Wstęp do programowania
Pracownia 5

Uwaga: Na tej liście też będą wprawki. 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.

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.(1pkt) $\star$ Na wykładzie pisaliśmy funkcję `only_one`, która przeglądała elementy listy i pozostawiała te, które do tej pory nie pojawiły się w wyniku. Działała ona 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 4.(1pkt) Powinieneś zmodyfikować „minijęzyk" do rysowania murków (program z Wykładu 4), który obsługuje polecenia `f,r,l`. W nowej wersji funkcja `murek` powinna mieć dwa argumenty: program (napis) oraz kolory (lista kilku kolorów biblioteki `turtle`). Twój program powinien obsługiwać następujące nowe komendy:

- `u` – odpowiednik `penup()`, czyli zaprzestania rysowania

- `d` – odpowiednik `pendown()`, czyli rozpoczęcia rysowania
- `0,1,2,3,...` – od teraz rysujemy w kolorze z listy `kolory` o jednocyfrowym numerze podanym w tym programie.
- `b` – ruch o 1 kwadracik do tyłu

Przykładowy program, który powinien narysować dwukolorowy krzyżyk:

```
4 * ('0' + 5 * 'f' + '1' + 5 * 'f' + 'u' + 10 * 'b' + 'r' + 'd')
```

Zaprezentuj działanie programu pisząc funkcję, rysującą 4-kolorowy kwadrat o boku `N` kwadracików, lub inny ciekawy, parametryzowany rysunek.

Zadanie 5.(1pkt)$\star$ Napisz inny interpreter do języka z poprzedniego zadania. Ten nowy interpreter powinien zmieniać kolory w sposób płynny. Załóżmy, że `x` oznacza polecenie rysujące kwadrat, a `y` inne polecenie (ale nie zmianę koloru). Wówczas program `0xxxyyxyxx1` powinien wyrysować 6 kwadratów których kolory zmieniają się płynnie między `kolory[0]` a `kolory[1]`. Zaprezentuj działania nowej wersji interpretera `murek`, za pomocą jakiegoś ciekawego rysunku (innego niż w poprzednim zadaniu)