**Generatory i Listy Składane:**

**Zad 1.**

Napisz funkcję generatora, która generować będzie kilka dowolnych wartości. Pobierz te wartości przy użyciu globalnej metody next() oraz metody generatora \_\_next\_\_(). Rzuć wyjątkiem wewnątrz generatora i zbadaj stack trace.

**Zad 2.**

Stwórz generator, który generował będzie kolejne liczby pierwsze.

**Zad 3.**

Jaki efekt da poniższy zapis?

examp = (i for i in range(10))

print(examp)

Jak należy zmienić kod, aby móc odczytywać kolejne liczby całkowite?

**Zad 3.**

Stwórz generator zwracający kolejne wartości ciągu Fibonacciego. Do obliczania kolejnych wyrazów, wykorzystaj poniższy zapis:

a = 1

b = 2

a, b = b, a + b

**Zad 4.**

Mając tak utworzoną listę liczb:

numbers = [1, -10, 2, 5, 10, -5, -20, 0, -30]

Wykorzystując, list comprehension, utwórz nową o nazwie filtered\_numbers, w której znajdą się tylko liczby niedodatnie z numbers.

**Zad 5.**

Bazując na następującym tekście: “*The quick brown fox jumps over the lazy dog is an English-language pangram—a sentence that contains all of the letters of the English alphabet*”, wydziel go na listę przechowującą długości kolejnych wyrazów z pominięciem słowa “the”, np.

length\_of\_words = [5, 5, 3, 5, ...], co odpowiada kolejno długościom wyrazów: *quick, brown, fox, jumps*.

**Zad. 6**

Mając do dyspozycji poniższą listę trójwymiarową:

three\_d = [

[[1, 2, 3, 4], [0, -1, -2, -3, -4, -5, -6]],

[[1, 10, 15, 12, 20, 20, 20], [-15, -13, 14, 20, -1]]

]

przefiltruj ją tak, by znalazły się tylko te najbardziej wewnętrzne listy, których ilość elementów przekracza 4.

Wynikiem powinna być lista:

[[0, -1, -2, -3, -4, -5, -6], [1, 10, 15, 1, 20, 20, 20], [-15, -13, 14, 20, -1]]