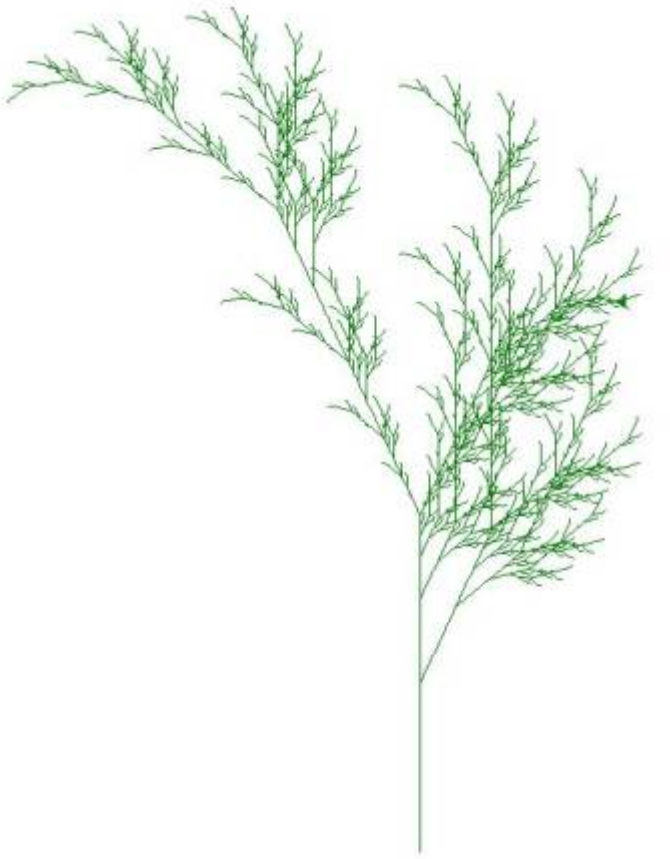
Fraktale w pythonie

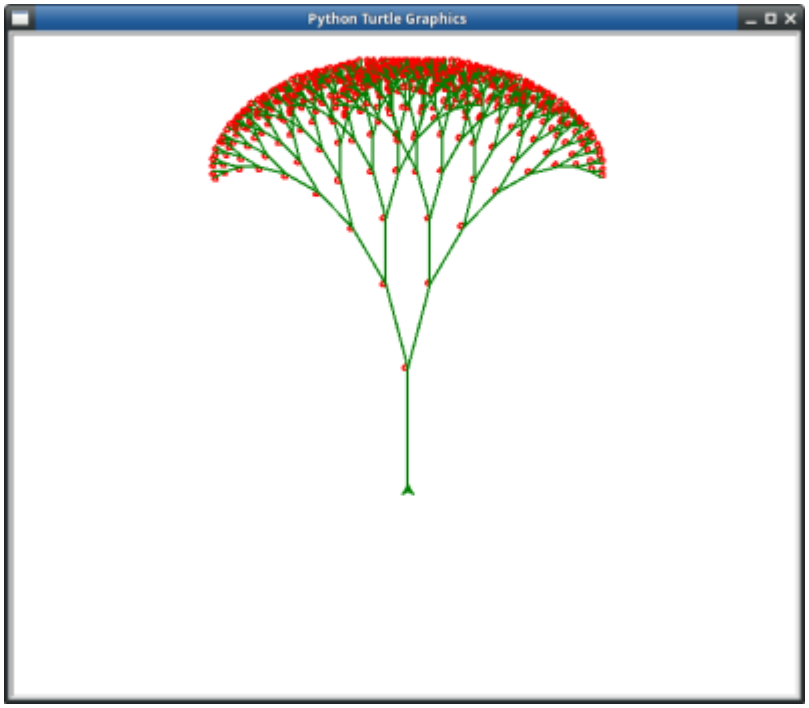
Wikipedia: L-systems



fractal_plant.py

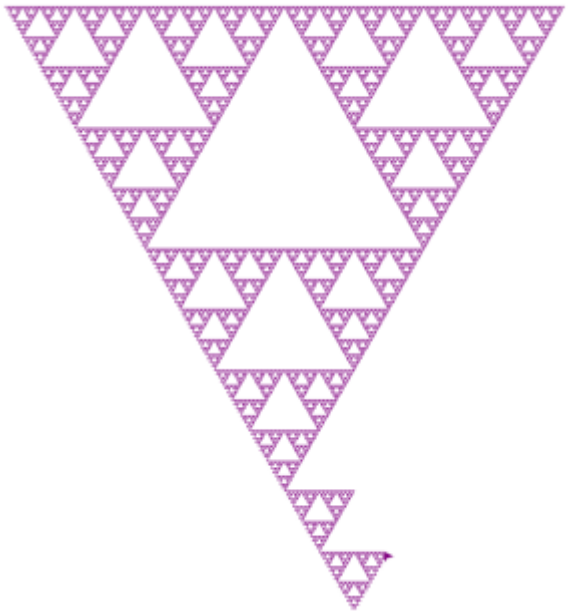
```
from turtle import *

tracer(0)
start="X"
dlugosc=4
kat=25

stos=[]
slZam={'X':'F+[[X]-X]-F[-FX]+X','F':'FF'}

iteracje=6
zolw='zolw'

def LSBuduj(st,ile,sl):
    nowy=""
    for litera in st:
        if litera in st:
            if litera in sl.keys():
```

```
                nowy+=sl[litera]
            else:
                nowy+=litera

    if ile>1:
        ile-=1
        return LSBuduj(nowy,ile,sl)
    else:
        return nowy
DoWykonania=LSBuduj(start,iteracje,slZam)
Polecenia={}
Polecenia['F']=[zolw+'.pd()',zolw+'.fd('+str(dlugosc)+')']
Polecenia['+']=[zolw+'.right('+str(kat)+')']
Polecenia['-']=[zolw+'.left('+str(kat)+')']
Polecenia['[']=['stos.append(('+zolw+'.xcor(),'+zolw+'.ycor(),'+zolw+'.
heading()))']
Polecenia[']']=[zolw+'.pu()',zolw+'.setx(stos[len(stos)-1][0])',
                            zolw+'.sety(stos[len(stos)-1][1])',
                            zolw+'.setheading(stos[len(stos)-1][2])',
                            'stos.pop()']

print(Polecenia)

zolw=Turtle()
zolw.pu()
zolw.goto(0,-300)
zolw.color('green')
zolw.pd()
zolw.setheading(90)
zolw.speed(0)
for litera in DoWykonania:
    if litera in Polecenia.keys():
        for rozkaz in Polecenia[litera]:
            eval(rozkaz)
update()
```



binary_tree.py

```
import turtle as t
t.speed(0)
t.pensize(2)
t.left(90)
t.backward(100)
t.color("green")

def draw(l):
    if(l<10):
        return
    else:
        t.forward(l)
        t.color("red")
        t.circle(2)
        t.color("green")
        t.left(45)
        draw(3*l/4)
        t.right(90)
        draw(3*l/4)
        t.left(45)
        t.backward(l)

draw(25)
t.exitonclick()
```



```
from turtle import *


start="F-G-G"
dlugosc=5
kat=120

slownik={}
slownik['G']="GG"
slownik['F']="F-G+F+G-F"

iteracje=10
zolw='zolw'

def LSBuduj(st,ile,sl):
    nowy=""
    for litera in st:
        if litera in slownik.keys():
            nowy+=sl[litera]
        else:
            nowy+=litera

    if ile>1:
        ile-=1
        return LSBuduj(nowy,ile,sl)
    else:
        return nowy

#print(len(LSBuduj(start,iteracje,slownik)))

DoWykonania=LSBuduj(start,iteracje,slownik)
```

```
Polecenia={}
Polecenia["G"]=[zolw+".fd("+str(dlugosc)+")"]
Polecenia["F"]=[zolw+".fd("+str(dlugosc)+")"]
Polecenia["+"]=[zolw+".left("+str(kat)+")"]
Polecenia["-"]=[zolw+".right("+str(kat)+")"]



zolw=Turtle()
zolw.pu()
zolw.goto(-300,200)
zolw.color('purple')
zolw.pd()
zolw.speed(0)
for litera in DoWykonania:
    if litera in Polecenia.keys():
        for rozkaz in Polecenia[litera]:
            eval(rozkaz)
```