Dokumentacja programu rysującego wybrany przez użytkownika kształt w wybranym rozmiarze i kolorach pisany w języku Python

1. Wprowadzenie z opisem teoretycznym algorytmu

W projekcie użyty zostały użyte podstawowe metody modułu Turtle. Ze względu na częste problemy z kodowaniem polskich znaków implementacja nie zawiera polskich liter. Projekt składa się z dwóch plików. Pierwszy - ”fractals.py” to moduł zawierający trzy funkcję służące do rysowania kształtów: fraktal - krzywa Kocha, fraktal - drzewo oraz spirala Fibonacciego nazywana na potrzeby użytkownika muszlą, funkcję służącą stworzeniu żółwia i ustawieniu go w podstawowych parametrach potrzebnych przy rysowaniu kształtów oraz ostatniej - funkcji służącej rysowaniu kształtów - „draw”.

set_turtle()

Funkcja zmienia kolor tła na czarny, tworzy żółwia oraz ukrywa żółwia, tak aby był niewidoczny podczas rysowania. Zwraca żółwia, który będzie dalej używany do rysowania kształtów

tree(color1, color2, turtle, length)

Funkcja rekurencyjnie rysuje poszczególne gałęzie drzewa. Zmienia kolor w jakim będzie rysowany kształt na jeden z tych podanych przez użytkownika. Dopóki długość gałęzi nie równa się minimalnej ustawionej przez nas długości funkcja rysuje gałąź a następnie zmienia długość gałęzi o 5 i kąty położenia żółwia rysując 2 odchodzące od niej gałęzie.

snowflake(color1, color2, turtle, iterations, length=300)

Funkcja rysująca krzywą Kocha. W zależności od głębokości fraktala (rozmiar podany przez użytkownika) funkcja wywołuje się rekurencyjnie aż do uzyskania odpowiedniej długości, następnie rysuje odpowiednie proste pod odpowiednim kątem. Narysowanie docelowej śnieżynki wymaga połączenia ze sobą 3 takich prostych (funkcja draw())

fiboshell(color1, color2, turtle, n)

Funkcja n razy (rozmiar podany przez użytkownika) rysuje ćwiartki koła o zależnej od kolejnych liczb fibonacciego średnicy

draw(turtle, type, size, color1, color2)

Funkcja rysująca podane przez użytkownika kształty w podanych przez niego kolorach. W funkcji znajduje się słownika kolorów, który konwertuje numer koloru podany przez użytkownika na kolor obsługiwany przez moduł Turtle. Program wykonuje potrzebne operacje przygotowujące do rysowania w zależności od kształtu a następnie wywołuje odpowiednią funkcje.

Drugi plik - „main.py” zbiera od użytkownika dane dotyczące kształtu jaki chce on narysować, rozmiaru oraz kolorów oraz wywołanie funkcji draw, która przetwarza dane i wywołuje odpowiednią funkcję rysującą fraktal.

type_input()

Pozyskuje informację od użytkownika dotyczące typu rysowanego kształtu. Dba o to aby dane wprowadzone przez użytkownika były poprawne. Zwraca uzyskane poprawne dane.

length_input()

Pozyskuje informację od użytkownika dotyczące rozmiaru rysowanego kształtu. Dba o to aby dane wprowadzone przez użytkownika były poprawne. Zwraca uzyskane poprawne dane.

color1_input()

Pozyskuje informację od użytkownika dotyczące jednego z kolorów rysowanego kształtu. Dba o to aby dane wprowadzone przez użytkownika były poprawne. Zwraca uzyskane poprawne dane.

color2_input()

Pozyskuje informację od użytkownika dotyczące drugiego z kolorów rysowanego kształtu. Dba o to aby dane wprowadzone przez użytkownika były poprawne. Zwraca uzyskane poprawne dane.

2. Opis interfejsu

Program działa w trybie tekstowym. Pytania skonstruowane są tak, aby być czytelne na ekranie terminala 80x25. Dla łatwiejszego wprowadzania danych oraz uniknięcia pomyłek używane są pierwsze litery kształtów oraz liczby, które ułatwiają wprowadzenie koloru. Interfejs modułu turtle używa pióra i płótna, które są automatycznie tworzone przy wywołaniu którejkolwiek z funkcji modułu.

```
Jaki ksztalt chcesz narysowac?
Do wyboru mamy drzewo, sniezynke i muszle. Wybierz d, s lub m :)
s
Jakiej wielkosci w skali od 1 do 10 ma byc ksztalt?
6
Bedziemy rysowac dwukolorowy ksztalt, jaki bedzie pierwszy kolor?
1 - czerwony
2 - pomaranczowy
3 - zolty
4 - zielony
5 - niebieski
6 - granatowy
7 - fioletowy
3
Jaki bedzie drugi kolor?
1 - czerwony
2 - pomaranczowy
3 - zolty
4 - zielony
5 - niebieski
6 - granatowy
7 - fioletowy
6
```

W przypadku wprowadzenia przez użytkownika złych danych program wypisuje odpowiednie komunikaty pomocne we wprowadzeniu odpowiednich danych.

```
Do wyboru masz jedynie d dla drzewa, s dla sniezynki i m dla muszli.
g
Do wyboru masz jedynie d dla drzewa, s dla sniezynki i m dla muszli.
s
Jakiej wielkosci w skali od 1 do 10 ma byc ksztalt?
fgf
To nie jest liczba
Jakiej wielkosci w skali od 1 do 10 ma byc ksztalt?
345
Liczba nie jest z odpowiedniego przedzialu
Jakiej wielkosci w skali od 1 do 10 ma byc ksztalt?
```

Aby użytkownik mógł popatrzeć dłużej na narysowany kształt użyty jest na końcu input

```
Kliknij cokolwiek kiedy bedziesz chcial wylaczyc ekran
```

3. Uwagi na temat implementacji

PLIK MAIN.PY

W type_input() do czasu aż wprowadzone dane nie będą dokładnie takie jakie są wymagane program będzie pytał o literę odpowiadającą kształtowi, która jest zawarta w zbiorze set_of_types. Zwraca odpowiednią literę należącą do zbioru o ile taka została wprowadzona

```
def type_input():
    """Sluzy uzyskaniu od uzytkownika informacji dotyczacej rysowanego ksztaltu"""
    set_of_types = {'d', 's', 'm'} #zbior ksztaltow do wyboru
    f_type = input("Jaki ksztalt chcesz narysowac? \nDo wyboru mamy drzewo, "
                   "sniezynke i muszle. Wybierz d, s lub m :)\n")
    while f_type not in set_of_types:
        f_type = input("Do wyboru masz jedynie d dla drzewa, s dla sniezynki"
                   " i m dla muszli.\n")
    return f_type
```

W length_input() najpierw za pomocą try/except sprawdzamy czy wprowadzone dane są typu int, jeżeli nie na ekran wypisujemy informację o tym, że wprowadzone dane nie są liczbą. Jeżeli wprowadzone dane są liczbą, sprawdzamy jeszcze czy należą do przedziału 1-10 - range(1, 11)

```
def length_input():
    """Sluzy uzyskaniu od uzytkownika informacji dotyczacej wielkosci rysowanego ksztaltu"""
    while True:
        x = input("Jakiej wielkosci w skali od 1 do 10 ma byc ksztalt?\n")
        try:
            size = int(x)
        except ValueError:
            print("To nie jest liczba")
        else:
            if size not in range(1, 11):
                print("Liczba nie jest z odpowiedniego przedzialu")
            else:
                break
    return size
```

color1_input() oraz color2_input() najpierw za pomocą try/except sprawdzamy czy wprowadzone dane są typu int, jeżeli nie, na ekran wypisujemy informację o tym, że wprowadzone dane nie są liczbą. Jeżeli wprowadzone dane są liczbą, sprawdzamy jeszcze czy należą do przedziału 1-7 - range(1, 8)

```
def color1_input():
    """Sluzy uzyskaniu od uzytkownika informacji dotyczacej pierwszego koloru"""
    while True:
        y = input("Bedziemy rysowac dwukolorowy ksztalt, jaki bedzie pierwszy "
                  "kolor?\n1 - czerwony\n2 - pomaranczowy\n3 - zolty\n4 - "
                  "zielony\n5 - niebieski\n6 - granatowy\n7 - fioletowy\n")
        try:
            color1 = int(y)
        except ValueError:
            print("Powinienes wprowadzic liczbe od 1 do 7.")
        else:
            if color1 not in range(1, 8):
                print("Liczba nie jest z odpowiedniego przedzialu")
            else:
                break
    return color1


def color2_input():
    """Sluzy uzyskaniu od uzytkownika informacji dotyczacej drugiego koloru"""
    while True:
        z = input("Jaki bedzie drugi kolor?\n1 - czerwony\n2 - "
                  "pomaranczowy\n3 - zolty\n4 - zielony\n5 - niebieski\n6 "
                  "- granatowy\n7 - fioletowy\n")
        try:
            color2 = int(z)
        except ValueError:
            print("Powinienes wprowadzic liczbe od 1 do 7.")
        else:
            if color2 not in range(1, 8):
                print("Liczba nie jest z odpowiedniego przedzialu")
            else:
                break
    return color2
```

W pliku main.py znajduje się wywołanie funkcji draw() z moduły fractals, jako argumenty podane są wywołania opisanych wcześniej funkcji zbierających dane i zwracających je.

```
fractals.draw(fractals.set_turtle(), type_input(), length_input(), color1_input(), color2_input())
```

PLIK FRACTALS.PY

Moduł obsługujący rysowanie kształtów przez żółwia.

Importujemy moduł turtle oraz math potrzebny aby uzyskać liczbę pi używaną przy rysowaniu spirali fibonacciego

```
import math
import turtle
```

W funkcji zastosowana jest metoda bgcolor w celu ustawienia koloru tła na czarne, tak aby rysowany kształt był bardziej widoczny. Następnie tworzymy obiekt turtle1 i schowamy żółwia, tak aby był on niewidoczny dla użytkownika i żeby polepszyć widoczność kształtu.

```
def set_turtle():
    """Funkcja ustawiajaca poczatkowe dane zolwia"""
    turtle.bgcolor("black")
    turtle1 = turtle.Turtle()
    turtle1.hideturtle()
    return turtle1
```

W tree(color1, color2, turtle, length) jako argumenty podany jest stworzony wcześniej obiekt turtle, zoperacjonalizowany rozmiar kształtu oraz dwa kolory. Rysuje gałąź główną drzewa wraz z rozgałęzieniami. Rozgałęzienia rekurencyjnie stają się gałęziami głównymi aż do osiągnięcia przez nie długości minimalnej. Do funkcji kolory podawane są w kolejności co drugie, żeby kolory poszczególnych gałęzi się zmieniały.

```
def tree(color1, color2, turtle, length):
    """Funkcja rysujaca fraktal przypominajacy drzewo"""
    turtle.color(color1)
    angle = 30
    miniumum_length = 5
    if length > miniumum_length:
        turtle.forward(length)
        new_length = length - 5
        turtle.left(angle)
        tree(color1, color2, turtle, new_length)
        turtle.right(2 * angle)
        tree(color2, color1, turtle, new_length)
        turtle.left(angle)
        turtle.color(color1)
        turtle.backward(length)
```

Funkcja snowflake jako argumenty przyjmuje dwa kolory, obiekt turtle, ilość iteracji - od których zależy to jak głęboki będzie fraktal (ta zmienna zmienia się w zależności od tego co na poda użytkownik) oraz zmienną length która początkowo będzie wynosić 300, jeżeli nie zostanie podane inaczej. Length wynosi 300 dlatego, że ta wartość pozwala na odpowiednią widoczność żółwia. Im głębszy będzie fraktal, tym mniejsza podana dalej rekurencyjnie wartość length. Length zmniejsza się 3 razy przy każdym zmniejszeniu wartości iterations.

```
def snowflake(color1, color2, turtle, iterations, length=300):
    """Funkcja rysujaca fraktal przypominajacy sniezynke"""
    turtle.color(color1)
    if iterations == 0:
        turtle.forward(length)
    else:
        iterations = iterations - 1
        length = length / 3
        snowflake(color1, color2, turtle, iterations, length)
        turtle.left(60)
        snowflake(color2, color1, turtle, iterations, length)
        turtle.right(60 * 2)
        snowflake(color1, color2, turtle, iterations, length)
        turtle.left(60)
        snowflake(color2, color1, turtle, iterations, length)
```

Funkcja fiboshell jako argumenty przyjmuje dwa kolor, obiekt turtle oraz liczbę n - od której zależy wielkość spirali. Na początku deklarujemy wartości dwóch pierwszych liczb fibonacciego. Następnie iteracyjne tyle razy ile podamy w wartości n rysujemy ćwiartkę koła. W tym celu liczymy długość jednej czwartej obwodu koła o promieniu równym wartościom kolejnych liczb fibonacciego. Następnie dzielimy tą długość na 90 i rysujemy po kawałku, zmieniając kąt pióra o 1. Pod koniec każdej iteracji zmieniamy wartość liczby b, dodając do niej poprzednia liczbę fibonacciego.

```
def fiboshell(color1, color2, turtle, n):
    """Funkcja rysujaca ksztalt przypominajacy spirale """
    a = 0
    b = 1
    for i in range(n):
        if i % 2 == 0:
            turtle.color(color1)
        else:
            turtle.color(color2)
        curve = math.pi * b * 3 / 2
        curve = curve / 90
        for j in range(90):
            turtle.forward(curve)
            turtle.left(1)
        temp = a
        a = b
        b = temp + b
```

Funkcja draw korzysta z opisanych wcześniej funkcji rysujących poszczególne kształty. Ta funkcja służy zoperacjonalizowaniu zmiennych wprowadzony przez użytkownika, aby przekazać je do funkcji w takiej formie, która będzie odpowiednia. Zawiera słownik kolorów, który ułatwia konwertowanie numeru koloru na jego nazwę przyjmowaną przez metodę color() modułu turtle. Funkcja w zależności od rysowanego kształtu {d, s, m} wykonuje odpowiednie działania przygotowujące obiekt. W przypadku drzewa ‚d' jest to przeskalowanie zmiennej size i ustawienie pióra w pozycji „do góry". W przypadku śnieżynki ‚s' funkcja przeskalowuje zmienną size a następnie przesuwa pióro w celu rysowania śnieżynki na środku ekranu. W przypadku muszli ‚m' przeskalowuje rozmiar. Następnie wywoływana jest jedna z opisanych wyżej funkcji. Na końcu, czekamy na aż użytkownik wciśnie enter, aby mógł na narysowany kształt popatrzeć dłużej.

```
def draw(turtle, type, size, color1, color2):
    """Funkcja rysujaca ksztalty w zaleznosci od podanych danych"""
    # zbior kolorow do wyboru
    set_of_colors = {'1': 'brown1',
                     '2': 'chocolate1',
                     '3': 'DarkGoldenrod2',
                     '4': 'DarkOliveGreen3',
                     '5': 'CadetBlue',
                     '6': 'blue2',
                     '7': 'BlueViolet'
                     }
    turtle_color1 = set_of_colors[str(color1)]
    turtle_color2 = set_of_colors[str(color2)]
    if type == 'd':
        length = 20 + (size - 1) * 3
        turtle.setheading(90)
        tree(turtle_color1, turtle_color2, turtle, length)
    elif type == 's':
        size = round(1 + (size - 1) * 0.3)
        turtle.up()
        turtle.left(180)
        turtle.forward(300 / 2)
        turtle.right(180)
        turtle.down()
        for i in range(3):
            snowflake(turtle_color1, turtle_color2, turtle, size, 300)
            turtle.right(120)
    elif type == 'm':
        length = round(1 + (size - 1) * 1.2)
        fiboshell(turtle_color1, turtle_color2, turtle, length)
    input("Kliknij enter kiedy bedziesz chcial wylaczyc ekran")
```

4. Testy

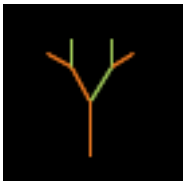
Ze względu na specyfikę projektu testy są trudne do przeprowadzenia. Przedstawię wyniki testów w zależności od wprowadzonych danych

```
1 - czerwony
2 - pomaranczowy
3 - zolty
4 - zielony
5 - niebieski
6 - granatowy
7 - fioletowy
```

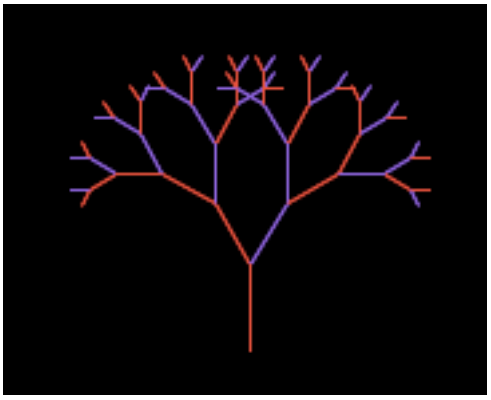
d - drzewo
s - śnieżynka
m - muszelka

rozmiary : 1-10

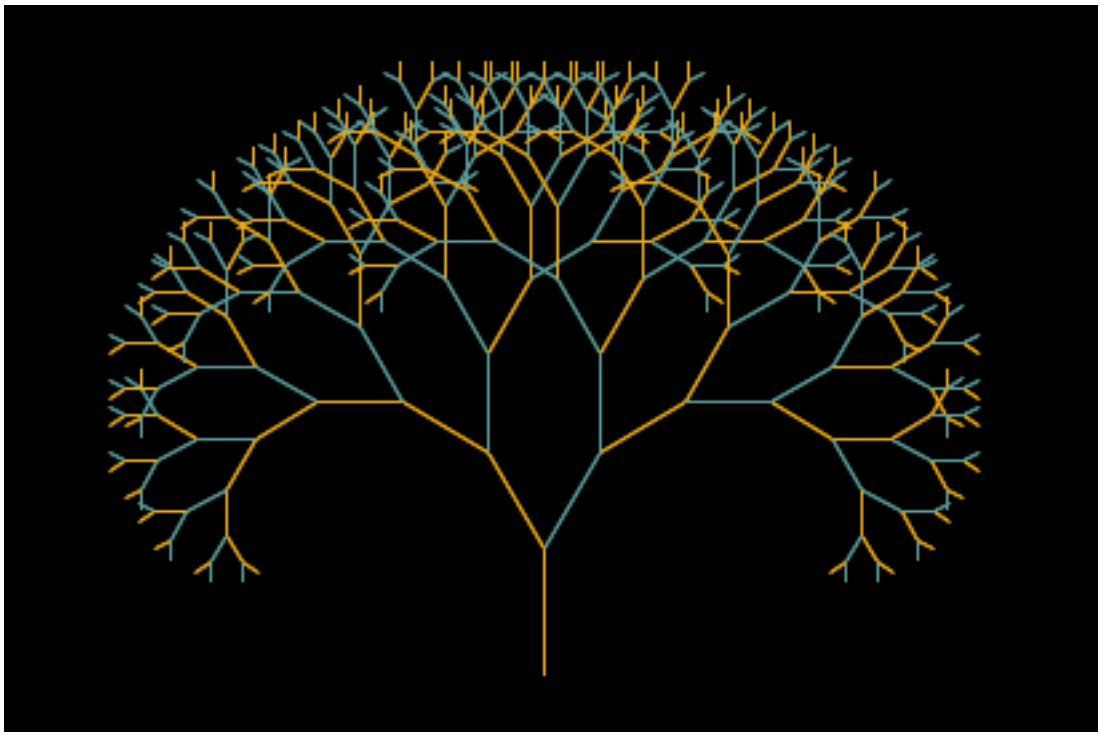
- Typ: „d” rozmiar: 1, kolory: 2 i 4



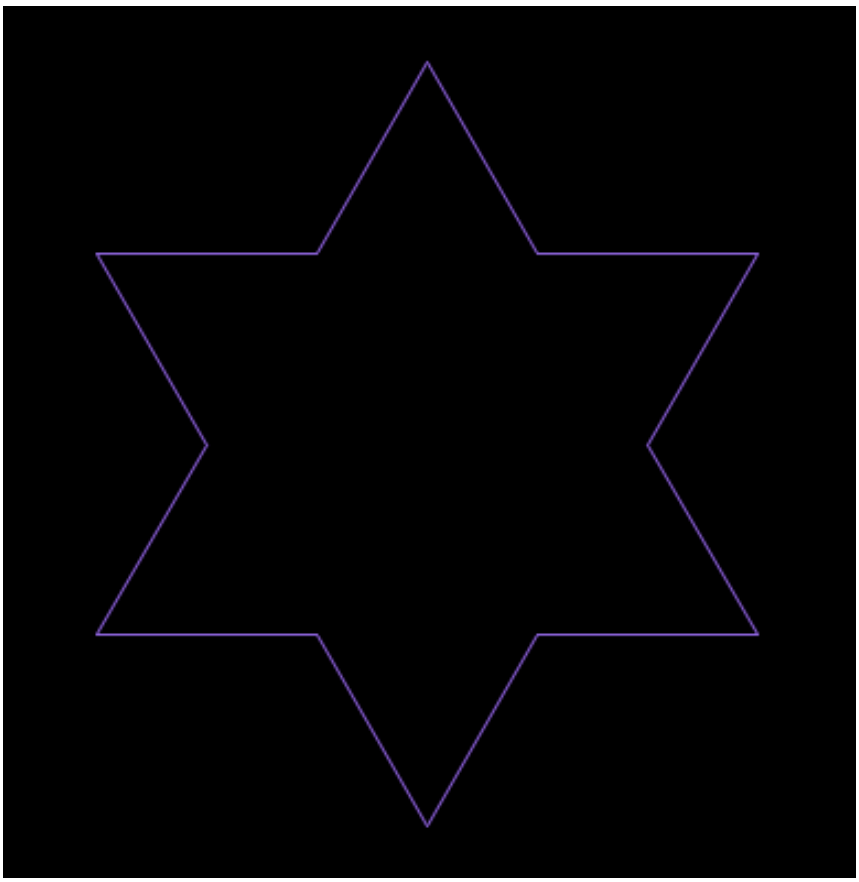
- Typ: „d” rozmiar: 5, kolory: 1 i 7



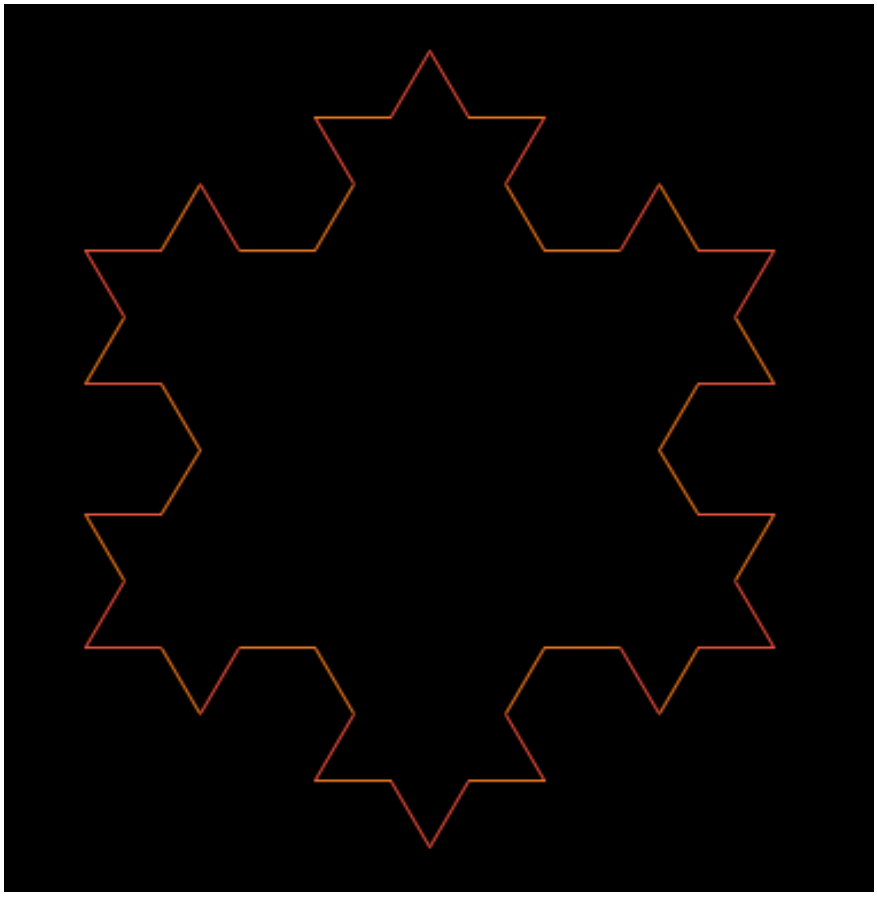
- Typ: „d” rozmiar: 10, kolory: 3 i 5



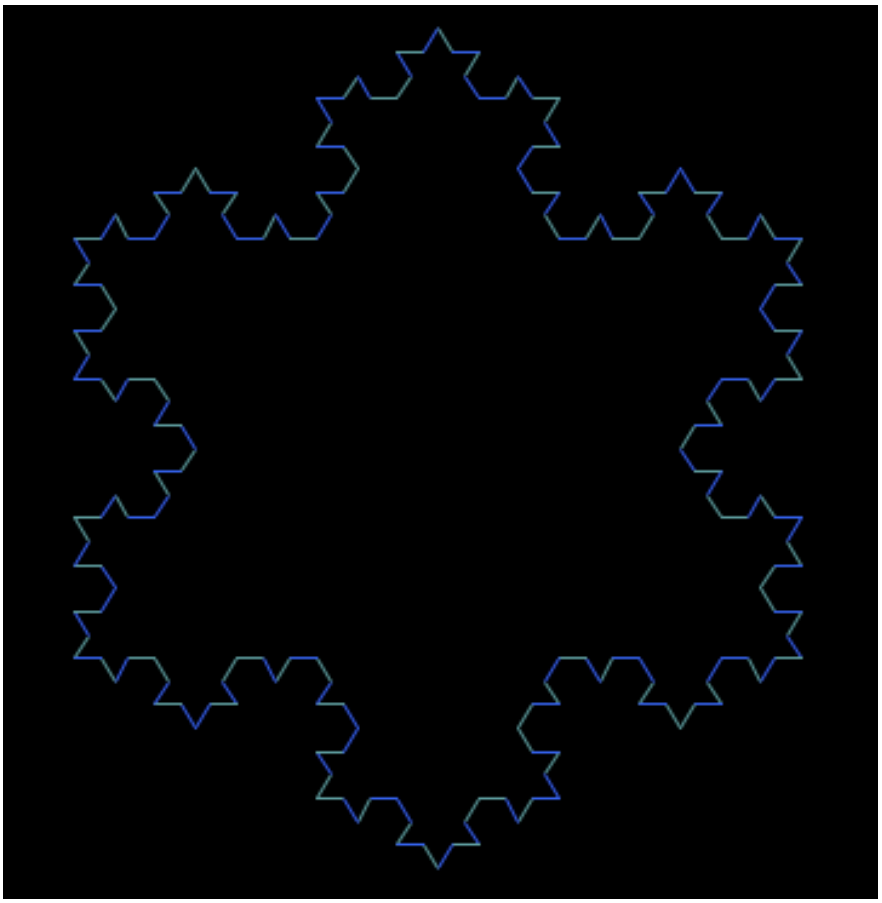
- Typ: „s” rozmiar: 1, kolory: 7 i 7



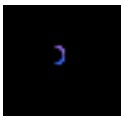
- Typ: „s” rozmiar: 4, kolory: 1 i 2



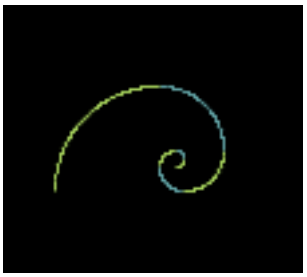
- Typ: „s” rozmiar: 7, kolory: 5 i 6



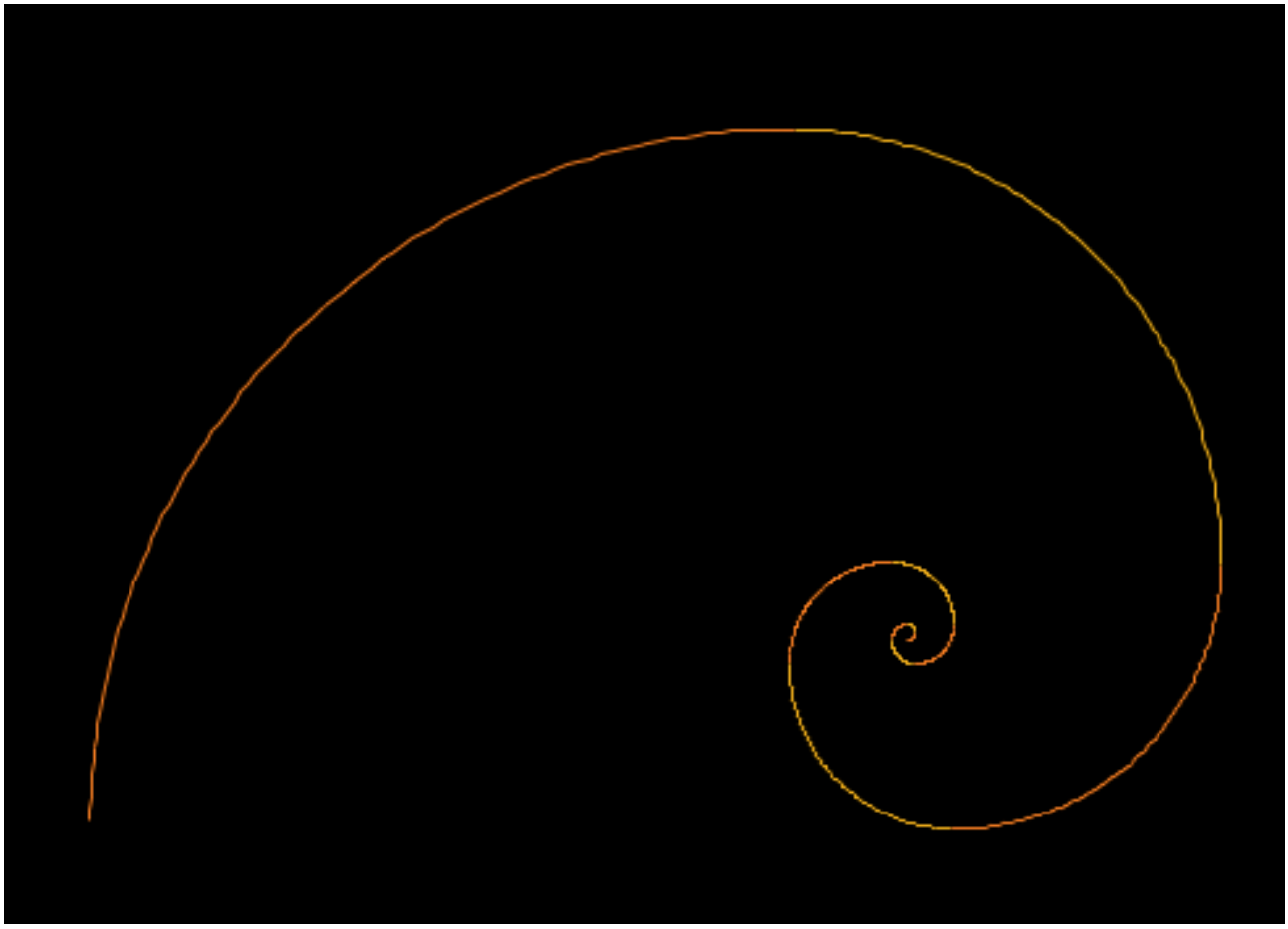
- Typ: „m” rozmiar: 2, kolory: 6 i 7



- Typ: „m” rozmiar: 6, kolory: 4 i 5



- Typ: „m” rozmiar: 9, kolory: 2 i 3



5. Literatura wykorzystana przy tworzeniu projektu

https://realpython.com/beginners-guide-python-turtle/
https://ufkapano.github.io/algorytmy/index.html
http://www.math.uni.wroc.pl/~elakalin/Prezentacja%20fraktale.pdf
https://home.agh.edu.pl/~zobmat/2020/III_kac_greg/index.html
https://www.geeksforgeeks.org/y-fractal-tree-in-python-using-turtle/
https://pl.wikipedia.org/wiki/Złota_spirala
https://stackoverflow.com/questions/6234798/turtle-graphics-how-do-i-control-when-the-window-closes
https://realpython.com/beginners-guide-python-turtle/