]]
12     ),
13     mcompose(
14         [rot(2*np.pi*i/m) for i in range(m)],
15         [compose(scale(-0.5), trans(1, 0))]]
16     )
17 ))(initgeom)

1 width, height = 300, 500
2 surface = cairo.PDFSurface('linefractal.pdf', width, height)
3 ctx = cairo.Context(surface)
4 ctx.translate(width / 2, height / 2)
5 side = 0.5 * min(width, height)
6 ctx.scale(side, -side)
7
8 ctx.scale(0.75, 0.75)
9
10 # ctx.set_source_rgba(0, 0, 0, 0.02)
11 # draw_poly(ctx, [[0,0],[0,1],[1,1],[1,0]])
12 # ctx.set_source_rgba(0, 1, 0, 0.5)
13 # ctx.arc(1/2, 1/2, 0.01, 0, 2*np.pi)
14 # ctx.fill()
15
16 ctx.set_line_width(1e-4)
17 ctx.set_source_rgba(0, 0, 1, 0.75)
18 for geom in geoms:
19     drawgeom(ctx, geom)
20
21 ctx.set_source_rgb(1, 0, 0)
22 ctx.arc(0, 0, 0.01, 0, 2*np.pi)
23 ctx.fill()
24
25 surface.finish()

```