Oppgave 3

- Problemstillingen
- Part A) The Triangle
  - Likesidet trekant med fordreiningsmatrise. Tilfeldig startpunkt. Generer en trekant med fraktaler. Farger basert på hvilken vei og på hvor langt ifra hjørner.
- Part B) Generalizing the Chaos Game
  - Generalisere: n-gon. Tilfeldig startpunkt. Med plot med gradientfarge. Lagre figuren. Tester.
- Part C) Barnsely Ferns
  - Generere en Fern. Affine funksjoner. Utvalg av affine transformasjoner. Plott fernen.
- Part D) Variations
  - Implementere en klasse som fordreier et mesh. Implementer fra en catalog. Bruk variasjon clasen på chaos_game og fern.

- Temaer
- Implementasjon av matematiske modeller.
- Sammenknytting av programmer

Svar:

- Part A) The Triangle
  - Lager trekanten med en rotasjonsmatrise.
  - Generer punkter. (x)
  - Genererer farger (col) og alternative farger.
- Part B) Generalizing the Chaos Game
  - Genererer n-gon
  - For x0 brukes samme metode.
  - Itererer.
- Part C) Barnsely Ferns
  - Ny iterate
  - Bruker affine transformation
- Part D) Variations
  - Metoder for hver variasjon.
  - Tester implementasjonen med å plotte ens effekt på et uniform scatterplot.
  - Plotter variasjon i bruk på fern, og chaos game.
  - Lager en Call som lager en kolleksjon av variasjoner som blir applikert delvis.

```python
def __call__(self, x, y):
    point = np.array((x, y))
    mat = np.array([[self.a, self.b], [self.c, self.d]], np.float)
    ef = np.array((self.e, self.f))
    return np.dot(mat, point) + ef
```

```python
def example_fern():
    parameters = (
        (0, 0, 0, 0.16, 0, 0),
        (0.85, 0.04, -0.04, 0.85, 0, 1.60),
        (0.20, -0.26, 0.23, 0.22, 0, 1.60),
        (-0.15, 0.28, 0.26, 0.24, 0, 0.44),
    )
    distribution = (0.01, 0.85, 0.07, 0.07)
    n = 50_000
    ex_fern = fern.Ferns(parameters, distribution, n)
    x = ex_fern.iterate()
    coords_varia = Variations(x[:,0], -x[:,1], colors="g")
    variations = ["linear", "handkerchief", "swirl", "disc"]
    plt.figure(10, figsize=(9, 9))
    for i in range(4):
        plt.subplot(221 + i)
        variation = (variations[i])
        coords_varia.collection[variation]()
        coords_varia.plot()
        plt.title(variation)
    plt.show()
```

```python
def __call__(self, coeffs):
    keys = list(coeffs)
    self.collection[keys[1]]()
    u1 = self.u.copy()
    v1 = self.v.copy()
    self.collection[keys[0]]()
    u2 = self.u.copy()
    v2 = self.v.copy()
    self._u = u1*coeffs[keys[1]] + u2*coeffs[keys[0]]
    self._v = v1*coeffs[keys[1]] + v2*coeffs[keys[0]]
```

```python
rotation = np.array(
          ((np.cos(theta), np.sin(theta)), (np.sin(-theta), np.cos(theta)))
           )
c2 = diff @ rotation
```

```python
x[0] = self.generate_point()
for i in range(1, n):
    x[i] = (x[i - 1] + self.c[k[i]]) / 2
```

```python
col = np.where(k == 0, col1, np.where(k == 1, col2, col3))
```

```python
col[0] = np.random.random(3)
for i in range(1, len(k)):
    col[i] = (col[i - 1] + r[k[i]]) / 2
```

```python
def _generate_ngon(self):
    n = self.n
    theta = np.linspace(0, 2 * np.pi, n + 1)
    c = np.zeros((n, 2))
    for i in range(n):
        c[i] = (np.sin(theta[i]), np.cos(theta[i]))
    self.c = c
```

```python
def iterate(self, steps, discard=5):
    r = self.r
    c = self.c
    k = np.random.randint(self.n, size=steps)
    x = np.zeros((steps, 2))
    x[0] = self._starting_point()
    for i in range(1, steps):
        x[i] = (r * x[i - 1]) + ((1 - r) * c[k[i]])
    self.x, self.k = x[discard:], k[discard:]
```

```python
col[0] = k[0]
for i in range(1, len(k)):
    col[i] = (col[i - 1] + k[i]) / 2
```

```python
def iterate(self):
    p = self.p
    d = self.d
    n = self.n
    eps = 1e-13
    assert sum(d) - 1 < eps, f"sum(d) must equal 1, not {sum(d)}"
    assert len(p) == len(d), f"p and d must be of equal length: p:{len(p)}, d:{len(d)}"
    f = []
    for i in p:
        f.append(AffineTransform(a=i[0], b=i[1], c=i[2], d=i[3], e=i[4], f=i[5]))
    x = np.zeros((n, 2))
    x[0] = np.array(self.x0)
    func_val = self.choose_func()
    for i in range(1, n):
        x[i] = f[func_val[i]](x[i - 1, 0], x[i - 1, 1])
    return x
```

Utfordringer

- Fårstå oppgaven.
  - Variasjoner er ikke så forklarende navn.
  - Forklart på en forkjsellig måte
- Presentasjonene
- Sammarbeid.

Testing, og sikring at programmet virker

- Tester om ChaosGame feiler på feil init verdier
- Tester for rett feilmelding om vi caller show før vi har lagd noe å vise.
- Tester for at alle punkt skal ha abs verdi større enn 1
- Tester mot negativ n