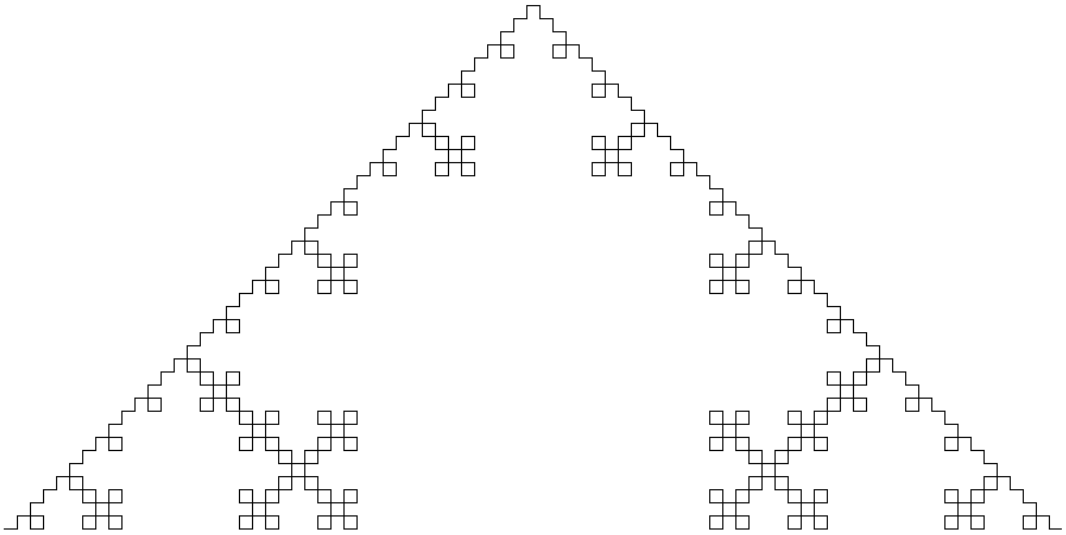
Implementacja systemu Lindenmayera, wykorzystanie go do rysowania fraktali w 2D

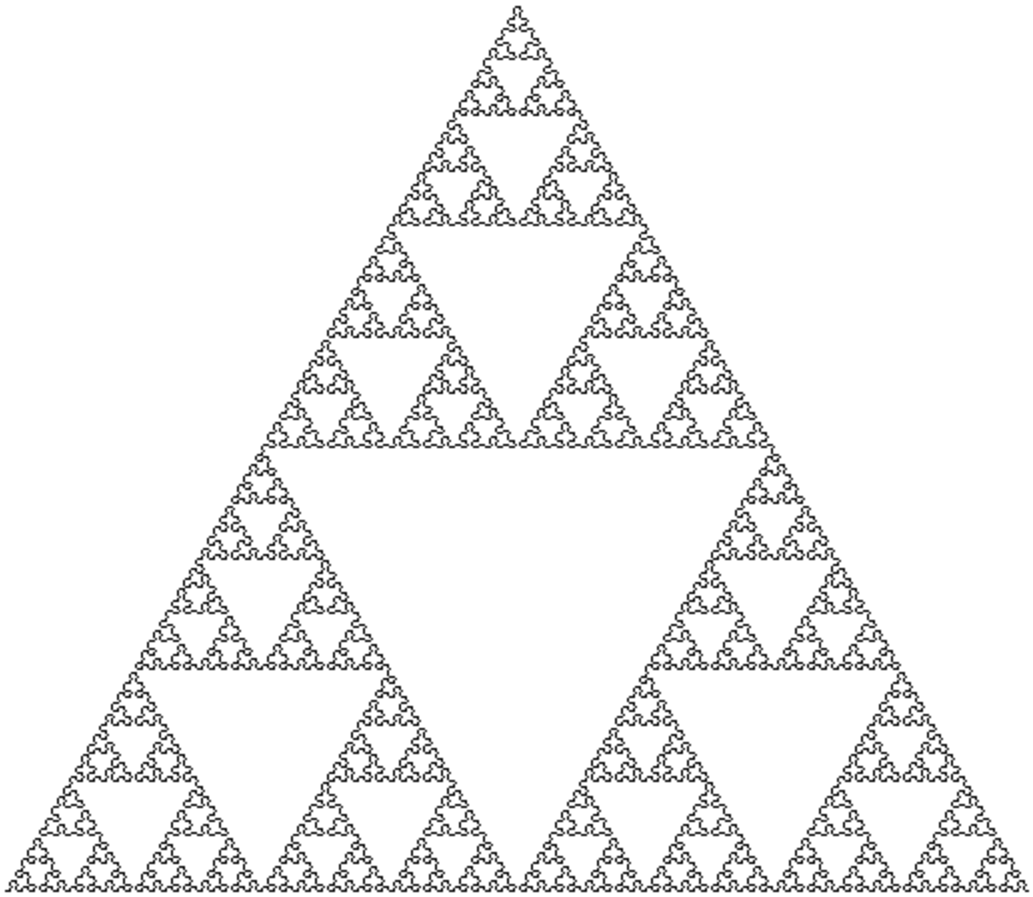
Projekt jest implementacją systemu Lindenmayera, nazywany również jako L-system. System taki może być przeznaczony do rysowania fraktali w 2D, na co kładzie nacisk ów implementacja.
Użytkownik może zdefiniować własny system lub wykorzystać któryś z już istniejących systemów, wbudowanych w moduł.

Opis

L-system jest zestawem reguł. Zawiera gramatykę składającą się z produkcji, które służą do wygenerowania słowa oraz zbiór reguł, na podstawie których rysowany jest fraktal, wykorzystując wygenerowane słowo.
Symbole w produkcjach nie mają podziału na terminale i nieterminale.
Implementacja zawiera dwie podstawowe operacje: rysowanie linii prostej oraz obrót.

Przykładowe fraktale:



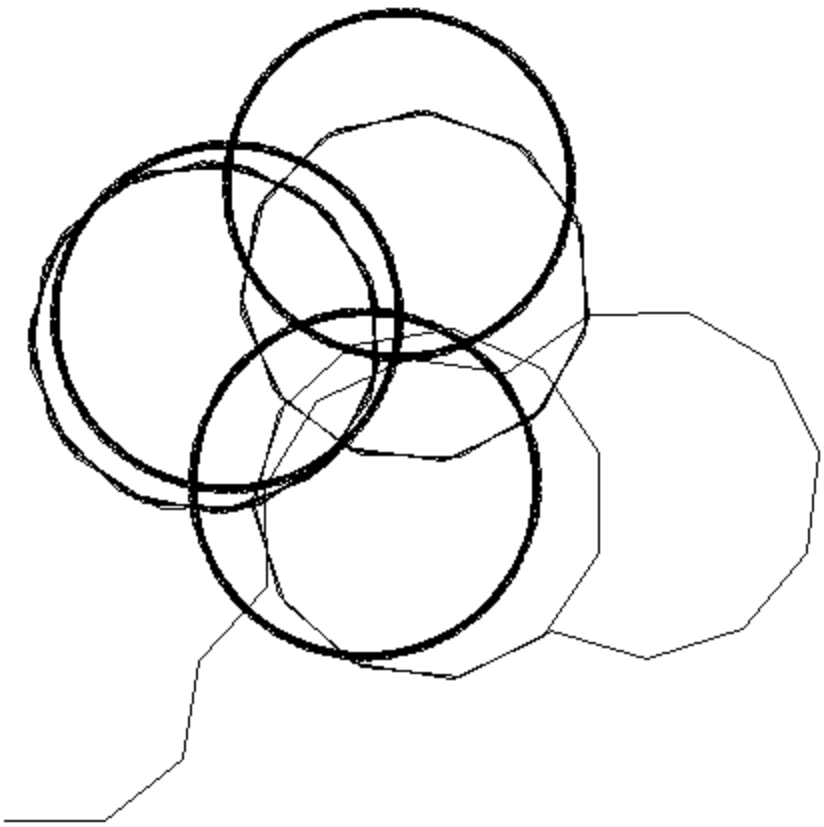


Przykładowe użytkowanie

 # Tworzę produkcje:

```
rule_1 = Rule('F', 'F+G')
rule_2 = Rule('G -> GGFG')
# Tworzę zbiór produkcji:
rules = Rules(rule_1, rule_2)
# Dodaję produkcję:
rules.add_rule(Rule('G', 'G-G'))
# Usuwam produkcję
rules.remove_rule(2)
# Tworzę L-system:
system = Lsystem(
        rules=rules,
        distance=50,
        angle=38,
        start='F',
        steps=10
)
# Definiuję własną regułę dla symbolu G
system.define_action(symbol='G', distance=50, angle=-5)
# Opcjonalnie zapisuję swój L-system, nadając nazwę
system.commit("Mój L-system!")
# Rysuję fraktal
system.draw()
```

Output



Przykładowe użytkowanie funkcji globalnych:

 # Inicjuję wbudowane L-systemy:

init_default_systems()

 # Wyświetlam numerowaną listę L-systemów:

print_systems()

 # Z wyświeltanej listy wybieram L-system:

system = get_system(4)

 # Rysuję go:

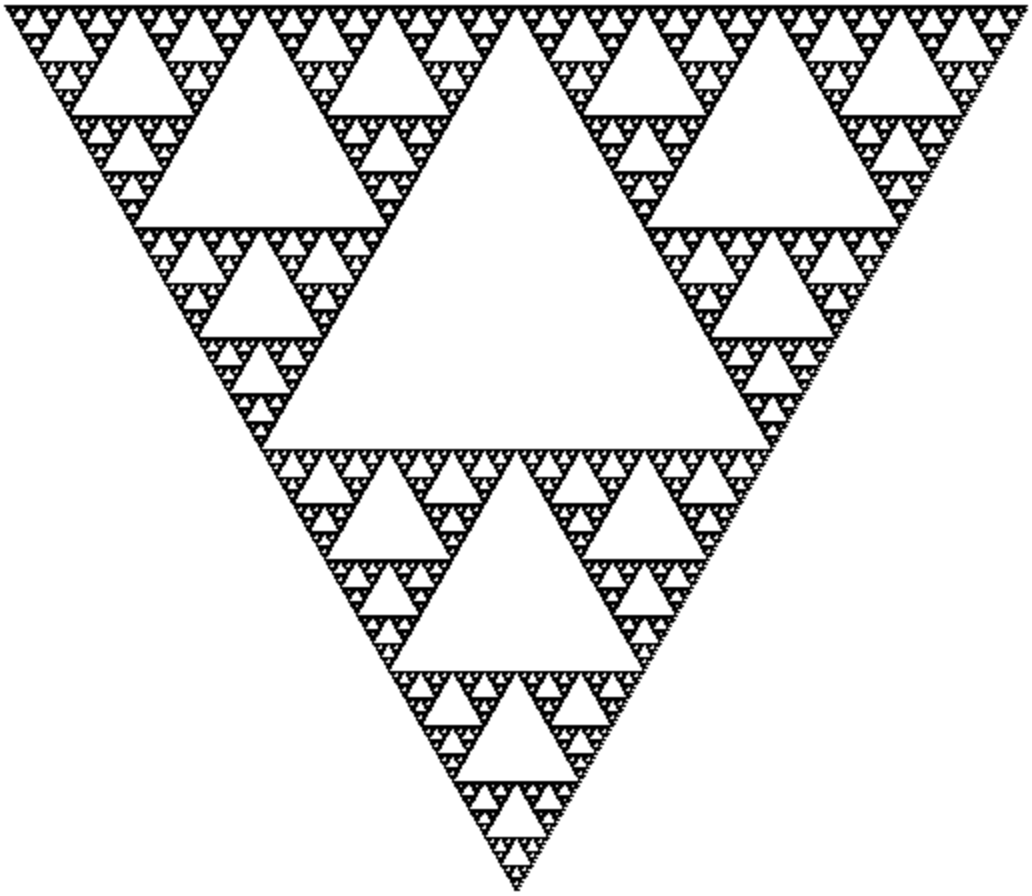
system.draw()

Output:

 # Wybieram inny i od razu rysuję:

get_system(2).draw()

Output:



Klasa Rule

Klasa Rule służy do zdefiniowania produkcji, poprzez stworzenie instacji. Można to zrobić na dwa sposoby:

1. Rule('[lewa strona produkcji] -> [prawa strona produkcji]')
2. Rule('[lewa strona produkcji]', '[prawa strona produkcji]')

Instancje klasy Rule zawsze są wyświetlane w postaci:
[lewa strona produkcji] -> [prawa strona produkcji]
np. *F -> F+G*

Klasa Rules

Klasa Rules służy do grupowania produkcji, uprzednio utworzonych za pomocą klasy Rule. Przy inicjalizacji instancji klasy Rules można podać dowolną liczbę argumentów, wszystkie natomaist muszą być instancjami klasy Rule.

Instancje klasy Rules zawsze są wyświetlane w postaci numerowanej listy wszystkich produkcji, które zawiera, np.
1. F -> F+G
2. G -> F-G-F

Metody dostępne dla użytkownika:

get_rules()
Zwraca listę z wszystkimi produkcjami.

add_rule(rule)
Dodaje nową produkcje, jako argument przyjmuje instancje klasy Rule.

get_rules_num()
Zwraca liczbę produkcji

remove_rule(index)
Usuwa, jeśli istnieje, produkcję ze wskazanym indeksem, jako argument przyjmuje liczbę całkowitą. Indeks danej produkcji można uzyskać wyświetlając instancję klasy Rules metodą print().

Klasa Lsystem

Główna klasa modułu. Służy do zdefiniowania systemu Lindenmayera.
Przy inicjalizacji instacji podawane są następujące argumenty nazwane:

- **start**
  Terminal startowy, rozpoczynający produkowanie słowa, na podstawie którego będzie rysowany fraktal. Terminal musi być ze zbioru terminali, przechowywanych jako lista *terminals* w zmiennych globalnych. Symbol startowy musi byc pojedynczym terminalem.
- **distance**
  Domyślna długość rysowanej prostej. Wszystkie terminale, dla których nie jest określona akcja, rysują prostą o długości równej tej wartości. Wartość

musi być liczbą.

- **angle**
  Kąt obrotu. Znaki "-" i "+" powodują obrót o wskazany kąt, kolejno zgodnie i przeciwnie do ruchu wskazówek zegara.
- **rules**
  Zbiór wszystkich produkcji. Argument musi być instancją klasy Rules.
- **steps**
  Liczba iteracji po wszystkich produkcjach, w celu utworzenia słowa służącego do rysowania fraktalu. Argument musi być liczbą całkowitą.

W przypadku nie wystąpienia któregoś z argumentów przy inicjalizacji instancji, brane są wartości domyślne ze zmiennych globalnych.

Metody dostępne dla użytkownika:

set_angle(angle)
Ustawia kąt obrotu (p. angle). Jako argument przyjmuje liczbę.

get_angle()
Zwraca kąt obrotu (p. angle).

set_start(start)
Ustawia symbol startowy. Jako argument przyjmuje terminal ze zbioru terminali, przechowywanych jako lista *terminals* w zmiennych globalnych.

get_start()
Zwraca symbol startowy.

set_distance(distance)
Ustawia domyślną długość rysowania prostej (p. distance). Jako argument przyjmuje liczbę.

get_distance()
Zwraca domyślną długość rysowania prostej (p. distance).

set_rules(rules)
Ustawia zbiór produkcji. Jako argument przyjmuje instancję klasy Rules.

get_rules()
Zwraca zbiór produkcji.

set_name(name)
Ustawia nazwę L-systemu, nie jest ona do niczego wykorzystywana.

get_name()
Zwraca nazwę L-systemu.

define_action(symbol, distance, angle)
Definiuje specjalną akcję dla danego symbolu. Należy podać argument nazwany *symbol*, do którego przypisana będzie akcja.
Podanie nazwanego argumentu *distance* sprawi, że akcją będzie rysowanie prostej o długości zgodnej z podaną wartścią. Musi to być liczba.
Podanie nazwanego argumentu *angle* sprawi, że akcją będzie obrót o podany kąt. Musi to być liczba.

draw()
Wykorzystując zdefiniowany system, w nowym oknie rysuje fraktal w 2D.

commit(name)
Zapisuje L-system globalnie. Wszystkie zapisane L-Systemy przechowywane są w postacji listy w zmiennej globalnej *systems.*

Funkcje globalne

init_default_systems()

W zmiennej globalnej *systems* umieszcza wcześniej przygotowane, przykładowe L-systemy, odpowiadające popularnym fraktalom.

print_systems()
Wypisuje numerowaną listę zapisanych systemów.

get_system(index)
Zwraca system o podanym indeksie. Jako argument przyjmuje liczbę całkowitą. Indeks można uzyskać za pomocą funkcji *print_systems().*

Zmienne globalne

special_characters
Lista ze specjalnymi znakami, są to "+" i "-"
terminals
Lista z dostępnymi terminalami, są to "F" i "G"
systems
Lista z zapisanymi systemami - instancjami klasy Lsystem
(p. commit(name))
DEFAULT_ANGLE
Domyślny kąt obrotu. Wynosi 45.
DEFAULT_DISTANCE
Domyślna długość rysowanej prostej. Wynosi 1.
DEFAULT_STEPS
Domyślna liczba kroków. Wynosi 5.

Wymagania

Biblioteka turtle.py
Python 3.0 lub nowszy