Python - Dywan Sierpińskiego

Uwagi ogólne

Uwaga! Wcięcia w kodzie, mają dla Pythona znaczenie (są konieczne w odpowiednich miejscach)! Python do pobrania (dla Windows): *https://www.python.org/downloads/windows/*

Uruchamianie skryptu:

C:\> **python naszSkrypt.py**

Uwaga: Nigdy nie wolno kopiować kodu z PDF-a, gdyż zawiera on niewidoczne znaki końca linii i tzw. twarde odstępy. Kod należy przepisać ze zrozumieniem.

Nasz skrypt

Dywan Sierpińskiego jest funkcją rekurencyjną (czyli wywołującą samą siebie), lub jak w poniższym przykładzie, definiującą na nowo samą siebie. Program tworzy najpierw kwadrat składający się z 9 elementów, przy następnej rekurencji tworzy większy kwadrat składający się z 9 większych elementów, z których każdy zawiera 9 mniejszych (sumarycznie mamy przy drugim przejściu pętli już 9x9=81 elementów) i tak w nieskończoność. Maksymalny poziom zagłębienia w poniższym algorytmie wynosi 4, gdyż ograniczeniem w tym przypadku jest wielkość i rozdzielczość monitora (poziom 4 dobrze prezentuje się jedynie na monitorach 4K).

Kod skryptu - wersja podstawowa

```
def dywan_Sierpinskiego(poziomZaglebienia):
    przedza = ["# "]
    for kazdyPrzebieg in range(poziomZaglebienia):
        przedza = [kwadrat + kwadrat + kwadrat for kwadrat in przedza] + \
        [kwadrat + kwadrat.replace("#"," ") + kwadrat for kwadrat in przedza] + \
        [kwadrat + kwadrat + kwadrat for kwadrat in przedza]
    return "\n".join(przedza)

print(dywan_Sierpinskiego(3))
input('Naciśnij ENTER, aby zakończyć...')
```

Kod skryptu - wersja rozbudowana

```
#!/usr/bin/env python
# Powyższa linia tylko dla użytkowników systemu Linux.

# ================= DYWAN SIERPIŃSKIEGO ================

from os import system # Wymagane do kolorowana składni w systemie Windows 10/11:
system("")
# Uwaga: Składnia nie będzie kolorowana, gdy uruchomimy kod w IDLE Shell (to nie jest
# prawdziwy terminal) oraz w Windows 7/8.
```

```
def dywan_Sierpinskiego(poziomZaglebienia):
    # Definiujemy zmienną jako stringa z kwadratem i spacją. Przędza reprezentuje w tym momencie poziom zagłębienia numer 1:
    przedza = ["■ "]
    # Ile razy zostanie wykonany obiekt składający się z dziewięciu kwadratów (środkowy kwadrat zostaje zastąpiony dwoma spacjami):
    for kazdyPrzebieg in range(poziomZaglebienia):
        # Serce algorytmu. Z każdym przebiegiem pętli, zmienna "przedza" oznacza coś innego, reprezentuje coraz większą strukturę. Wyrażenie "kwadrat" też oznacza coś innego.
        przedza = [kwadrat + kwadrat  + kwadrat for kwadrat in przedza] + \
        [kwadrat + kwadrat.replace("■"," ") + kwadrat for kwadrat in przedza] + \
        [kwadrat + kwadrat + kwadrat for kwadrat in przedza]
        # Uwaga: po pierwszym przebiegu, przędza nie jest już pojedynczym kwadracikiem, ale kwadratem składającym się z 9 (8) kwadracików. Po drugim przebiegu, przędza jest już kwadratem składającym się z 81 kwadracików (wliczając w to czarne pola). Po każdym przebiegu, kwadrat oznacza coraz większy element, itd.


    # Dopiero w tym momencie, całość jest scalana w jedną formę i wyświetlana na ekranie:
    return "\n".join(przedza)


try:
    print("\n\n\033[1;48;5;22m ===== DYWAN SIERPIŃSKIEGO ===== \033[0m\n")
    poziomZaglebienia = int(input("Podaj poziom zagłębienia rekurencji (od 1 do 4): "))
    if poziomZaglebienia < 1 or poziomZaglebienia > 4:
        print("\n\n\033[1;37;41m BŁĄD: \033[0m")
        print("\033[1;31;40mWprowadzona wartość musi być z zakresu od 1 do 4.\033[0m\n")
    else:
        print("\033[0;33;40m\n\n", dywan_Sierpinskiego(poziomZaglebienia), "\033[0m", sep="")

except ValueError:
    print("\n\n\033[1;37;41m BŁĄD: \033[0m")
    print("\033[1;31;40m1. Wprowadzona wartość albo nie jest dodatnią liczbą całkowitą...\n2. Albo w ogóle nie wprowadzono żadnych wartości.\n\nSpróbuj jeszcze raz.\033[0m\n")

finally:
    input('\n\n\033[1;30;40mNaciśnij ENTER, aby zakończyć...\033[0m\n')
```

Ostatnia aktualizacja: 6 grudnia 2022.