Laboratorium

Tematy różne: pomiar czasu, pliki, assert, fraktale, pygame

Zadanie 1

Przetestuj poniższy program do mierzenia czasu.

```
import time

start = time.time()

for i in range(10000):
    for j in range(10000):
        a = i - j

end = time.time()

print(end-start)
```

Wykonaj eksperymenty:

a) Ile czasu zajmie sortowanie 100 liczb metoda bąbelkową?
b) Ile czasu zajmie sortowanie 1000 liczb metoda bąbelkową?
c) Ile czasu zajmie sortowanie 10000 liczb metoda bąbelkową?

Wykorzystaj kod z poprzednich zajęć.

Zadanie 2

Komenda sleep robi pauzę w programie na podaną liczbę sekund. Przetestuj jak działa program:

```
import time

i = 0
while i < 30:
    print(i)
    i = i + 1
    time.sleep(1)
```

Następnie zmodyfikuj program tak, aby licznik w tym programie przyśpieszał. Na początku liczby pojawiają się wolno, ale każda kolejna pojawia się trochę szybciej niż poprzednia.

Zadanie 3

Przeczytaj o komendach do obsługi plików w Pythonie:

- https://www.w3schools.com/python/python_file_handling.asp
- https://www.w3schools.com/python/python_file_open.asp
- https://www.w3schools.com/python/python_file_write.asp
- https://www.w3schools.com/python/python_file_remove.asp
- https://www.geeksforgeeks.org/reading-writing-text-files-python/
- https://www.geeksforgeeks.org/read-a-file-line-by-line-in-python/

Następnie stwórz program, który:

a) Stworzy nowy plik liczby.txt i wpisze do niego liczby od 1 do 100, każda w osobnej linijce.
b) Zamknie ten plik.
c) Otworzy ten plik do odczytu i wypisze jego zawartość.
d) Zamknie ten plik.
e) Ponownie otworzy ten plik do edycji i do wszystkich liczb parzystych doda 10.
f) Zawartość zaktualizowanego pliku wyświetlimy i zamkniemy.

Zadanie 4

Do odczytu plików z uporządkowanym tekstem (np. tabelki, słowniki) można użyć paczki `pandas` (poczytaj o tej paczce)

a) Ściągnij plik miasta.csv ze strony i umieść go w folderze z plikami pythonowymi.
b) Użyj paczki `pandas`, żeby pobrać tabelę (data frame) do pythona z tego pliku.
c) Dodaj za pomocą odpowiedniej instrukcji wiersz do tabeli z ludnością w 2010 roku: `2010,460,555,405`
d) Zmodyfikowaną tabelę zapisz do pliku miasta.csv.

Zadanie 5

Instrukcja assert w Pythonie jest używana jako pomoc w debugowaniu, która sprawdza określony warunek. Pomaga zweryfikować, czy podczas wykonywania programu spełnione są określone warunki. Jeśli warunek jest prawdziwy, program kontynuuje wykonywanie. Jeśli warunek jest fałszywy, zgłaszany jest wyjątek AssertionError, opcjonalnie z niestandardowym komunikatem błędu.
assert jest zazwyczaj używany podczas programowania, aby wcześnie wychwycić błędy i upewnić się, że założenia w kodzie są prawidłowe.

Załóżmy, że mamy funkcję obliczającą średnią ocen, ale chcemy upewnić się, że lista ocen nie jest pusta. W takim przypadku można użyć assert, aby wykryć potencjalne błędy w użyciu funkcji:

```
def oblicz_srednia(oceny):
    assert len(oceny) > 0, "Lista ocen nie może być pusta"
    return sum(oceny) / len(oceny)

# Przykład poprawny
print(oblicz_srednia([4, 5, 3]))  # Wynik: 4.0

# Przykład z błędem
print(oblicz_srednia([]))  # Zgłasza AssertionError: Lista ocen nie może być pusta
```

Sprawdź czy powyższa funkcja działa i czy w przypadku podania pustej listy komenda assert zgłosi błąd.

Zadanie 6

Napisz funkcję oblicz_pole_prostokata(a, b), która oblicza pole prostokąta.

- Użyj assert, aby upewnić się, że oba wymiary prostokąta są liczbami dodatnimi.
- Jeśli warunek nie jest spełniony, funkcja powinna zgłaszać AssertionError z odpowiednim komunikatem.
-

```
print(oblicz_pole_prostokata(5, 10))  # Wynik: 50
print(oblicz_pole_prostokata(-5, 10))  # AssertionError: Wymiary muszą być dodatnie
```

Zadanie 7

Napisz funkcję sprawdz_uzytkownika(uzytkownik), która sprawdza, czy słownik użytkownika zawiera klucze imie i wiek.

- Użyj assert, aby upewnić się, że oba klucze istnieją w słowniku.
- Jeśli warunek nie jest spełniony, funkcja powinna zgłaszać AssertionError z komunikatem Niepoprawny obiekt użytkownika.

```
uzytkownik1 = {"imie": "Anna", "wiek": 25}
uzytkownik2 = {"imie": "Piotr"}

print(sprawdz_uzytkownika(uzytkownik1))  # Brak błędu
print(sprawdz_uzytkownika(uzytkownik2))  # AssertionError: Niepoprawny obiekt
użytkownika
```

Zadanie 8

Fraktale to geometryczne kształty, które cechuje samopodobieństwo – ich struktura powtarza się w różnych skalach, niezależnie od poziomu powiększenia. Są one tworzone na podstawie prostych reguł matematycznych, które często wykorzystują rekurencję, przez co idealnie nadają się do wizualizacji w programowaniu. Fraktale znajdują zastosowanie w nauce, sztuce i technologii, np. w modelowaniu struktur naturalnych, takich jak drzewa, chmury czy linie brzegowe.

Poniżej wklejono trzy programy z fraktali. Skopiuj je, uruchom i przeanalizuj.

Drzewo fraktalne:

```
import turtle

def draw_tree(branch_length, t):
    if branch_length > 5:  # Base case
        # Draw main branch
        t.forward(branch_length)

        # Recursive call for the right subtree
        t.right(20)
        draw_tree(branch_length - 15, t)

        # Return to the main branch and adjust direction
        t.left(40)
```

```
        draw_tree(branch_length - 15, t)

        # Reset orientation
        t.right(20)
        t.backward(branch_length)

# Setup Turtle
screen = turtle.Screen()
screen.bgcolor("white")
t = turtle.Turtle()
t.color("green")
t.speed("fastest")
t.left(90)  # Start pointing upwards
t.penup()
t.goto(0, -300)  # Start near the bottom of the screen
t.pendown()

# Draw fractal tree
draw_tree(100, t)

screen.exitonclick()
```

Śnieżynka Kocha:

```
import turtle

def koch_curve(t, length, level):
    if level == 0:  # Base case
        t.forward(length)
    else:
        length /= 3.0
        koch_curve(t, length, level - 1)
        t.left(60)
        koch_curve(t, length, level - 1)
        t.right(120)
        koch_curve(t, length, level - 1)
        t.left(60)
        koch_curve(t, length, level - 1)

# Setup Turtle
screen = turtle.Screen()
screen.bgcolor("white")
t = turtle.Turtle()
t.speed("fastest")

# Draw Koch snowflake
for i in range(3):  # 3 sides of the snowflake
    koch_curve(t, 300, 3)  # Level 3 recursion
    t.right(120)
```

Trójkąt Sierpińskiego:

```
import turtle

def sierpinski(t, length, level):
    if level == 0:
        for _ in range(3):  # Base case: Draw a triangle
            t.forward(length)
            t.left(120)
    else:
        length /= 2
        sierpinski(t, length, level - 1)  # Top triangle
        t.forward(length)
        sierpinski(t, length, level - 1)  # Bottom-right triangle
        t.backward(length)
        t.left(60)
        t.forward(length)
        t.right(60)
        sierpinski(t, length, level - 1)  # Bottom-left triangle
        t.left(60)
        t.backward(length)
        t.right(60)

# Setup Turtle
screen = turtle.Screen()
screen.bgcolor("white")
t = turtle.Turtle()
t.speed("fastest")

# Draw Sierpiński Triangle
sierpinski(t, 300, 4)  # Level 4 recursion

screen.exitonclick()
```

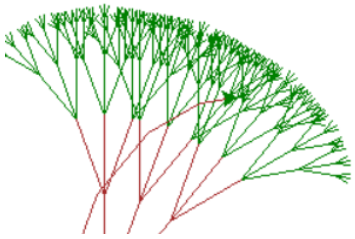
Zadanie 9

Wprowadź modyfikacje dla drzewa rekurencyjnego.

a) Dodaj kolorwanie gałęzi kolorem zielonym (liście) lub brązowym (większe gałęzie)

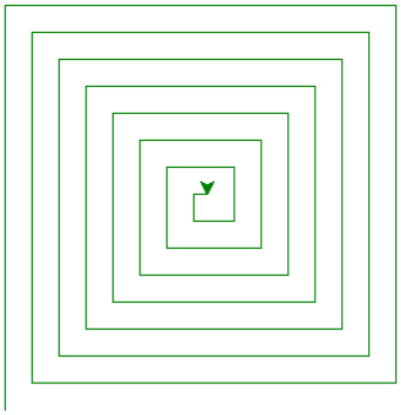
```
    if branch_length > 30:
        t.color("brown")
    else:
        t.color("green")
```

b) Zrób rozgałęzienia na trzy gałęzie a nie na dwie. Trzecia gałąź powinna wyrastać między lewą i prawą (po środku)



Zadanie 10

Napisz własny program na rysowanie fraktala w postaci spirali. Ścianki się skracają i zakręcają o 90 stopni. Możesz posłużyć się kodem drzewa jako podstawą.
Przykładowy rysunek:



Zadanie 11

Przetestuj możliwości paczki `pygame`, która służy do robienia gier z interfejsem graficznym. Przerób jeden z samouczków dostępnych w internecie i pochwal się wynikiem (prymitywną grą).

Należy wybrać samouczek po samodzielnych poszukwianiach, ale jeśli ktoś nie ma pomysłu to niżej kilka przykładowych.

Przykładowe samouczki:

- https://realpython.com/pygame-a-primer/

- (trzyczęściowy)
  https://coderslegacy.com/python/python-pygame-tutorial/
  https://coderslegacy.com/python/pygame-tutorial-part-2/
  https://coderslegacy.com/python/pygame-tutorial-part-3/

- (długi, z filmami)
  https://pythonprogramming.net/pygame-python-3-part-1-intro/
  https://pythonprogramming.net/displaying-images-pygame/?completed=/pygame-python-3-part-1-intro/
  https://pythonprogramming.net/pygame-tutorial-moving-images-key-input/?completed=/displaying-images-pygame/
  https://pythonprogramming.net/adding-boundaries-pygame-video-game/?completed=/pygame-tutorial-moving-images-key-input/
  https://pythonprogramming.net/displaying-text-pygame-screen/?completed=/adding-boundaries-pygame-video-game/
  https://pythonprogramming.net/drawing-objects-pygame-tutorial/?completed=/displaying-text-pygame-screen/

https://pythonprogramming.net/pygame-crashing-objects/?completed=/drawing-objects-pygame-tutorial/
https://pythonprogramming.net/adding-score-pygame-video-game/?completed=/pygame-crashing-objects/