

# **Algorytmika**

Zadania zaliczeniowe

**Metody Wytwarzania Oprogramowania**  
**Łukasz Kaczmarek**

# Zadanie 1

## układ N równań liniowych z N niewiadomymi

### Kod:

```
import numpy as np
```

```
def create_file():
    file = open("matrix.txt", "w")
    matrix = [3,
               [1, 2, 3, 7],
               [-1, 2, 4, 6],
               [2, 1, 1, 13]]

    for line_number in range(len(matrix)):
        if line_number == 0:
            file.write(str(matrix[0]) + "\n")
        else:
            for number_index in range(len(matrix[line_number])):
                file.write(str(matrix[line_number][number_index]))
                if number_index != (len(matrix[line_number]) - 1):
                    file.write(" ")
            if line_number != (len(matrix) - 1):
                file.write("\n")
    file.close()

def read_file():
    matrix_txt = open("matrix.txt", "r").read()
    first_line = True
    matrix_A = []
    matrix_B = []

    for line in matrix_txt.split("\n"):
        if first_line:
            number_of_factors = int(line)
            first_line = False
        else:
            matrix_line = []
            numbers = line.split(" ")
            for number_index in range(len(numbers)):
                if number_index < number_of_factors:
                    matrix_line.append(float(numbers[number_index]))
            matrix_A.append(matrix_line)
            matrix_B.append([float(numbers[number_of_factors])])
    return np.array(matrix_A), np.array(matrix_B)

def solve():
    matrix_A, matrix_B = read_file()
    matrixes_Wi = []

    for i in range(matrix_A.shape[1]):
        matrix_Wi = matrix_A.copy()
        matrix_Wi[:, i] = matrix_B[:, 0]
        matrixes_Wi.append(matrix_Wi)

    Wis = []
    all_Wi_is_zero = True;
    for matrix_Wi in matrixes_Wi:
        Wi = np.linalg.det(matrix_Wi)
        Wis.append(Wi)
        if Wi != 0:
            all_Wi_is_zero = False
```

```

W = np.linalg.det(matrix_A)

if W == 0 and all_Wi_is_zero:
    print("Układ rownan ma nieskonczenie wiele rozwiazan")
elif W == 0:
    print("Układ rownan sprzecznych")
else:
    for i in range(matrix_B.shape[0]):
        print("x{}={}".format(i + 1, Wis[i] / W))

if __name__ == "__main__":
    # create_file()
    solve()

```

## Opis:

Do rozwiązywania zadanie wykorzystano twierdzenie Cramera:

Jeżeli wyznacznik macierzy współczynników układu jest różny od zera ( $\det A \neq 0$ ), to układ równań liniowych ma dokładnie jedno rozwiązanie dane wzorami:

$$x_i = \frac{W_i}{W},$$

gdzie

- $W = \det A$  - jest to wyznacznik macierzy współczynników układu.
- $W_i$  - jest to wyznacznik z macierzy, która powstaje z macierzy  $A$ , przez zastąpienie kolumny współczynników niewiadomej  $x_i$  przez kolumnę wyrazów wolnych.

Program składa się z trzech funkcji:

*create\_file()* - zapisuje w roboczym katalogu plik matrix.txt zawierający wartość N (ilość niewiadomych i równań w układzie) w pierwszej linii oraz współczynniki równań i wyraz wolny w kolejnych liniach. Funkcja stworzona do testowania programu i nie musi być wywoływana jeśli w folderze roboczym znajduje się już plik matrix.txt z danymi w formacie zgodnym z opisem zadania.

*read\_file()* - odczytuje z pliku matrix.txt w folderze roboczym współczynniki i wyrazy wolne poszczególnych równań układu oraz zwraca je jako dwie tablice numpy.array. Pierwsza zawiera macierz współczynników równań układu (macierz A - zgodnie z powyższym opisem twierdzenia Cramera) a druga macierz wyrazów wolnych.

*solve()* - zasadnicza funkcja rozwiązująca układ równań. W pierwszej kolejności wywołuje funkcję *read\_file()* i zapamiętuje jej wynik w zmiennych *matrix\_A* (macierz współczynników układu) oraz *matrix\_B* (macierz wyrazów wolnych). Lista *matrixes\_Wi* zawiera macierze  $W_i$  (zgodnie z opisem twierdzenia Cramera) - w pętli kolejne kolumny macierzy współczynników układu zastępowane są przez kolumnę wyrazów wolnych. Kolejna pętla oblicza wyznaczniki poszczególnych macierzy  $W_i$  i przechowuje je w liście *Wis*. Jeśli przynajmniej jeden wyznacznik macierzy  $W_i$  jest różny od zera to ustawiana jest wartość zmiennej *all\_Wi\_is\_zero* = True. Wyznacznik macierzy A zapisywany jest w zmiennej *W*. Następnie program sprawdza warunki układu równań nieoznaczonych (wyznaczniki wszystkich macierzy  $W_i$  i A są równe zero) oraz układu równań sprzecznych (wyznacznik macierzy A równy zero i przynajmniej jeden wyznacznik macierzy  $W_i$  różny od zera) i jeśli zachodzi któryś z tych przypadków to wypisywany jest odpowiedni komunikat. Jeśli układ jest oznaczony to obliczane i wypisywane są wartości rozwiązań przez obliczenie kolejnych ilorazów  $W_i/W$ .

## Przykładowe dane i wyniki:

Dla danych podanych w treści zadania tj.:

3

1 2 3 7

-1 2 4 6

2 1 1 13

Program wyświetla wynik:

```
/Users/lukaszkaezmarek/PycharmProjects/algorvymika/venv/bin/python /Users/lukaszkaezmarek/PycharmProject
```

```
x1=17.9999999999999968
```

```
x2=-57.999999999999997
```

```
x3=34.999999999999997
```

```
Process finished with exit code 0
```

Widoczny jest błąd związany z reprezentacją liczb zmiennoprzecinkowych.

## Zadanie 2

### Liczby Armstronga

#### Kod:

```
import numpy as np

n = 7
powers = np.zeros((10, n), dtype=int)

for x in range(10):
    for y in range(n):
        powers[x][y] = x ** (y + 1)

for liczba in range(10, 10 ** n, 10):
    temp = 0
    length = len(str(liczba))
    liczba_temp = liczba

    for i in range(length - 1, 0, -1):
        digit = int(liczba_temp / 10 ** i)
        liczba_temp -= digit * 10 ** i
        temp += powers[digit][length - 1]
        if (temp > liczba + 9):
            break

    if (temp < liczba + 9):
        for i in range(10):
            if (temp + powers[i][length-1]) == (liczba + i):
                print(liczba + i)
```

#### Opis:

Program wyszukuje liczby Armstronga od 10 (liczby mniejsze pominięto jako rozwiązania trywialne) do  $10^{n+9}$  (n może być zmieniane przez modyfikację 3 linii programu). Algorytm „brut force” sprawdzałby po kolei każdą z liczb N-cyfrowych w zadanym przedziale obliczając sumę cyfr podniesionych do potęgi N i porównując ją z tą liczbą. Program taki choć poprawny jest dosyć wolny dla liczb wielocyfrowych. W zaproponowanym rozwiązaniu zastosowałem 3 usprawnienia mające na celu przyspieszenie jego działania:

1. W celu uniknięcia wielokrotnego obliczania potęg liczb od 0 do 9 na początku programu tworzona jest tablica `numpy.array`s o nazwie `powers`, która zawiera potęgi od 1 do n (kolumny) liczb od 0 do 9 (wiersze tablicy). Czyli np. wynik 9 do potęgi 5 jest szybko dostępny jako `powers[9][5-1]` (przyp: 5-1 ze względu na indeksowanie od 0)
2. Zastosowano „przytrzymywanie wyniku tymczasowego”. To znaczy program zapamiętuje sumę potęg cyfr kolejnych liczb z krokiem 10 w zmiennej `temp` a następnie w pętli dodaje do wartości `temp` kolejne liczby z przedziału 0-9 podniesione do potęgi N. Powoduje to, że suma potęg wszystkich cyfr jest obliczana jedynie raz na 10 kolejnych liczb, co skutkuje przyspieszeniem w stosunku do algorytmu „brut force”, który obliczałby sumę potęg wszystkich cyfr dla każdej kolejnej liczby. Koncepcję tą można by rozwinąć stosując

wielostopniowe przytrzymywanie sum potęg dla różnych poziomów (setek, tysięcy itd - aż do 10 do potęgi N). Jednak nie spowoduje to już istotnego przyspieszenia algorytmu.

3. Sprawdzanie danej liczby jest przerywane jeśli suma potęg kolejnych liczb przekracza sprawdzaną liczbę. Dalsze obliczenia nie mają sensu bo wynik na pewno będzie większy od sprawdzanej liczby.

## Przykładowe dane i wynik:

Dla  $n=7$  (przeszukiwany przedział od 10 do  $10^{7+9}$ ) program zwraca następujący wynik:

```
/Users/lukaszczmarek/PycharmProjects/algorytmika/venv/bin/python /Users/lukaszczmarek/PycharmProjects
153
370
371
407
1634
8208
9474
54748
92727
93084
548834
1741725
4210818
9800817
9926315

Process finished with exit code 0
```

## Zadanie 3

### Palindromy

#### Kod:

```
def reverse(a):
    return int(str(a)[::-1])

def generatePalindrome(a, start, count):
    if a == reverse(a):
        print("Got palindrome {} from {} after {}
transformations!".format(a, start, count))
    else:
        try:
            generatePalindrome(a + reverse(a), start, count+1)
        except:
            print("Cold not achieve a palindrome from {} after {}
transformations...".format(start, count))

if __name__ == "__main__":
    for i in range(10, 200):
        generatePalindrome(i, i, 0)
```

#### Opis:

Do rozwiązania zastosowano algorytm rekurencyjny. Główna pętla programu wywołuje funkcję rekurencyjną *generatePalindrome* dla kolejnych liczb naturalnych od 10 do 200. Funkcja ta sprawdza czy liczba przekazana jako argument jest palindromem (zastosowano funkcję pomocniczą *reverse*, która odwraca kolejność cyfr w danej liczbie) i w takim wypadku wyświetla odpowiedni komunikat zawierający informację o tym jaki palindrom osiągnięto, z jakiej liczby początkowej oraz po ilu przekształceniach. Jeśli argument funkcji nie jest palindromem to wywołuje ona rekurencyjnie samą siebie ze zmienionym argumentem będącym sumą sprawdzanej liczby i liczby otrzymanej przez odwrócenie kolejności jej cyfr. Jednocześnie inkrementowany jest licznik transformacji.

Ze względu na możliwość spowodowania błędu z powodu przekroczenia maksymalnej ilości wywołań rekurencyjnych kolejne wywołanie realizowane jest w konstrukcji try - except. W przypadku wystąpienia wyjątku wyświetlany jest komunikat dla jakiej liczby nie udało się osiągnąć palindromu i po jakiej ilości transformacji.

#### Wynik:

Poniżej przedstawiono wynik uruchomienia programu dla liczb od 10 do 200. Uwagę zwraca liczba 196, dla której nawet po 994 przekształceniach (maksymalna ilość wywołań rekurencyjnych dla komputera, na którym uruchomiono program) nie udało się uzyskać palindromu.

```
/Users/lukaszkaaczmarek/PycharmProjects/algorytmika/venv/bin/python /Users/lukaszkaaczmarek/PycharmProjects/algorytmika/palindromy.py
```

```
Got palindrome 11 from 10 after 1 transformations!
Got palindrome 11 from 11 after 0 transformations!
Got palindrome 33 from 12 after 1 transformations!
Got palindrome 44 from 13 after 1 transformations!
Got palindrome 55 from 14 after 1 transformations!
```

Got palindrome 66 from 15 after 1 transformations!  
Got palindrome 77 from 16 after 1 transformations!  
Got palindrome 88 from 17 after 1 transformations!  
Got palindrome 99 from 18 after 1 transformations!  
Got palindrome 121 from 19 after 2 transformations!  
Got palindrome 22 from 20 after 1 transformations!  
Got palindrome 33 from 21 after 1 transformations!  
Got palindrome 22 from 22 after 0 transformations!  
Got palindrome 55 from 23 after 1 transformations!  
Got palindrome 66 from 24 after 1 transformations!  
Got palindrome 77 from 25 after 1 transformations!  
Got palindrome 88 from 26 after 1 transformations!  
Got palindrome 99 from 27 after 1 transformations!  
Got palindrome 121 from 28 after 2 transformations!  
Got palindrome 121 from 29 after 1 transformations!  
Got palindrome 33 from 30 after 1 transformations!  
Got palindrome 44 from 31 after 1 transformations!  
Got palindrome 55 from 32 after 1 transformations!  
Got palindrome 33 from 33 after 0 transformations!  
Got palindrome 77 from 34 after 1 transformations!  
Got palindrome 88 from 35 after 1 transformations!  
Got palindrome 99 from 36 after 1 transformations!  
Got palindrome 121 from 37 after 2 transformations!  
Got palindrome 121 from 38 after 1 transformations!  
Got palindrome 363 from 39 after 2 transformations!  
Got palindrome 44 from 40 after 1 transformations!  
Got palindrome 55 from 41 after 1 transformations!  
Got palindrome 66 from 42 after 1 transformations!  
Got palindrome 77 from 43 after 1 transformations!  
Got palindrome 44 from 44 after 0 transformations!  
Got palindrome 99 from 45 after 1 transformations!  
Got palindrome 121 from 46 after 2 transformations!  
Got palindrome 121 from 47 after 1 transformations!  
Got palindrome 363 from 48 after 2 transformations!  
Got palindrome 484 from 49 after 2 transformations!  
Got palindrome 55 from 50 after 1 transformations!  
Got palindrome 66 from 51 after 1 transformations!  
Got palindrome 77 from 52 after 1 transformations!  
Got palindrome 88 from 53 after 1 transformations!  
Got palindrome 99 from 54 after 1 transformations!  
Got palindrome 55 from 55 after 0 transformations!  
Got palindrome 121 from 56 after 1 transformations!  
Got palindrome 363 from 57 after 2 transformations!  
Got palindrome 484 from 58 after 2 transformations!  
Got palindrome 1111 from 59 after 3 transformations!  
Got palindrome 66 from 60 after 1 transformations!  
Got palindrome 77 from 61 after 1 transformations!  
Got palindrome 88 from 62 after 1 transformations!  
Got palindrome 99 from 63 after 1 transformations!  
Got palindrome 121 from 64 after 2 transformations!  
Got palindrome 121 from 65 after 1 transformations!  
Got palindrome 66 from 66 after 0 transformations!  
Got palindrome 484 from 67 after 2 transformations!  
Got palindrome 1111 from 68 after 3 transformations!  
Got palindrome 4884 from 69 after 4 transformations!  
Got palindrome 77 from 70 after 1 transformations!  
Got palindrome 88 from 71 after 1 transformations!  
Got palindrome 99 from 72 after 1 transformations!  
Got palindrome 121 from 73 after 2 transformations!  
Got palindrome 121 from 74 after 1 transformations!



Got palindrome 363 from 75 after 2 transformations!  
Got palindrome 484 from 76 after 2 transformations!  
Got palindrome 77 from 77 after 0 transformations!  
Got palindrome 4884 from 78 after 4 transformations!  
Got palindrome 44044 from 79 after 6 transformations!  
Got palindrome 88 from 80 after 1 transformations!  
Got palindrome 99 from 81 after 1 transformations!  
Got palindrome 121 from 82 after 2 transformations!  
Got palindrome 121 from 83 after 1 transformations!  
Got palindrome 363 from 84 after 2 transformations!  
Got palindrome 484 from 85 after 2 transformations!  
Got palindrome 1111 from 86 after 3 transformations!  
Got palindrome 4884 from 87 after 4 transformations!  
Got palindrome 88 from 88 after 0 transformations!  
Got palindrome 8813200023188 from 89 after 24 transformations!  
Got palindrome 99 from 90 after 1 transformations!  
Got palindrome 121 from 91 after 2 transformations!  
Got palindrome 121 from 92 after 1 transformations!  
Got palindrome 363 from 93 after 2 transformations!  
Got palindrome 484 from 94 after 2 transformations!  
Got palindrome 1111 from 95 after 3 transformations!  
Got palindrome 4884 from 96 after 4 transformations!  
Got palindrome 44044 from 97 after 6 transformations!  
Got palindrome 8813200023188 from 98 after 24 transformations!  
Got palindrome 99 from 99 after 0 transformations!  
Got palindrome 101 from 100 after 1 transformations!  
Got palindrome 101 from 101 after 0 transformations!  
Got palindrome 303 from 102 after 1 transformations!  
Got palindrome 404 from 103 after 1 transformations!  
Got palindrome 505 from 104 after 1 transformations!  
Got palindrome 606 from 105 after 1 transformations!  
Got palindrome 707 from 106 after 1 transformations!  
Got palindrome 808 from 107 after 1 transformations!  
Got palindrome 909 from 108 after 1 transformations!  
Got palindrome 1111 from 109 after 2 transformations!  
Got palindrome 121 from 110 after 1 transformations!  
Got palindrome 111 from 111 after 0 transformations!  
Got palindrome 323 from 112 after 1 transformations!  
Got palindrome 424 from 113 after 1 transformations!  
Got palindrome 525 from 114 after 1 transformations!  
Got palindrome 626 from 115 after 1 transformations!  
Got palindrome 727 from 116 after 1 transformations!  
Got palindrome 828 from 117 after 1 transformations!  
Got palindrome 929 from 118 after 1 transformations!  
Got palindrome 1331 from 119 after 2 transformations!  
Got palindrome 141 from 120 after 1 transformations!  
Got palindrome 121 from 121 after 0 transformations!  
Got palindrome 343 from 122 after 1 transformations!  
Got palindrome 444 from 123 after 1 transformations!  
Got palindrome 545 from 124 after 1 transformations!  
Got palindrome 646 from 125 after 1 transformations!  
Got palindrome 747 from 126 after 1 transformations!  
Got palindrome 848 from 127 after 1 transformations!  
Got palindrome 949 from 128 after 1 transformations!  
Got palindrome 1551 from 129 after 2 transformations!  
Got palindrome 161 from 130 after 1 transformations!  
Got palindrome 131 from 131 after 0 transformations!  
Got palindrome 363 from 132 after 1 transformations!  
Got palindrome 464 from 133 after 1 transformations!  
Got palindrome 565 from 134 after 1 transformations!

Got palindrome 666 from 135 after 1 transformations!  
Got palindrome 767 from 136 after 1 transformations!  
Got palindrome 868 from 137 after 1 transformations!  
Got palindrome 969 from 138 after 1 transformations!  
Got palindrome 1771 from 139 after 2 transformations!  
Got palindrome 181 from 140 after 1 transformations!  
Got palindrome 141 from 141 after 0 transformations!  
Got palindrome 383 from 142 after 1 transformations!  
Got palindrome 484 from 143 after 1 transformations!  
Got palindrome 585 from 144 after 1 transformations!  
Got palindrome 686 from 145 after 1 transformations!  
Got palindrome 787 from 146 after 1 transformations!  
Got palindrome 888 from 147 after 1 transformations!  
Got palindrome 989 from 148 after 1 transformations!  
Got palindrome 1991 from 149 after 2 transformations!  
Got palindrome 303 from 150 after 2 transformations!  
Got palindrome 151 from 151 after 0 transformations!  
Got palindrome 707 from 152 after 2 transformations!  
Got palindrome 909 from 153 after 2 transformations!  
Got palindrome 1111 from 154 after 2 transformations!  
Got palindrome 4444 from 155 after 3 transformations!  
Got palindrome 6666 from 156 after 3 transformations!  
Got palindrome 8888 from 157 after 3 transformations!  
Got palindrome 11011 from 158 after 3 transformations!  
Got palindrome 1221 from 159 after 2 transformations!  
Got palindrome 343 from 160 after 2 transformations!  
Got palindrome 161 from 161 after 0 transformations!  
Got palindrome 747 from 162 after 2 transformations!  
Got palindrome 949 from 163 after 2 transformations!  
Got palindrome 2662 from 164 after 3 transformations!  
Got palindrome 4884 from 165 after 3 transformations!  
Got palindrome 45254 from 166 after 5 transformations!  
Got palindrome 88555588 from 167 after 11 transformations!  
Got palindrome 13431 from 168 after 3 transformations!  
Got palindrome 1441 from 169 after 2 transformations!  
Got palindrome 383 from 170 after 2 transformations!  
Got palindrome 171 from 171 after 0 transformations!  
Got palindrome 787 from 172 after 2 transformations!  
Got palindrome 989 from 173 after 2 transformations!  
Got palindrome 5115 from 174 after 4 transformations!  
Got palindrome 9559 from 175 after 4 transformations!  
Got palindrome 44044 from 176 after 5 transformations!  
Got palindrome 8836886388 from 177 after 15 transformations!  
Got palindrome 15851 from 178 after 3 transformations!  
Got palindrome 1661 from 179 after 2 transformations!  
Got palindrome 747 from 180 after 3 transformations!  
Got palindrome 181 from 181 after 0 transformations!  
Got palindrome 45254 from 182 after 6 transformations!  
Got palindrome 13431 from 183 after 4 transformations!  
Got palindrome 2552 from 184 after 3 transformations!  
Got palindrome 4774 from 185 after 3 transformations!  
Got palindrome 6996 from 186 after 3 transformations!  
Got palindrome 8813200023188 from 187 after 23 transformations!  
Got palindrome 233332 from 188 after 7 transformations!  
Got palindrome 1881 from 189 after 2 transformations!  
Got palindrome 45254 from 190 after 7 transformations!  
Got palindrome 191 from 191 after 0 transformations!  
Got palindrome 6996 from 192 after 4 transformations!  
Got palindrome 233332 from 193 after 8 transformations!  
Got palindrome 2992 from 194 after 3 transformations!

Got palindrome 9339 from 195 after 4 transformations!  
Cold not achieve a palindrome from 196 after 994 transformations...  
Got palindrome 881188 from 197 after 7 transformations!  
Got palindrome 79497 from 198 after 5 transformations!  
Got palindrome 3113 from 199 after 3 transformations!

Process finished with exit code 0

## Zadanie 4

### Sito Eratostenesa

#### Kod:

```
import numpy as np
import time

def eratostenes_list(n):
    numbers = list(range(2, n))
    i = 0
    while (i < len(numbers)):
        number = numbers[i]
        x = number
        while (x < n):
            x += number
            try:
                numbers.remove(x)
            except:
                pass
        i += 1
    return numbers

def eratostenes_dict(n):
    numbers = {i: i for i in range(2, n)}
    i = 2
    while (i < len(numbers)):
        x = i
        while (x < n):
            x += i
            try:
                del numbers[x]
            except:
                pass
        i += 1
    return numbers.keys()

def eratostenes_numpy(n):
    numbers = np.array(range(2, n))
    i = 0
    while (i < len(numbers)):
        number = numbers[i]
        x = number
        while (x < n):
            x += number
            try:
                numbers = numbers[numbers != x]
            except:
                pass
        i += 1
    return numbers
```

```

def tester(function, n):
    start = time.perf_counter()
    print(function(n))
    end = time.perf_counter()
    print("Completed with the {} in {} s.\n".format(function.__name__,
end - start))

if __name__ == "__main__":
    n = 10000
    print("Erathosthenes sieve for range 2 to", n, "\n")
    functions = (eratosthenes_list, eratosthenes_dict, eratosthenes_numpy)
    for function in functions:
        tester(function, n)

```

## Opis:

W programie zaimplementowano 3 funkcje znajdujące liczby pierwsze z zakresie od 2 do n metodą sita Eratostenesa ale z wykorzystaniem różnych struktur danych:

*eratosthenes\_list* - wykorzystuje listę

*eratosthenes\_dict* - wykorzystuje słownik

*eratosthenes\_numpy* - wykorzystuje tablicę `numpy.array`

Funkcje *eratosthenes\_list* i *eratosthenes\_numpy* działają w bardzo podobny sposób. W pierwszej kolejności tworzona jest struktura danych *numbers* (lista lub tablica `numpy`) zawierająca liczby naturalne od 2 do n. Zmienna *i* przechowuje indeks wskazujący na początek struktury danych. Następnie w zmiennej *number* przechowywana jest liczba na którą wskazuje *i*. Z *numbers* usuwane są wielokrotności *number* (zmienna *x* zwiększana o *number* w każdym przebiegu pętli do momentu aż *x* > n). Ze względu na możliwość wystąpienia błędu przy próbie usunięcia liczby, która już wcześniej została usunięta ze struktury danych zastosowano konstrukcję try - except. Następnie indeks *i* zwiększany jest o 1 tak aby wskazywał na kolejną liczbę w *numbers*, której wielokrotności należy usunąć. W momencie kiedy *i* wskazuje na koniec struktury danych *numbers* jest ona zwracana jako zawierająca liczby pierwsze.

Funkcja *eratosthenes\_dict* działa w sposób zbliżony do wyżej opisanych jednak ze względu na to, że słownik nie zapewnia określonej kolejności elementów nie można zastosować indeksów wskazujących na kolejną nie wykreśloną liczbę. Stąd funkcja próbuje usuwać wielokrotności kolejnych liczby naturalnych od 2 do n, co pozornie wydaje się mniej efektywne niż w przypadku listy czy tablicy `numpy`, do których elementów można odwoływać się przez indeksowanie. Jednak zrealizowanie podobnego algorytmu dla słownika wymagałoby tworzenia tymczasowej sortowalnej struktury (np. krotka) zawierającej klucze słownika oraz jej posortowania - co jest operacją czasochłonną. Pomimo tej niedoskonałości ze względu na szybkość dostępu i usuwania elementów funkcja wykorzystująca słownik działa najszybciej z przetestowanych, co widać w wyniku pomiaru czasu działania poszczególnych funkcji. W tym celu przygotowano funkcję *tester()*, która uruchamia inną funkcję przekazywaną jej jako parametr oraz dokonuje pomiaru czasu jej wykonania przy pomocy *perf\_counter()* z modułu *time*. Zdecydowanie najwolniej działa funkcja korzystająca ze słownika.

## Wynik:

/Users/lukaszczmarek/PycharmProjects/algorytmika/venv/bin/python /Users/lukaszczmarek/PycharmProjects/algorytmika/eratosthenes.py

Erathosthenes sieve for range 2 to 10000

[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997, 1009, 1013, 1019, 1021, 1031, 1033, 1039, 1049, 1051, 1061, 1063, 1069, 1087, 1091, 1093, 1097, 1103, 1109, 1117, 1123, 1129, 1151, 1153, 1163, 1171, 1181, 1187, 1193, 1201, 1213, 1217, 1223, 1229, 1231, 1237, 1249, 1259, 1277, 1279, 1283, 1289, 1291, 1297, 1301, 1303, 1307, 1319, 1321, 1327, 1361, 1367, 1373, 1381, 1399, 1409, 1423, 1427, 1429, 1433, 1439, 1447, 1451, 1453, 1459, 1471, 1481, 1483, 1487, 1489, 1493, 1499, 1511, 1523, 1531, 1543, 1549, 1553, 1559, 1567, 1571, 1579, 1583, 1597, 1601, 1607, 1609, 1613, 1619, 1621, 1627, 1637, 1657, 1663, 1667, 1669, 1693, 1697, 1699, 1709, 1721, 1723, 1733, 1741, 1747, 1753, 1759, 1777, 1783, 1787, 1789, 1801, 1811, 1823, 1831, 1847, 1861, 1867, 1871, 1873, 1877, 1879, 1889, 1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993, 1997, 1999, 2003, 2011, 2017, 2027, 2029, 2039, 2053, 2063, 2069, 2081, 2083, 2087, 2089, 2099, 2111, 2113, 2129, 2131, 2137, 2141, 2143, 2153, 2161, 2179, 2203, 2207, 2213, 2221, 2237, 2239, 2243, 2251, 2267, 2269, 2273, 2281, 2287, 2293, 2297, 2309, 2311, 2333, 2339, 2341, 2347, 2351, 2357, 2371, 2377, 2381, 2383, 2389, 2393, 2399, 2411, 2417, 2423, 2437, 2441, 2447, 2459, 2467, 2473, 2477, 2503, 2521, 2531, 2539, 2543, 2549, 2551, 2557, 2579, 2591, 2593, 2609, 2617, 2621, 2633, 2647, 2657, 2659, 2663, 2671, 2677, 2683, 2687, 2689, 2693, 2699, 2707, 2711, 2713, 2719, 2729, 2731, 2741, 2749, 2753, 2767, 2777, 2789, 2791, 2797, 2801, 2803, 2819, 2833, 2837, 2843, 2851, 2857, 2861, 2879, 2887, 2897, 2903, 2909, 2917, 2927, 2939, 2953, 2957, 2963, 2969, 2971, 2999, 3001, 3011, 3019, 3023, 3037, 3041, 3049, 3061, 3067, 3079, 3083, 3089, 3109, 3119, 3121, 3137, 3163, 3167, 3169, 3181, 3187, 3191, 3203, 3209, 3217, 3221, 3229, 3251, 3253, 3257, 3259, 3271, 3299, 3301, 3307, 3313, 3319, 3323, 3329, 3331, 3343, 3347, 3359, 3361, 3371, 3373, 3389, 3391, 3407, 3413, 3433, 3449, 3457, 3461, 3463, 3467, 3469, 3491, 3499, 3511, 3517, 3527, 3529, 3533, 3539, 3541, 3547, 3557, 3559, 3571, 3581, 3583, 3593, 3607, 3613, 3617, 3623, 3631, 3637, 3643, 3659, 3671, 3673, 3677, 3691, 3697, 3701, 3709, 3719, 3727, 3733, 3739, 3761, 3767, 3769, 3779, 3793, 3797, 3803, 3821, 3823, 3833, 3847, 3851, 3853, 3863, 3877, 3881, 3889, 3907, 3911, 3917, 3919, 3923, 3929, 3931, 3943, 3947, 3967, 3989, 4001, 4003, 4007, 4013, 4019, 4021, 4027, 4049, 4051, 4057, 4073, 4079, 4091, 4093, 4099, 4111, 4127, 4129, 4133, 4139, 4153, 4157, 4159, 4177, 4201, 4211, 4217, 4219, 4229, 4231, 4241, 4243, 4253, 4259, 4261, 4271, 4273, 4283, 4289, 4297, 4327, 4337, 4339, 4349, 4357, 4363, 4373, 4391, 4397, 4409, 4421, 4423, 4441, 4447, 4451, 4457, 4463, 4481, 4483, 4493, 4507, 4513, 4517, 4519, 4523, 4547, 4549, 4561, 4567, 4583, 4591, 4597, 4603, 4621, 4637, 4639, 4643, 4649, 4651, 4657, 4663, 4673, 4679, 4691, 4703, 4721, 4723, 4729, 4733, 4751, 4759, 4783, 4787, 4789, 4793, 4799, 4801, 4813, 4817, 4831, 4861, 4871, 4877, 4889, 4903, 4909, 4919, 4931, 4933, 4937, 4943, 4951, 4957, 4967, 4969, 4973, 4987, 4993, 4999, 5003, 5009, 5011, 5021, 5023, 5039, 5051, 5059, 5077, 5081, 5087, 5099, 5101, 5107, 5113, 5119, 5147, 5153, 5167, 5171, 5179, 5189, 5197, 5209, 5227, 5231, 5233, 5237, 5261, 5273, 5279, 5281, 5297, 5303, 5309, 5323, 5333, 5347, 5351, 5381, 5387, 5393, 5399, 5407, 5413, 5417, 5419, 5431, 5437, 5441, 5443, 5449, 5471, 5477, 5479, 5483, 5501, 5503, 5507, 5519, 5521, 5527, 5531, 5557, 5563, 5569, 5573, 5581, 5591, 5623, 5639, 5641, 5647, 5651, 5653, 5657, 5659, 5669, 5683, 5689, 5693, 5701, 5711, 5717, 5737, 5741, 5743, 5749, 5779, 5783, 5791, 5801, 5807, 5813, 5821, 5827, 5839, 5843, 5849, 5851, 5857, 5861, 5867, 5869, 5879, 5881, 5897, 5903, 5923, 5927, 5939, 5953, 5981, 5987, 6007, 6011, 6029, 6037, 6043, 6047, 6053, 6067, 6073, 6079, 6089, 6091, 6101, 6113, 6121, 6131, 6133, 6143, 6151, 6163, 6173, 6197, 6199, 6203, 6211, 6217, 6221, 6229, 6247, 6257, 6263, 6269, 6271, 6277, 6287, 6299, 6301, 6311, 6317, 6323, 6329, 6337, 6343, 6353, 6359, 6361, 6367, 6373, 6379, 6389, 6397, 6421, 6427,

6449, 6451, 6469, 6473, 6481, 6491, 6521, 6529, 6547, 6551, 6553, 6563, 6569, 6571, 6577, 6581, 6599, 6607, 6619, 6637, 6653, 6659, 6661, 6673, 6679, 6689, 6691, 6701, 6703, 6709, 6719, 6733, 6737, 6761, 6763, 6779, 6781, 6791, 6793, 6803, 6823, 6827, 6829, 6833, 6841, 6857, 6863, 6869, 6871, 6883, 6899, 6907, 6911, 6917, 6947, 6949, 6959, 6961, 6967, 6971, 6977, 6983, 6991, 6997, 7001, 7013, 7019, 7027, 7039, 7043, 7057, 7069, 7079, 7103, 7109, 7121, 7127, 7129, 7151, 7159, 7177, 7187, 7193, 7207, 7211, 7213, 7219, 7229, 7237, 7243, 7247, 7253, 7283, 7297, 7307, 7309, 7321, 7331, 7333, 7349, 7351, 7369, 7393, 7411, 7417, 7433, 7451, 7457, 7459, 7477, 7481, 7487, 7489, 7499, 7507, 7517, 7523, 7529, 7537, 7541, 7547, 7549, 7559, 7561, 7573, 7577, 7583, 7589, 7591, 7603, 7607, 7621, 7639, 7643, 7649, 7669, 7673, 7681, 7687, 7691, 7699, 7703, 7717, 7723, 7727, 7741, 7753, 7757, 7759, 7789, 7793, 7817, 7823, 7829, 7841, 7853, 7867, 7873, 7877, 7879, 7883, 7901, 7907, 7919, 7927, 7933, 7937, 7949, 7951, 7963, 7993, 8009, 8011, 8017, 8039, 8053, 8059, 8069, 8081, 8087, 8089, 8093, 8101, 8111, 8117, 8123, 8147, 8161, 8167, 8171, 8179, 8191, 8209, 8219, 8221, 8231, 8233, 8237, 8243, 8263, 8269, 8273, 8287, 8291, 8293, 8297, 8311, 8317, 8329, 8353, 8363, 8369, 8377, 8387, 8389, 8419, 8423, 8429, 8431, 8443, 8447, 8461, 8467, 8501, 8513, 8521, 8527, 8537, 8539, 8543, 8563, 8573, 8581, 8597, 8599, 8609, 8623, 8627, 8629, 8641, 8647, 8663, 8669, 8677, 8681, 8689, 8693, 8699, 8707, 8713, 8719, 8731, 8737, 8741, 8747, 8753, 8761, 8779, 8783, 8803, 8807, 8819, 8821, 8831, 8837, 8839, 8849, 8861, 8863, 8867, 8887, 8893, 8923, 8929, 8933, 8941, 8951, 8963, 8969, 8971, 8999, 9001, 9007, 9011, 9013, 9029, 9041, 9043, 9049, 9059, 9067, 9091, 9103, 9109, 9127, 9133, 9137, 9151, 9157, 9161, 9173, 9181, 9187, 9199, 9203, 9209, 9221, 9227, 9239, 9241, 9257, 9277, 9281, 9283, 9293, 9311, 9319, 9323, 9337, 9341, 9343, 9349, 9371, 9377, 9391, 9397, 9403, 9413, 9419, 9421, 9431, 9433, 9437, 9439, 9461, 9463, 9467, 9473, 9479, 9491, 9497, 9511, 9521, 9533, 9539, 9547, 9551, 9587, 9601, 9613, 9619, 9623, 9629, 9631, 9643, 9649, 9661, 9677, 9679, 9689, 9697, 9719, 9721, 9733, 9739, 9743, 9749, 9767, 9769, 9781, 9787, 9791, 9803, 9811, 9817, 9829, 9833, 9839, 9851, 9857, 9859, 9871, 9883, 9887, 9901, 9907, 9923, 9929, 9931, 9941, 9949, 9967, 9973]

Completed with the eratostenes\_list in 0.726709635 s.

dict\_keys([2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997, 1009, 1013, 1019, 1021, 1031, 1033, 1039, 1049, 1051, 1061, 1063, 1069, 1087, 1091, 1093, 1097, 1103, 1109, 1117, 1123, 1129, 1151, 1153, 1163, 1171, 1181, 1187, 1193, 1201, 1213, 1217, 1223, 1229, 1231, 1237, 1249, 1259, 1277, 1279, 1283, 1289, 1291, 1297, 1301, 1303, 1307, 1319, 1321, 1327, 1361, 1367, 1373, 1381, 1399, 1409, 1423, 1427, 1429, 1433, 1439, 1447, 1451, 1453, 1459, 1471, 1481, 1483, 1487, 1489, 1493, 1499, 1511, 1523, 1531, 1543, 1549, 1553, 1559, 1567, 1571, 1579, 1583, 1597, 1601, 1607, 1609, 1613, 1619, 1621, 1627, 1637, 1657, 1663, 1667, 1669, 1693, 1697, 1699, 1709, 1721, 1723, 1733, 1741, 1747, 1753, 1759, 1777, 1783, 1787, 1789, 1801, 1811, 1823, 1831, 1847, 1861, 1867, 1871, 1873, 1877, 1879, 1889, 1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993, 1997, 1999, 2003, 2011, 2017, 2027, 2029, 2039, 2053, 2063, 2069, 2081, 2083, 2087, 2089, 2099, 2111, 2113, 2129, 2131, 2137, 2141, 2143, 2153, 2161, 2179, 2203, 2207, 2213, 2221, 2237, 2239, 2243, 2251, 2267, 2269, 2273, 2281, 2287, 2293, 2297, 2309, 2311, 2333, 2339, 2341, 2347, 2351, 2357, 2371, 2377, 2381, 2383, 2389, 2393, 2399, 2411, 2417, 2423, 2437, 2441, 2447, 2459, 2467, 2473, 2477, 2503, 2521, 2531, 2539, 2543, 2549, 2551, 2557, 2579, 2591, 2593, 2609, 2617, 2621, 2633, 2647, 2657, 2659, 2663, 2671, 2677, 2683, 2687, 2689, 2693, 2699, 2707, 2711, 2713, 2719, 2729, 2731, 2741, 2749, 2753, 2767, 2777, 2789, 2791, 2797, 2801, 2803, 2819, 2833, 2837, 2843, 2851, 2857, 2861, 2879, 2887, 2897, 2903, 2909, 2917, 2927, 2939, 2953, 2957, 2963, 2969, 2971, 2999, 3001, 3011, 3019, 3023, 3037, 3041, 3049, 3061, 3067, 3079, 3083, 3089, 3109, 3119, 3121, 3137, 3163, 3167, 3169, 3181, 3187, 3191, 3203, 3209, 3217, 3221, 3229, 3251, 3253, 3257, 3259, 3271, 3299, 3301, 3307, 3313, 3319, 3323, 3329, 3331, 3343, 3347, 3359, 3361, 3371, 3373, 3389, 3391, 3407, 3413, 3433, 3449, 3457, 3461, 3463, 3467, 3469, 3491, 3499, 3511, 3517, 3527, 3529, 3533, 3539, 3541, 3547, 3557, 3559, 3571, 3581, 3583, 3593, 3607, 3613,

3617, 3623, 3631, 3637, 3643, 3659, 3671, 3673, 3677, 3691, 3697, 3701, 3709, 3719, 3727, 3733, 3739, 3761, 3767, 3769, 3779, 3793, 3797, 3803, 3821, 3823, 3833, 3847, 3851, 3853, 3863, 3877, 3881, 3889, 3907, 3911, 3917, 3919, 3923, 3929, 3931, 3943, 3947, 3967, 3989, 4001, 4003, 4007, 4013, 4019, 4021, 4027, 4049, 4051, 4057, 4073, 4079, 4091, 4093, 4099, 4111, 4127, 4129, 4133, 4139, 4153, 4157, 4159, 4177, 4201, 4211, 4217, 4219, 4229, 4231, 4241, 4243, 4253, 4259, 4261, 4271, 4273, 4283, 4289, 4297, 4327, 4337, 4339, 4349, 4357, 4363, 4373, 4391, 4397, 4409, 4421, 4423, 4441, 4447, 4451, 4457, 4463, 4481, 4483, 4493, 4507, 4513, 4517, 4519, 4523, 4547, 4549, 4561, 4567, 4583, 4591, 4597, 4603, 4621, 4637, 4639, 4643, 4649, 4651, 4657, 4663, 4673, 4679, 4691, 4703, 4721, 4723, 4729, 4733, 4751, 4759, 4783, 4787, 4789, 4793, 4799, 4801, 4813, 4817, 4831, 4861, 4871, 4877, 4889, 4903, 4909, 4919, 4931, 4933, 4937, 4943, 4951, 4957, 4967, 4969, 4973, 4987, 4993, 4999, 5003, 5009, 5011, 5021, 5023, 5039, 5051, 5059, 5077, 5081, 5087, 5099, 5101, 5107, 5113, 5119, 5147, 5153, 5167, 5171, 5179, 5189, 5197, 5209, 5227, 5231, 5233, 5237, 5261, 5273, 5279, 5281, 5297, 5303, 5309, 5323, 5333, 5347, 5351, 5381, 5387, 5393, 5399, 5407, 5413, 5417, 5419, 5431, 5437, 5441, 5443, 5449, 5471, 5477, 5479, 5483, 5501, 5503, 5507, 5519, 5521, 5527, 5531, 5557, 5563, 5569, 5573, 5581, 5591, 5623, 5639, 5641, 5647, 5651, 5653, 5657, 5659, 5669, 5683, 5689, 5693, 5701, 5711, 5717, 5737, 5741, 5743, 5749, 5779, 5783, 5791, 5801, 5807, 5813, 5821, 5827, 5839, 5843, 5849, 5851, 5857, 5861, 5867, 5869, 5879, 5881, 5897, 5903, 5923, 5927, 5939, 5953, 5981, 5987, 6007, 6011, 6029, 6037, 6043, 6047, 6053, 6067, 6073, 6079, 6089, 6091, 6101, 6113, 6121, 6131, 6133, 6143, 6151, 6163, 6173, 6197, 6199, 6203, 6211, 6217, 6221, 6229, 6247, 6257, 6263, 6269, 6271, 6277, 6287, 6299, 6301, 6311, 6317, 6323, 6329, 6337, 6343, 6353, 6359, 6361, 6367, 6373, 6379, 6389, 6397, 6421, 6427, 6449, 6451, 6469, 6473, 6481, 6491, 6521, 6529, 6547, 6551, 6553, 6563, 6569, 6571, 6577, 6581, 6599, 6607, 6619, 6637, 6653, 6659, 6661, 6673, 6679, 6689, 6691, 6701, 6703, 6709, 6719, 6733, 6737, 6761, 6763, 6779, 6781, 6791, 6793, 6803, 6823, 6827, 6829, 6833, 6841, 6857, 6863, 6869, 6871, 6883, 6899, 6907, 6911, 6917, 6947, 6949, 6959, 6961, 6967, 6971, 6977, 6983, 6991, 6997, 7001, 7013, 7019, 7027, 7039, 7043, 7057, 7069, 7079, 7103, 7109, 7121, 7127, 7129, 7151, 7159, 7177, 7187, 7193, 7207, 7211, 7213, 7219, 7229, 7237, 7243, 7247, 7253, 7283, 7297, 7307, 7309, 7321, 7331, 7333, 7349, 7351, 7369, 7393, 7411, 7417, 7433, 7451, 7457, 7459, 7477, 7481, 7487, 7489, 7499, 7507, 7517, 7523, 7529, 7537, 7541, 7547, 7549, 7559, 7561, 7573, 7577, 7583, 7589, 7591, 7603, 7607, 7621, 7639, 7643, 7649, 7669, 7673, 7681, 7687, 7691, 7699, 7703, 7717, 7723, 7727, 7741, 7753, 7757, 7759, 7789, 7793, 7817, 7823, 7829, 7841, 7853, 7867, 7873, 7877, 7879, 7883, 7901, 7907, 7919, 7927, 7933, 7937, 7949, 7951, 7963, 7993, 8009, 8011, 8017, 8039, 8053, 8059, 8069, 8081, 8087, 8089, 8093, 8101, 8111, 8117, 8123, 8147, 8161, 8167, 8171, 8179, 8191, 8209, 8219, 8221, 8231, 8233, 8237, 8243, 8263, 8269, 8273, 8287, 8291, 8293, 8297, 8311, 8317, 8329, 8353, 8363, 8369, 8377, 8387, 8389, 8419, 8423, 8429, 8431, 8443, 8447, 8461, 8467, 8501, 8513, 8521, 8527, 8537, 8539, 8543, 8563, 8573, 8581, 8597, 8599, 8609, 8623, 8627, 8629, 8641, 8647, 8663, 8669, 8677, 8681, 8689, 8693, 8699, 8707, 8713, 8719, 8731, 8737, 8741, 8747, 8753, 8761, 8779, 8783, 8803, 8807, 8819, 8821, 8831, 8837, 8839, 8849, 8861, 8863, 8867, 8887, 8893, 8923, 8929, 8933, 8941, 8951, 8963, 8969, 8971, 8999, 9001, 9007, 9011, 9013, 9029, 9041, 9043, 9049, 9059, 9067, 9091, 9103, 9109, 9127, 9133, 9137, 9151, 9157, 9161, 9173, 9181, 9187, 9199, 9203, 9209, 9221, 9227, 9239, 9241, 9257, 9277, 9281, 9283, 9293, 9311, 9319, 9323, 9337, 9341, 9343, 9349, 9371, 9377, 9391, 9397, 9403, 9413, 9419, 9421, 9431, 9433, 9437, 9439, 9461, 9463, 9467, 9473, 9479, 9491, 9497, 9511, 9521, 9533, 9539, 9547, 9551, 9587, 9601, 9613, 9619, 9623, 9629, 9631, 9643, 9649, 9661, 9677, 9679, 9689, 9697, 9719, 9721, 9733, 9739, 9743, 9749, 9767, 9769, 9781, 9787, 9791, 9803, 9811, 9817, 9829, 9833, 9839, 9851, 9857, 9859, 9871, 9883, 9887, 9901, 9907, 9923, 9929, 9931, 9941, 9949, 9967, 9973])

Completed with the eratostenes\_dict in 0.02501316099999995 s.

[ 2 3 5 ... 9949 9967 9973]

Completed with the eratostenes\_numpy in 0.20489040599999997 s.

Process finished with exit code 0



## Zadanie 5

### Licznik słów

#### Kod:

```
import operator

text = open("pan_tadeusz.txt", "r").readlines()
words = {}
for line in text:
    for word in line.split():
        word = word.strip(".,;:!?()[ ]-").lower()

        if (len(word) > 4):
            try:
                words[word] += 1
            except:
                words[word] = 1

sorted_words = sorted(words.items(), key=operator.itemgetter(1),
reverse=True)
counter = 1
for item in sorted_words[:20]:
    print("{} słowo "{} występuje {} razy.".format(counter,
item[0], item[1]))
    counter += 1
```

#### Opis:

Program wczytuje plik „pan\_tadeusz.txt” znajdujący się w katalogu roboczym i jego zawartość rozdzieloną na poszczególne wiersze zapamiętuje w zmiennej *text*. Następnie dwie zagnieżdżone pętle przechodzą przez każdą linię tekstu i rozdzielają ją na poszczególne wyrazy, z których dodatkowo usuwane są znaki przestankowe i nawiasy znajdujące się na początku lub końcu. Do zliczenia krotności wystąpień poszczególnych wyrazów zastosowano słownik *words*, w którym kluczami są słowa a wartościami krotności ich wystąpień w tekście. Jeśli słowo ma długość większą niż 4 znaki to jeśli w słowniku już znajduje się odpowiedni klucz wartość zwiększana jest o 1. Jeśli słowo pojawia się po raz pierwszy (w słowniku brak odpowiadającego klucza) to tworzona jest para klucz : wartość (słowo:1). W celu znalezienia 20 najliczniej występujących w tekście wyrazów zastosowano funkcję *sorted*, która zwraca posortowaną po wartościach listę par klucz:wartość. Dodatkowo w celu uzyskania malejącej kolejności sortowania funkcję *sorted* wywołano z parametrem *reverse = True*. Ostatnia pętla wypisuje 20 najliczniej występujących słów wraz z informacją o krotności wystąpień.

#### Wynik:

Dla tekstu Pana Tadeusza program wyświetlił następujący wynik:

```
/Users/lukaszczmarek/PycharmProjects/algorytmika/venv/bin/python /Users/lukaszczmarek/PycharmProjects/algorytmika/slowa.py
```

1. słowo "rzekł" występuje 155 razy.
2. słowo "tylko" występuje 147 razy.
3. słowo "hrabia" występuje 128 razy.
4. słowo "sędzia" występuje 126 razy.

5. slowo "tadeusz" wystepuje 106 razy.
6. slowo "przed" wystepuje 101 razy.
7. slowo "przez" wystepuje 99 razy.
8. slowo "jeszcze" wystepuje 97 razy.
9. slowo "gdzie" wystepuje 93 razy.
10. slowo "wszyscy" wystepuje 90 razy.
11. slowo "wojski" wystepuje 89 razy.
12. slowo "potem" wystepuje 86 razy.
13. slowo "teraz" wystepuje 80 razy.
14. slowo "kiedy" wystepuje 73 razy.
15. slowo "który" wystepuje 72 razy.
16. slowo "nawet" wystepuje 71 razy.
17. slowo "znowu" wystepuje 71 razy.
18. slowo "jeśli" wystepuje 68 razy.
19. slowo "jakby" wystepuje 67 razy.
20. slowo "wszystko" wystepuje 66 razy.

Process finished with exit code 0

## Zadanie 6

### Szyfr Gronsfelda

Kod:

```
def getLetter(a, shift):
    return (chr(ord(a)+ shift))

def getShift(key):
    index = -1
    while True:
        index += 1
        if index == len(key):
            index = 0
        yield int(key[index])

def storeFile(targetFileName, text):
    try:
        file = open(targetFileName, "w")
        file.write(text)
        file.close()
    except:
        print("Error while writing to {}
file".format(targetFileName))

def code(sourceFileName, targetFileName, key):
    text = open(sourceFileName, "r").read()
    coded_text = ""

    for letter in text:
        coded_text += getLetter(letter, next(getShift(key)))

    storeFile(targetFileName, coded_text)

def decode(sourceFileName, targetFileName, key):
    coded_text = open(sourceFileName, "r").read()
    decoded_text = ""
    for letter in coded_text:
        decoded_text += getLetter(letter, -1*next(getShift(key)))

    storeFile(targetFileName, decoded_text)

if __name__ == "__main__":
    code("pan_tadeusz.txt", "coded.txt", "145238")
    decode("coded.txt", "decoded.txt", "145238")
```

## Opis:

W programie zaimplementowano trzy funkcje pomocnicze:

*getLetter(a, shift)* - funkcja odczytuje kod ASCII przekazanego do niej znaku *a*, zwiększa go o wartość przekazaną jako parametr *shift* a następnie zwraca znak odpowiadający tak zmienionemu kodowi.

*getShift(key)* - jest to generator który zwraca wartość przesunięcia odpowiadającą kolejnym pozycjom klucza przekazanego jako *key* (string np. „12345”). Jeśli osiągnięto ostatnią cyfrę klucza to wskaźnik ponownie ustawiamy na pierwszą pozycję.

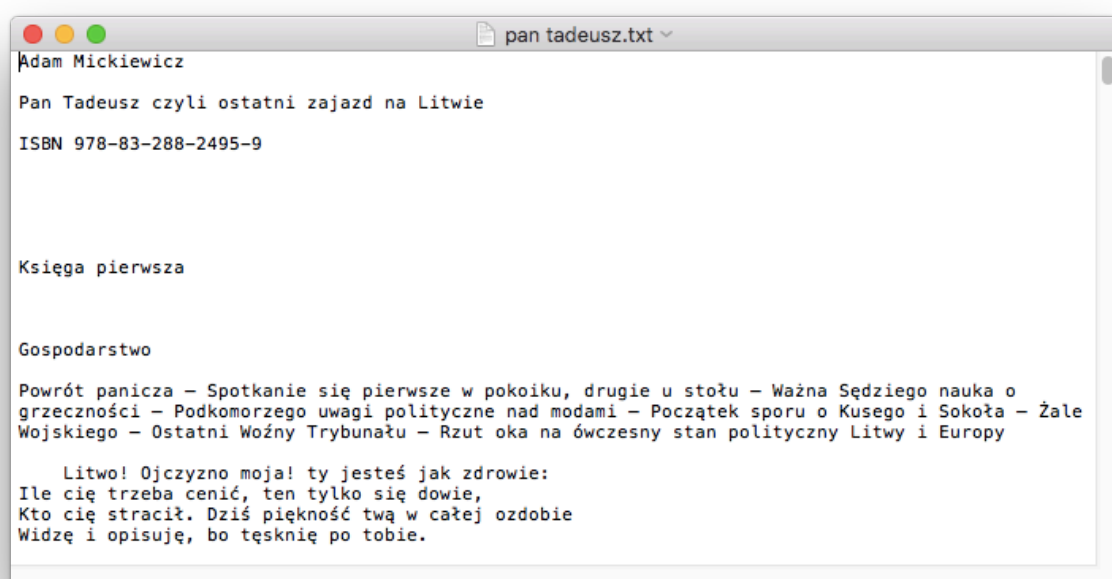
*storeFile(targetFileName, text)* - funkcja zapisuje w pliku *targetFileName* tekst przekazany jako argument.

Szyfrowanie realizowane jest przez funkcję *code*. W pierwszej kolejności wczytuje ona z pliku tekst, który ma być zaszyfrowany. Następnie iteruje ona po każdej literze tekstu wywołując dla niej wyżej opisaną funkcję *getLetter*. Wartość przesunięcia dla kolejnej litery określana jest przez wywołanie generatora *getShift*. Zaszyfrowany tekst tworzony jest przez dodawanie kolejnych zmienionych w ten sposób znaków do zmiennej *coded\_text*. Po zakończeniu pętli szyfrującej zakodowany tekst zapisywany jest do pliku przez wywołanie funkcji *storeFile*.

Deszyfrowanie realizowane jest przez funkcję *decode*, która działa w sposób analogiczny do funkcji szyfrującej. Zasadnicza różnica polega na tym, że wywołanie funkcji *getLetter* następuje z parametrem *shift* uzyskiwanym przez pomnożenie przez -1 wyniku dostarczanego przez generator *getShift()*.

## Wynik:

Program wykonano dla tekstu Pana Tadeusza użytego uprzednio w zadaniu 5 oraz klucza szyfrującego 145238. Ze względu na obszerność tekstu poniżej podano jedynie początkowe fragmenty tekstów oryginalnego, zaszyfrowanego a następnie zdeszyfrowanego za pomocą programu.



```
pan tadeusz.txt
Adam Mickiewicz

Pan Tadeusz czyli ostatni zajazd na Litwie

ISBN 978-83-288-2495-9


Księga pierwsza


Gospodarstwo

Powrót panicza – Spotkanie się pierwsze w pokoiku, drugie u stołu – Ważna Sędziego nauka o grzeczności – Podkomorzego uwagi polityczne nad modami – Początek sporu o Kusego i Sokoła – Żale Wojskiego – Ostatni Woźny Trybunału – Rzut oka na ówczesny stan polityczny Litwy i Europy

    Litwo! Ojczyzno moja! ty jesteś jak zdrowie:
    Ile cię trzeba cenić, ten tylko się dowie,
    Kto cię stracił. Dziś piękność twą w całej ozdobie
    Widzę i opisuję, bo tęsknię po tobie.
```

```
coded.txt
Bebn!Njdljfxjd{
Qbo!Ubefvt{!d{zmj!ptubuo{j!{bkb{e!ob!Mjuxjf
JTC0!:89.94.399.35:6.:

LtjĖhb!qjfsxt{b

Hptqpebstuxp

Qpxsôu!qbojd{b!-!Tqpulbojf!tjĖ!qjfsxt{f!x!qplpjlv-!esvhjf!v!tupNv!-!XbŽob!TĖe{jfhp!obv!b!p!
hs{fd{opšdj!-!Qpelnpns{fhp!vxbhj!qpmjuzd{of!obe!npebnj!-!Qpd{Ĉufl!tqpsv!p!Lvtfhp!j!TplpNb!-!
žbmfiXpktljfhp!-!Ptubuo{j!XpŽoz!UšzcvoBNv!-!S{vu!plb!ob!ôxd{ftoz!tubo!qpmjuzd{oz!Mjuxz!j!Fvspqz

!!!Mjuxp"!Pkd{z{op!npkb"!uz!kftufš!kbl!{espjxf;
Jmf!djĖ!us{fcb!dfojĈ-!ufo!uzmlp!tjĖ!epjxf-
Lup!djĖ!tusbdjN/!E{jš!qjĖlopšĈ!uxĈ!x!dbNfk!p{epcjf
Xje{Ė!j!pqjtvkĖ-!cp!uĖtlojĖ!qp!upcjf/
```

```
decoded.txt
Adam Mickiewicz

Pan Tadeusz czyli ostatni zajazd na Litwie

ISBN 978-83-288-2495-9

Księga pierwsza

Gospodarstwo

Powrót panicza – Spotkanie się pierwsze w pokoiku, drugie u stołu – Ważna Sędziego nauka o
grzeczności – Podkomorzego uwagi polityczne nad modami – Początek sporu o Kusego i Sokoła – Żale
Wojskiego – Ostatni Woźny Trybunału – Rzut oka na ówczesny stan polityczny Litwy i Europy

Litwo! Ojczyzno moja! ty jesteś jak zdrowie:
Ile cię trzeba cenić, ten tylko się dowie,
Kto cię stracił. Dziś piękność twą w całej ozdobie
Widzę i opisuję, bo tęsknię po tobie.
```

## Zadanie 7

### Wykresy

Kod:

```
import matplotlib.pyplot as plt
import numpy as np

def getFunction():
    function_txt = input("Podaj wzor funkcji >>> y=")
    return lambda x: eval(function_txt), function_txt

def getRange():
    while True:
        try:
            leftEnd = input("Podaj lewy koniec przedzialu >>>")
            leftEnd = float(leftEnd)
            rightEnd = input("Podaj prawy koniec przedzialu >>>")
            rightEnd = float(rightEnd)
            if leftEnd < rightEnd:
                break
            else:
                print("Lewy koniec musi byc mniejszy od prawego!")
        except:
            print("Nieprawidłowa wartość!")

    return np.linspace(leftEnd, rightEnd, 201)

if __name__ == "__main__":
    function, function_txt = getFunction()
    x = getRange()
    plt.plot(x, function(x))
    plt.title("y=" + function_txt)
    plt.show()
```

### Opis:

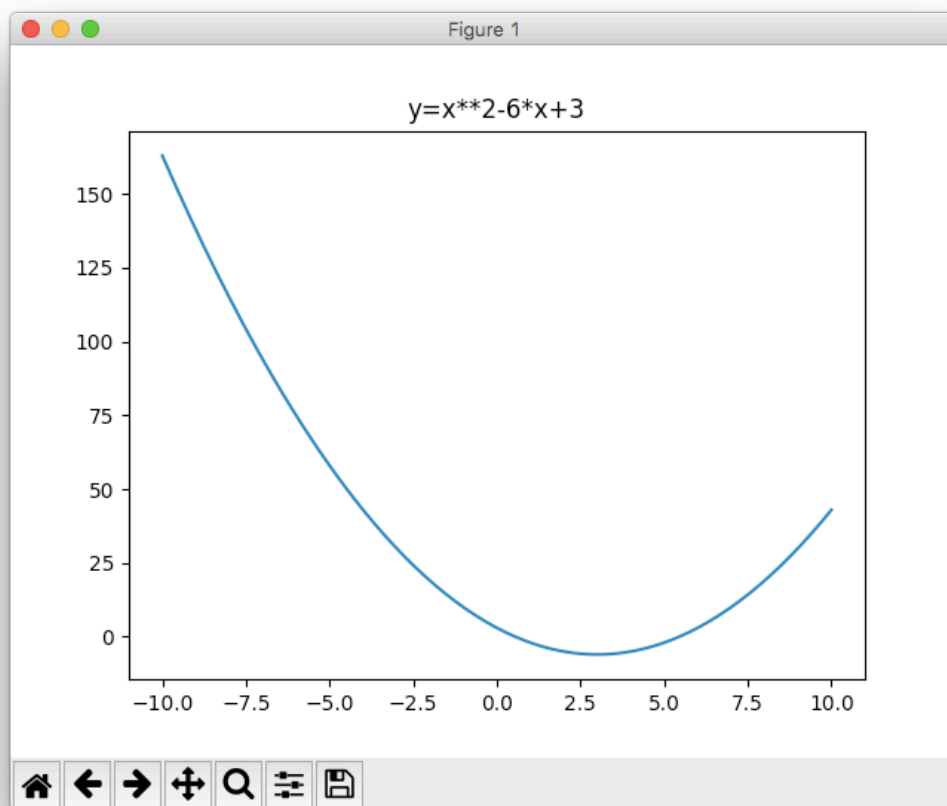
W pierwszej kolejności wywoływana jest funkcja *getFunction()*, która wczytuje z klawiatury wzór funkcji, której wykres należy narysować. Wpisany z klawiatury wzór (string) następnie zamieniany jest przy pomocy wyrażen *lambda* i *eval()* w obiekt klasy *function*, który zwracany jest wraz z wzorem funkcji (string - do wykorzystania do wyświetlenia wzoru funkcji na wykresie). Następnie przy pomocy funkcji *getRange()* wczytywany jest z klawiatury lewy i prawy kraniec przedziału argumentów funkcji, dla którego należy narysować wykres. Ze względu na możliwość wprowadzenia z klawiatury ciągu znaków niedającego się przekształcić na liczbę zastosowano konstrukcję *try - except* w nieskończonej pętli przerywanej po udanym wczytaniu dwóch liczb. Funkcja dodatkowo sprawdza czy lewy kraniec przedziału jest mniejszy od prawego. Jako wynik zwracana jest jednowymiarowa tablica *numpy.array* zawierająca równomiernie rozłożonych 201

(arbitralnie przyjęta rozdzielczość wykresu) liczb z przedziału określonego przez wprowadzony z klawiatury lewy i prawy krańce przedziału rysowania.

Następnie wywoływana jest funkcja `matplotlib.pyplot.plot()`, która przygotowuje wykres na podstawie przygotowanych za pomocą wyżej opisanych funkcji argumentów. Ze względu na zastosowanie bibliotek `matplotlib` do rysowania wykresu oraz `numpy.array` do przechowywania danych dla wykresu prawidłowo obsługiwane są sytuacje typu nieoznaczoność funkcji dla niektórych argumentów (np. dzielenie przez 0). W takim wypadku wartość funkcji w tablicy `numpy` oznaczana jest jako „inf” i punkt pomijany jest przy rysowaniu wykresu (widoczne na drugim z poniższych przykładów). Wzór funkcji ustawiany jest jako tytuł wykresu przy pomocy funkcji `matplotlib.pyplot.title()`. Następnie całość wyświetlana jest przy pomocy `matplotlib.pyplot.show()`

## Wynik:

Przykładowy wynik dla funkcji  $y = x^2 - 6x + 3$



Przykładowy wynik dla funkcji  $y = 1/x - 2x^2$  (nieciągła dziedzina funkcji - nieznaczona dla  $x=0$ )

