1 Functional graphics

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

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

```
def apply(f, *args, **kwargs):
    return f(*args, **kwargs)
```

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

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

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

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

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

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

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

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

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

1.1 2D regions

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

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

```
transforms = [
    circle,
    scale(0.5),
    trans(1/2, 1/2),
] * 2
transforms.reverse()
shape = compose(*transforms)
shape([1/2, 1/2])
```

```
array([0.5 , 0.75])
```

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

Grid

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

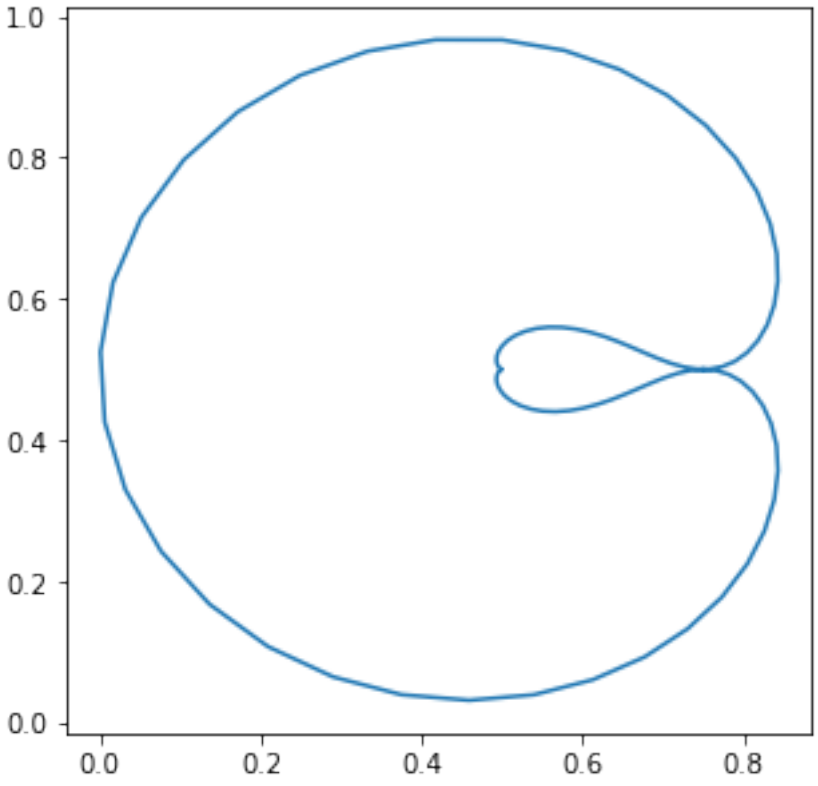
Boundary

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

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

Draw the shape

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

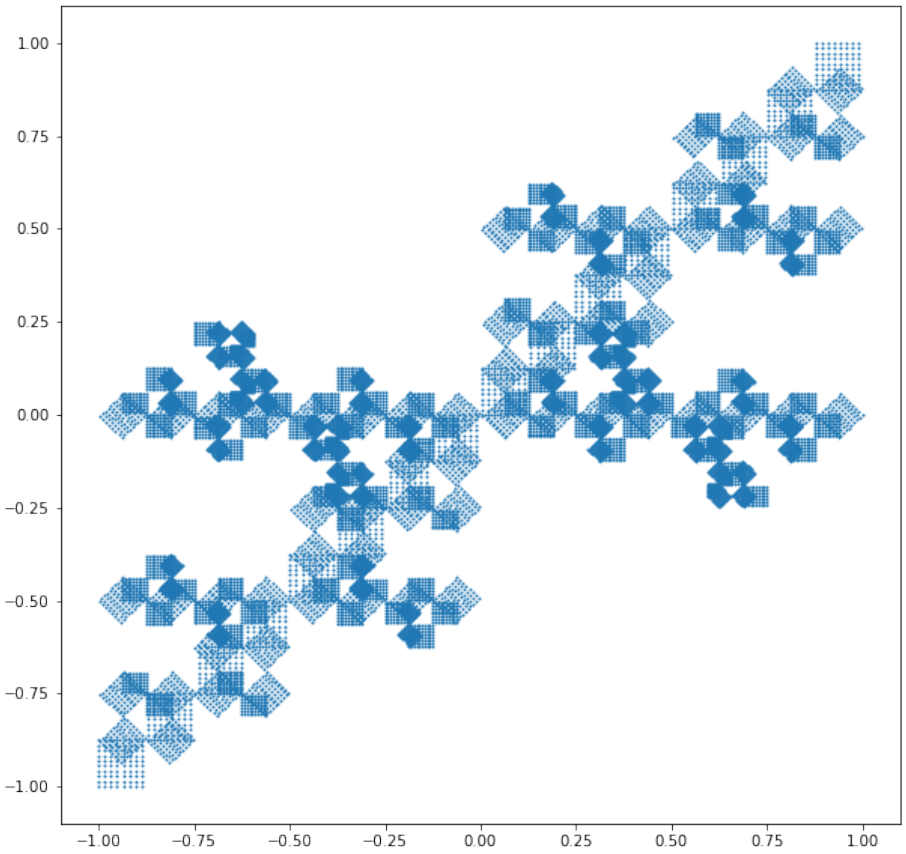


Now fractals

```
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)
```

```
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)))
shape = iterate(4)(frac)(square)
```

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



Now use a graphics library like a regular person.

```
import cairocffi as cairo
from PIL import Image
```

```
def draw_point(ctx, p, r=0.005):
    ctx.arc(*p, r, 0, 2*np.pi)
    ctx.fill()
    ctx.stroke()

def draw_line(ctx, line):
    p1, p2 = line
    ctx.move_to(*p1)
    ctx.line_to(*p2)
    ctx.stroke()
```

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

Draw a mandala-like fractal

```
# drawgeom, mapgeom, initgeom = draw_line, linegeom, [([-1, 0], [1, 0])]
drawgeom, mapgeom, initgeom = draw_point, pointgeom, [[0, 0]]

langle = np.pi / 8
n = 4
geoms = iterate(n)(frac(mapgeom)(*mcompose(
    [I, rot(np.pi / 2)],
    [I, resize(-1, 1)],
    [trans(1, 0)],
    [I, resize(-0.5, 1)],
    [I, resize(1, -1)],
    [compose(trans(0, np.tan(langle)), rot(-langle), scale(1 / (2*np.cos(langle))), trans(-1, 0))]
)))(initgeom)
```

```
width, height = 300, 500
surface = cairo.PDFSurface('mandala.pdf', width, height)
ctx = cairo.Context(surface)
ctx.translate(width / 2, height / 2)
side = 0.5 * min(width, height)
ctx.scale(side, side)
ctx.move_to(0, 0)

ctx.set_source_rgba(0, 0, 1, 0.25)
ctx.set_line_width(1e-4)
ctx.scale(0.5, 0.5)
for geom in geoms:
    drawgeom(ctx, geom)

ctx.set_source_rgba(1, 0, 0)
ctx.arc(0, 0, 0.01, 0, 2*np.pi)
ctx.fill()
ctx.stroke()

surface.finish()
```

Distort a region

```
def draw_poly(ctx, poly):
    ctx.move_to(*poly[0])
    for p in poly[1:]:
        ctx.line_to(*p)
    ctx.set_source_rgba(1, 0, 1, 0.5)
#     ctx.fill()
    ctx.fill_preserve()
    ctx.set_source_rgba(0, 1, 0, 0.5)
    ctx.stroke()

polygeom = lambda f, poly: [f(p) for p in poly]
```

```
drawgeom, mapgeom, initgeom = (
    draw_poly,
    polygeom,
#     [[circle([a, 1]) for a in np.linspace(0, 1, 4)]]
    [
        [[0,0], [1,0], [1/2,1]],
```

```
        [[1,1], [1,0], [1/2,1]],
        [[0,1], [1/2,1], [0,0]]
    ]
)

geoms = iterate(8)(frac(mapgeom)(*mcompose(
    [resize(1, 1/2), compose(scale(1/2), trans(0, 1)), compose(scale(1/2), trans(1, 1))]
)))(initgeom)
geoms = iterate(2)(frac(mapgeom)(*mcompose(
    [compose(scale(0.5), trans(1, 1), circle)]
)))(geoms)
# geoms = frac(mapgeom)(circle)(geoms)
# geoms = iterate(3)(frac(mapgeom)(*mcompose(
#     [I, rot(np.pi / 2)],
#     [I, resize(-1, 1)],
#     [trans(1, 0)],
#     [I, resize(-0.5, 1)],
#     [I, resize(1, -1)],
#     [compose(trans(0, np.tan(langle)), rot(-langle), scale(1 / (2*np.cos(langle))), trans(-1, 0))],
# )))(geoms)
```

```
width, height = 300, 500
surface = cairo.PDFSurface('distort.pdf', width, height)
ctx = cairo.Context(surface)
ctx.translate(width / 2, height / 2)
side = 0.5 * min(width, height)
ctx.scale(side, -side)

ctx.scale(0.5, 0.5)

# ctx.set_source_rgba(0, 0, 0, 0.02)
# draw_poly(ctx, [[0,0],[0,1],[1,1],[1,0]])
# ctx.set_source_rgba(0, 1, 0, 0.5)
# ctx.arc(1/2, 1/2, 0.01, 0, 2*np.pi)
# ctx.fill()

ctx.set_line_width(1e-4)
ctx.set_source_rgba(0, 0, 1, 0.25)
for geom in geoms:
    drawgeom(ctx, geom)

ctx.set_source_rgb(1, 0, 0)
ctx.arc(0, 0, 0.01, 0, 2*np.pi)
ctx.fill()

surface.finish()
```

A line fractal

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

```
            [rot(2*np.pi*i/m) for i in range(m)],
            [compose(scale(-0.5), trans(1, 0))]
        )],
        mcompose(
            [rot(2*np.pi*i/m) for i in range(m)],
            [compose(scale(-0.5), trans(1, 0))]
        )
    ))(initgeom)
```

```
width, height = 300, 500
surface = cairo.PDFSurface('linefractal.pdf', width, height)
ctx = cairo.Context(surface)
ctx.translate(width / 2, height / 2)
side = 0.5 * min(width, height)
ctx.scale(side, -side)

ctx.scale(0.75, 0.75)

# ctx.set_source_rgba(0, 0, 0, 0.02)
# draw_poly(ctx, [[0,0],[0,1],[1,1],[1,0]])
# ctx.set_source_rgba(0, 1, 0, 0.5)
# ctx.arc(1/2, 1/2, 0.01, 0, 2*np.pi)
# ctx.fill()

ctx.set_line_width(1e-4)
ctx.set_source_rgba(0, 0, 1, 0.75)
for geom in geoms:
    drawgeom(ctx, geom)

ctx.set_source_rgb(1, 0, 0)
ctx.arc(0, 0, 0.01, 0, 2*np.pi)
ctx.fill()

surface.finish()
```