Wstęp do programowania 2017
Pracownia 5

Uwaga: Na tej liście też będą wprawki i na kolejnych też. Podczas tych zajęć można oddawać zadania z listy trzeciej za 0.5 i czwartej za 1. Bonus (0.5) dla tej listy jest zdobycie ponad jednego punktu. Terminem zerowym dla tej listy są zajęcia w tygodniu 6-10 listopada.

Zadanie 1.(1pkt) Będziemy rozważać problem Collatza. Funkcja F zdefiniowana jest następująco:

```
def F(n):
   if n % 2 == 0:
      return n / 2
   else:
      return 3 * n + 1
```

Będziemy rozważać ciąg: $a, F(a), FF(a), FFF(a), FFFF(a) \ldots$, gdzie a jest całkowite dodatnie. Istnieje hipoteza, że ciąg ten zawsze kończy się powtarzającą w nieskończoność sekwencją $1, 4, 2, 1, 4, 2, 1, \ldots$. Dla danej liczby jej energią nazwiemy pozycję, na której w tym ciągu po raz pierwszy pojawia się jedynka. Przykładowo, energią liczby 7 jest 16, ponieważ ciąg Collatza rozpoczynający się od wartości 7 wygląda tak (z jedynką na 16-tej pozycji, licząc od zera):

```
7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

Napisz funkcję, która wylicza energię liczby. Napisz funkcję analizaCollatza(a,b), która oblicza wszystkie energie liczb od a do b oraz wypisuje:

- Średnią energię
- Medianę energii
- Maksymalną i minimalną energię

Uwaga: Zadanie będzie miało dalszy ciąg. **Uwaga 2:** Oczywiście możesz założyć, że hipoteza Collatza jest prawdziwa i zawsze kiedyś dojdziemy do jedynki.

Zadanie 2.(1pkt) Wybierz jeszcze jeden rysunek: albo czarno-biały (ale bez literki Ł, albo kolorowy), którego jeszcze nie robiłeś i napisz korzystając z modułu turtle program, który ten rysunek wykonuje.

Zadanie 3.(0.5pkt) Napisz funkcję `only_one(L)`, która dla listy `L` przegląda jej elementy i kopiuje do wyniku, pomijając te, które już wcześniej w tym wyniku się pojawiły. Przykładowo `only_one([1,2,3,1,2,3,8,2,2,2,9,9,4])` powinno zwrócić `[1,2,3,8,9,4]`.

Implementacja powinna być prosta, powinieneś w niej skorzystać z warunku `element in Lista`.

Zadanie 4.(1pkt)$\star$ Implementacja z powyższego zadania działa niestety bardzo wolno dla większych list. Napisz tę funkcję efektywniej, korzystając z sortowania oraz tego, że listę par Python sortuje patrząc w pierwszej kolejności na pierwszy element, potem na drugi. **Nie wolno** korzystać ze zbiorów, o których powiemy na kolejnym wykładzie.

Wskazówka 1: zobacz, co dzieje się, gdy napiszemy:

```
L1 = sorted([5,4,3,1,2,6])
L2 = [5,3,1,4,6,4,9]
L2.sort()
```

Wskazówka 2: (www.rot13.com): Zbman mnzvravp yvfgr an yvfgr cne mnjvrenwnplpu jnegbfp j beltvanyarw yvfpvr v cbmlpwr, anfgrcavr gr yvfgr cbfbegbjnp, hfhanp qhcyvxngl jnegbfpv v cbfbegbjnp wrfmpmr enm mr jmtyrqh an cbmlpwr.

Zadanie 5.(0.5pkt) Wracamy do gry ze smokiem, ale tym razem sprawdzimy ją porządnie. Punkty za to zadanie otrzymasz jedynie wówczas, gdy wyniki Twojego programu będą bardzo bliskie 7% w wersji z ostrą nierównością i 50% w wersji ze słabą nierównością[1]. Prawdopodobieństwa szacuj wykonując 100 tysięcy gier. Jeżeli nie oddałeś do tej pory zadania ze smokiem, możesz, jeśli chcesz dostać za nie dodatkowe 0.5 punktu w ramach tego zadania.

[1] Te liczby otrzymane zostały przez wykładowcę. Jeżeli program wykładowcy (a dokładniej dwa programy) jest błędny, to wykładowca będzie się wstydził, a na SKOS pojawi się odpowiednia informacja