Домашнє завдання №4

1. Розробити функцію polygon_area(vertices) для визначення площі опуклого многокутника, яка приймає список координат його вершин, що записані в порядку обходу його вершин за годинниковою стрілкою або проти годинникової стрілки і повертає визначене значення. Для врахування похибки обчислень потрібно використати один з наступних способів:

- розробити допоміжну функцію, яка буде повертати True, якщо різниця двох її аргументів менша за третій аргумент. Якщо num1, num2, epsilon аргументи функції то вона повертає abs(num2 - num1) < epsilon

- скористатися функцією math.isclose()

Площа опуклого многокутника визначається за формулою: S = 1/2|(x1*y2 - x2*y1) + (x2y3 - x3y2)... + (xny1 - x1yn)|

2. Розробити функцію polynomial_eval(coefficients, value) для обчислення значення многочлена. Многочлен описується як список коефіцієнтів (coefficients) його членів а значення x (value) як float. Наприклад, многочлен 2x^3 + 3x^2 + 4 буде описаний як список [2,3,0,4].

3. Розробити функцію polynomials_multiply(polynom1, polynom2) для обчислення добутку двох многочленів, які задані списками коефіцієнтів їх членів.

4. Розробити функцію pattern_number(sequence), яка приймає послідовність (sequence) і повертає кортеж (шаблон, кількість повторів), де кількість повторів це ціле значення >= 2. Шаблон це елементи послідовності, що повторюються. Список не просто містить послідовності, що повторюються а повністю складається з таких послідовностей. Функція повинна повертати None, якщо знайти шаблони, що повторюються не вдалося.

5. Розробити функцію one_swap_sorting(sequence), яка приймає список і повертає True, якщо перестановка двох елементів списку перетворить його у відсортований список і False в іншому випадку. Потрібно врахувати випадок коли список є пустим.

6. Розробити функцію numbers_ulam(n) для пошуку n чисел Улама. Функція повинна повертати список з n чисел Улама.

Число Улама це елемент послідовності цілих чисел. Ці числа названі іменем математика Улама, який їх описав в 1964. Стандартна послідовнічть чисел Улама це числа 1 та 2. Якщо позначити їх через U1 = 1 та U2 = 2, то наступні числа послідовності, при n > 2, визначаються як найменше ціле число, яке:

є сумою двох різних чисел, які вже є в послідовності;

це число можна знайти тільки одним способом;

це число більше за останній член послодовності.

Наприклад,

U(1) = 1

U(2) = 2

Тоді, 1 + 2 = 3 і це число відповідає всім вимогам

Отже, U(3)=3

Нова послідовність, U={1, 2, 3}

1 + 2 = 3 (3 вже входить у послідовність)

1 + 3 = 4 (найменше і знайдене тільки одним способом)

2 + 3 = 5

Отже, U(4)=4

Нова послідовність, U={1, 2, 3, 4}

1 + 2 = 3

1 + 3 = 4

1 + 4 = 5

2 + 3 = 5 (найменше але знайдене двома способами)

2 + 4 = 6 (найменше і знайдене тільки одним способом)

3 + 4 = 7

Отже, U(5)=6

...

Послідовність перших 54 чисел Улама є наступною:

1, 2, 3, 4, 6, 8, 11, 13, 16, 18, 26, 28, 36, 38, 47, 48, 53, 57, 62, 69, 72, 77, 82, 87, 97, 99, 102, 106, 114, 126, 131, 138, 145, 148, 155, 175, 177, 180, 182, 189, 197, 206, 209, 219, 221, 236, 238, 241, 243, 253, 258, 260, 273, 282, ...

7. Розробити функцію happy_number(n) для перевірки чи число n щасливе. Функція повинна приймати число і повертати True, якщо число щасливе і False в іншому випадку. Натуральне число називається щасливим числом, якщо послідовність, яка починається з цього числа, і кожен наступний член якої є сумою квадратів цифр попереднього, містить член рівний одиниці. Наприклад, число 32 є щасливим бо якщо побудувати послідовність a(1) = 32, a(2) = 3^2 + 2^2 = 13, a(3) = 1^2 + 3^2 = 10, a(4) = 1^2 + 0^2 = 1 то в цій послідовності є число 1.

Функція повинна працювати з цілими числами як в десятковій так і в двійковій формі.

8. Розробити функцію sum_of_divisors(n, lst), яка приймає число n та список чисел lst та повертає суму всіх непарних чисел зі списку, які діляться на n.

9. Розробити функцію turn_over(n, lst), яка приймає число n та список lst та повертає цей список але n елементів списку обернені. Потрібно контролювати щоб число n не було більшим за кількість елементів у списку.

10. Розробити модуль calculator.py який дозволяє зчитати з клавіатури два числа та операцію (a - додавання, m - множення, s - сума квадратів), яку з ними потрібно виконати, та виводить на екран результат цієї операції. Для цього потрібно створити 3 функції (addition, multiplication, sum_of_squares), які виконуватимуть відповідні дії та функцію calculator(num1, num2, operation), яка буде забезпечувати виконання обчислень та дій по введенню/виведенню даних. Нижче наведено приклади роботи програми.

Приклад 1:

3 <- Ввід першого числа

4 <- Ввід другого числа

a <- Ввід операції

7 <- Вивід суми

Приклад 2:

3 <- Ввід першого числа

4 <- Ввід другого числа

M <- Ввід операції (операції повинні не бути чутливими до регістру)

12 <- Вивід добутку

Якщо введений символ це не символ операції то функція повинна вивести Error на екран.

11. У Google Перекладачі є функція розпізнавання мови. Спробуєте реалізувати одну з можливих версій цього алгоритму для автоматичного розпізнавання декількох мов.

Розробіть модуль text_analysis.py який забезпечить зчитування тексту з клавіатури і визначатиме мову (англійська, німецька, шведська та французька), якою цей текст написаний. Вважаємо, що для написання тексту використовуються лише 26 літер латинської абетки, пробіл та розділові знаки.

В модулі потрібно реалізуваим наступні функції:

get_number_of_letter(text), що обчислює та повертає кількість літер латинської абетки в text;

compute_absolute_frequency(text), що обчислює кількість входжень кожної з літери у текст та повертає список з цими даними (вважаємо, що елементові списку під номером 0 відповідає літера 'a', а елементові списку під номером 25 відповідає літера 'z'). Не потрібно забувати про регістр літер.

compute_relative_frequency(text), що обчислює відносну частоту входжень кожної з літер у текст та повертає список з цими даними (вважаємо, що елементові списку під номером 0 відповідає літера 'a', а елементові списку під номером 25 відповідає літера 'z'). Не потрібно забувати про регістр. При розробці цієї функції можна використовувати уже написані до цього функції get_number_of_letter та compute_relative_frequency.

compare_text_frequencies(lst1, lst2), що порівнює відносні частоти двох текстів. Для цього функція знаходить квадрат різниці елементів списку. Наприклад, якщо у нас є списки lst1 = [0.1, 0.2, 0.3, 0.4, ...] та lst2 = [0.6, 0.7, 0.8, 0.9, ...], то програма повинна повернути (0.1−0.6)^2+(0.2−0.7)^2+(0.3−0.8)^2+(0.4−0.9)^2+⋯(0.1−0.6)^2+(0.2−0.7)^2+(0.3−0.8)^2+(0.4−0.9)^2+⋯.

Для перевірки працездатності модуля:

- збережіть у рядках eng, ger, swe, fre приклади текстів з файлів doyle.txt, goethe.txt, lagerlof.txt, verne.txt.

- для кожного з рядків визначте відносну частоту (compute_relative_frequency)

- визначте відносну частоту для введеного тексту невідомою мовою (compute_relative_frequency)

- для кожної з пар (текст відомою мовою, текст невідомою мовою) визначте частоти текстів (compare_text_frequencies)

- порівняйте отримані значення для різних пар. Для якої мови цей результат буде найменшим, тією мовою текст ймовірно написано.

_____

Резільтати виконання завдань потрібно подати до завершення граничного терміну у вигляді zip архіву з назвою Name_Surname.zip, який буде містити три модулі. Перший модуль повинен містити функції для вирішення перших дев'яти завдань а два інших модуля повинні містити рішення для десятого і одинадцятого завдань. Назви модулів повинні бути наступні hw4.py, text_analysis.py, calculator.py. Всі функції повинні мати коректні назви та містити вичерпну документацію.