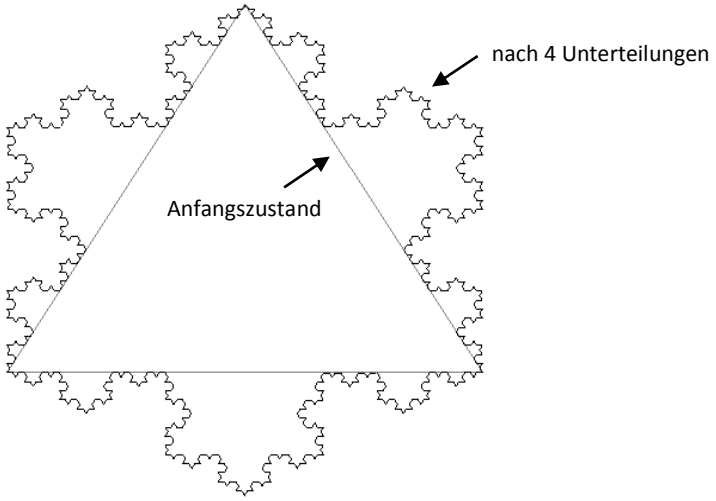
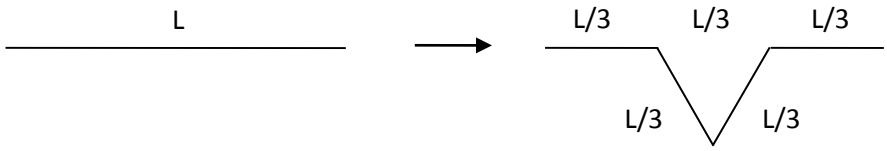
Aufgabe 1 – Koch-Schneeflocke 13 Punkte

Entwickeln Sie ein Programm zur Berechnung der Koch-Schneeflocke:



Die Koch-Schneeflocke ist rekursiv definiert: Beginne mit dem gleichseitigen Dreieck $(0,0), (1,0), \left(\frac{1}{2}, \frac{\sqrt{3}}{2}\right)$. Teile jede vorhandene Strecke der Länge L in vier neue Strecken der Länge L/3 gemäß untenstehender Skizze (die Spitze bildet wieder ein gleichseitiges Dreieck):



Die entstehenden Strecken werden analog weiter unterteilt. Theoretisch kann dies unendlich oft wiederholt werden, praktisch wird man nach etwa 4 bis 8 Unterteilungen aufhören.

a) Geben Sie die formale Unterteilungsregel an: Wie werden aus gegebenen Endpunkten p_1, p_2 einer Strecke die neuen Punkte berechnet? Hinweis: Die auf einem gegebenen Vektor $\vec{u} = \begin{pmatrix} s \\ t \end{pmatrix}$ senkrecht stehenden Vektoren sind $\vec{v} = \begin{pmatrix} t \\ -s \end{pmatrix}$ und $\vec{v}' = \begin{pmatrix} -t \\ s \end{pmatrix}$. *6 Punkte*

b) Implementieren Sie die Funktion `points=kochSnowflake(level)`. Die Anzahl der Unterteilungen wird durch den Parameter `level` bestimmt und die resultierende Punktliste als Python-Array (`numpy.ndarray`) zurückgegeben. Geben Sie die Implementation `snowflake.py` ab. Zeichnen Sie die Schneeflocke mit `matplotlib.pyplot` (http://matplotlib.org) wie folgt *7 Punkte*

```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 8))
plt.axis('equal')
plt.fill(x, y)
plt.show()
```

wobei x und y die Punktkoordinatenarrays darstellen. Geben Sie das Bild als Resultat ab.

Aufgabe 2 – Generisches Sortieren 8 Punkte

Die `sort()`-Funktion des Python-Arrays sortiert standardmäßig aufsteigend. Wenn eine andere Sortierung benötigt wird, kann man der Funktion einen unären Funktor (`myKey`) übergeben, der den Vergleichsschlüssel zurückgibt und/oder den Parameter `reverse` für eine absteigende Sortierung verwenden:

```
>>> array.sort(key = myKey, reverse = True)
```

Informieren Sie sich auf https://docs.python.org/3/howto/sorting.html über die Parameter der sort Funktion.

a) Implementieren bzw. benutzen Sie vorgegebene Funktoren für folgende Sortierprobleme und geben Sie Ihre Lösung zusammen mit geeigneten Tests im File `generic_sort.py` ab: *4 Punkte*

   I. Gegeben ist ein Array mit positiven und negativen Zahlen. Es soll nach aufsteigendem Absolutbetrag sortiert werden.

   II. Gegeben ist ein Array von Paaren (tuples), wobei das zweite Element in jedem Paar eine Zahl ist. Das Array soll nach absteigendem Absolutbetrag dieser Zahlen sortiert werden.

b) Manche Situationen können nicht mit einem unären Funktor ausgedrückt werden, z.B. wenn ein Array so sortiert werden muss, dass zuerst alle geraden Zahlen aufsteigend und danach alle ungeraden Zahlen absteigend erscheinen (z.B. [1, 4, 3, 5, 7, 2, 6] → [2, 4, 6, 7, 5, 3, 1]). In solchen Fällen kann man das Hilfsobjekt functools `.cmp_to_key` verwenden, *4 Punkte*

```
from functools import cmp_to_key
array.sort(key=cmp_to_key(myCompare))
```

das als Parameter einen binären Funktor `myCompare` einnimmt. Der Funktor `myCompare(x,y)` sollte eine negative Zahl zurückgeben, wenn x < y, eine positive, wenn x>y und 0, wenn x=y ist. Für Zahlen könnte ein solcher Funktor wie folgt implementiert werden:

```
def myCompare (x, y):
    return (x > y) - (x < y)
```

Implementieren Sie Ihre Lösung für die oben beschriebene Sortierung und geben Sie sie mit geeigneten Tests ebenfalls im File `generic_sort.py` ab.