­­ИГИ

Лабораторная работа №3 python, вариант 13

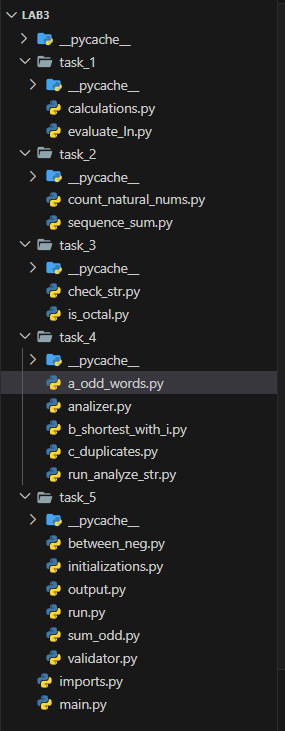
Гр. 253505

Малашкевич М.И

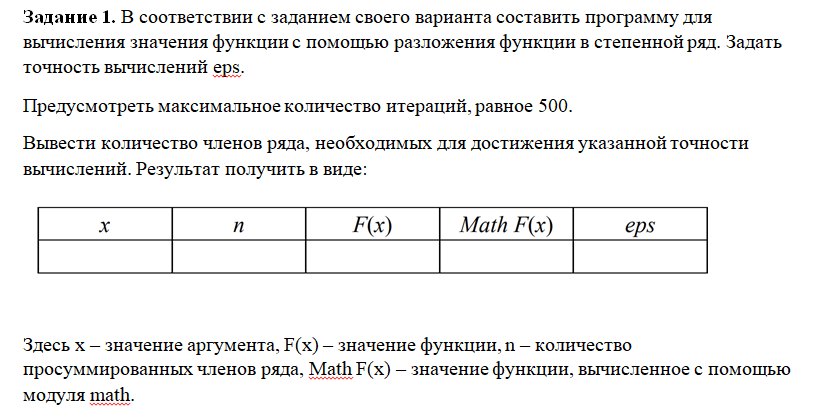
**Тема**: Стандартные типы данных, коллекции, функции, модули.

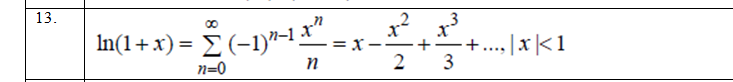
**Цель**: освоить базовый синтаксис языка Python, приобрести навыки работы со стандартными типами данных, коллекциями, функциями, модулями и закрепить их на примере разработки интерактивных приложений.

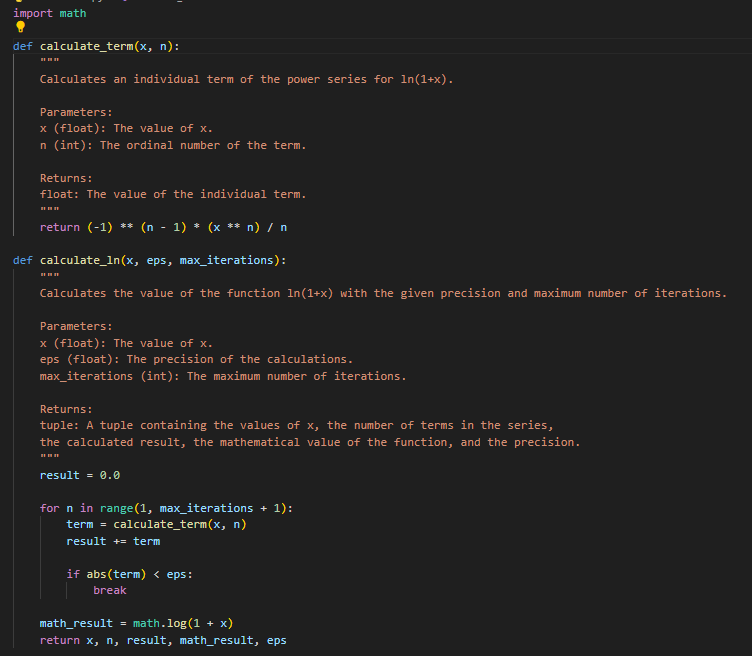
Структура проекта:

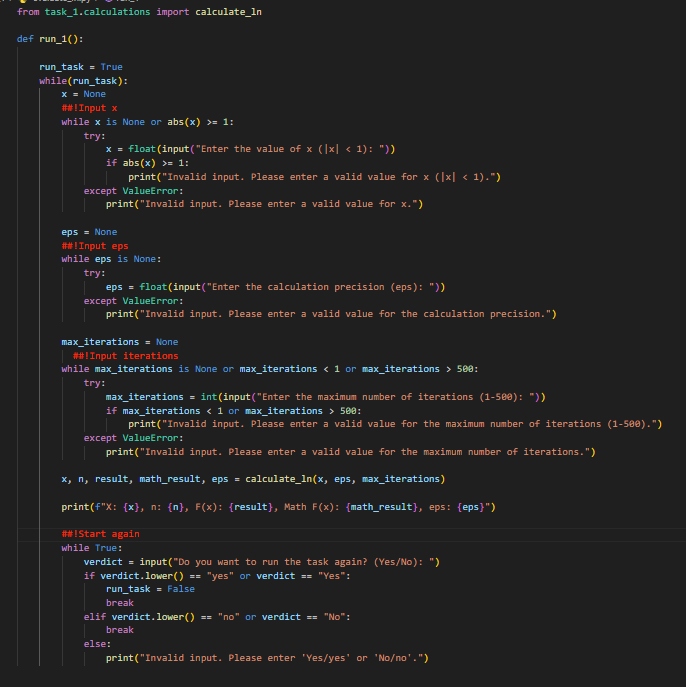


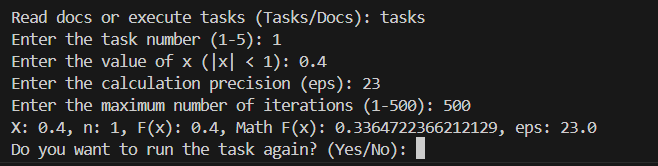
Задание 1:





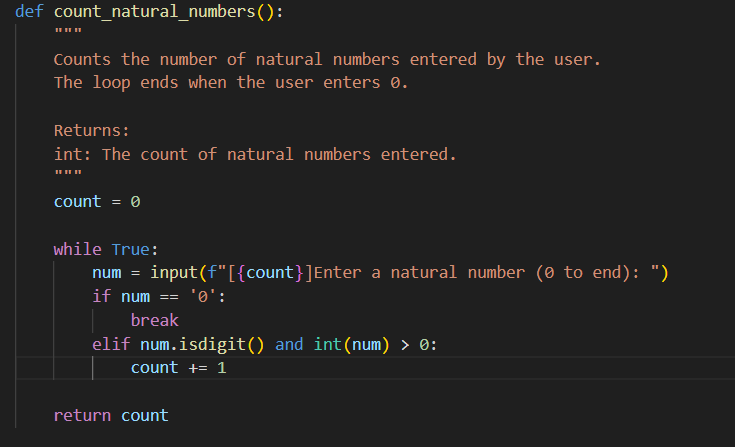


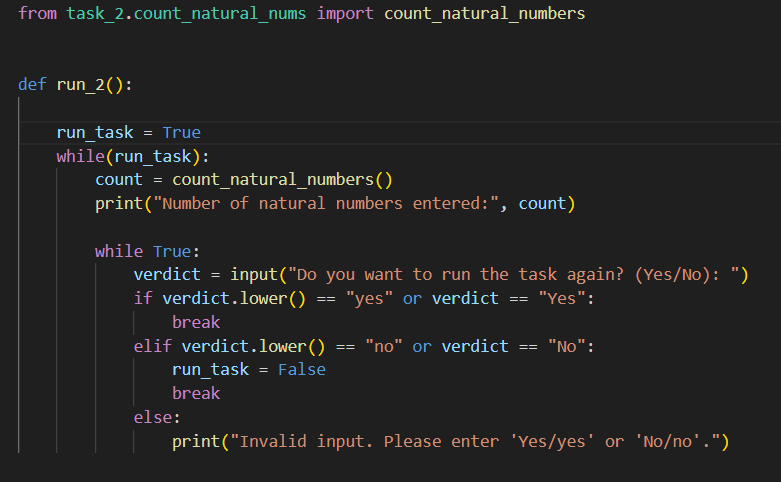


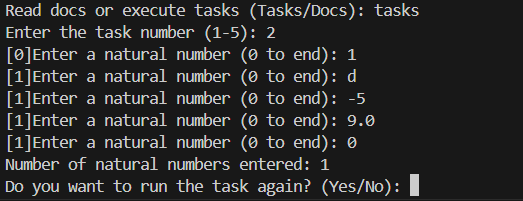
****

**Задание 2.** В соответствии с заданием своего варианта составить программу для нахождения суммы последовательности чисел.



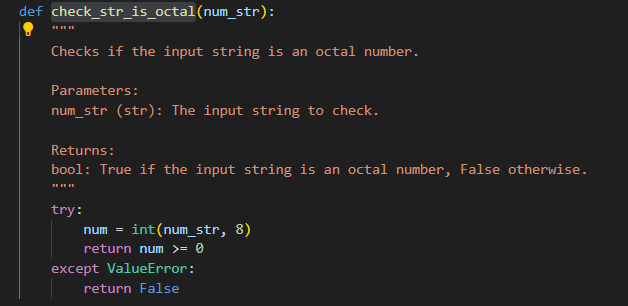


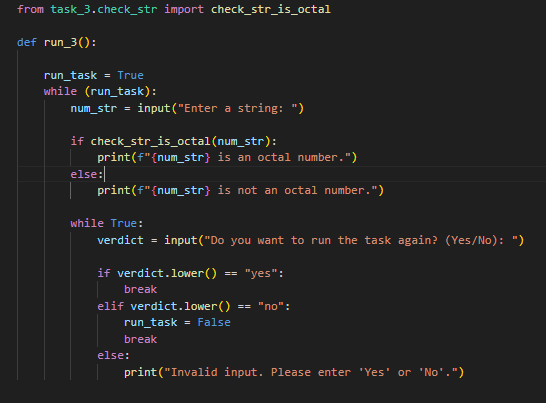


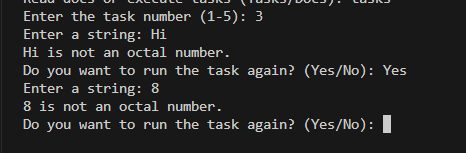


**Задание 3.** **Не использовать регулярные выражения**. В соответствии с заданием своего варианта составить программу для анализа текста, вводимого с клавиатуры.





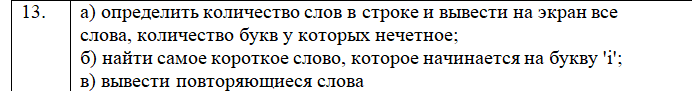


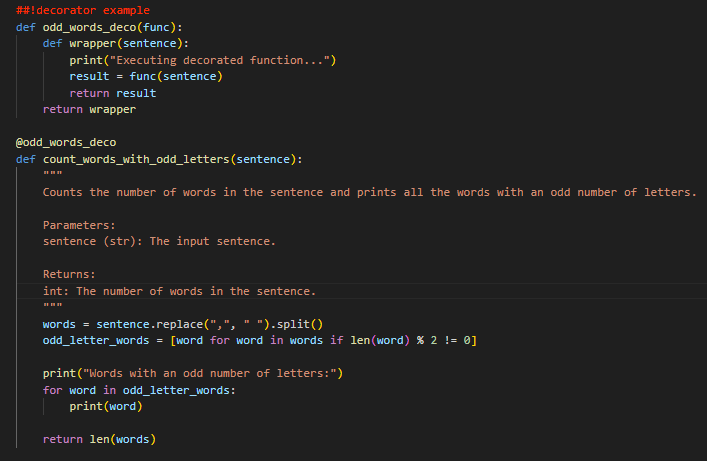


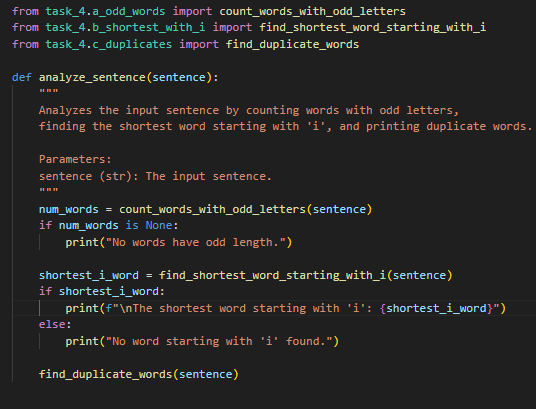
**Задание 4. Не использовать регулярные выражения**. Дана строка текста, в которой слова разделены пробелами и запятыми. В соответствии с заданием своего варианта составьте программу для анализа строки, инициализированной в коде программы:

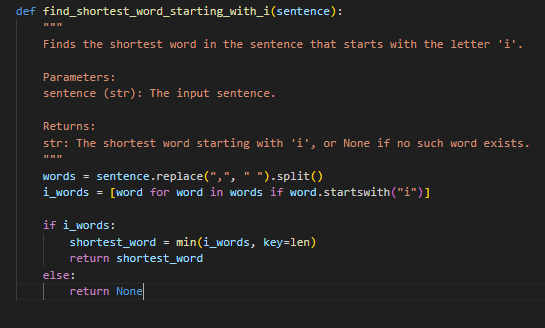
«So she was considering in her own mind, as well as she could, for the hot day made her feel very sleepy and stupid, whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.»

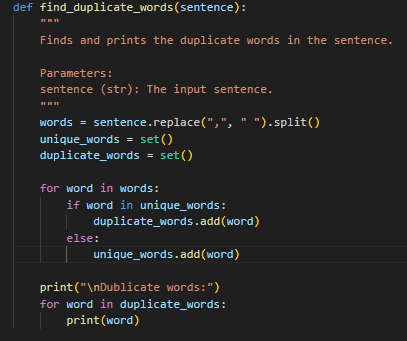
Если не оговорено иное, то регистр букв при решении задачи не имеет значения.

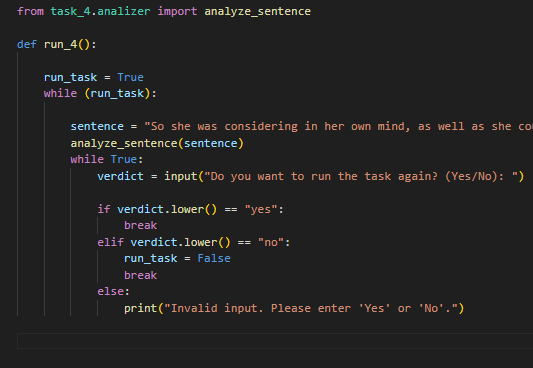


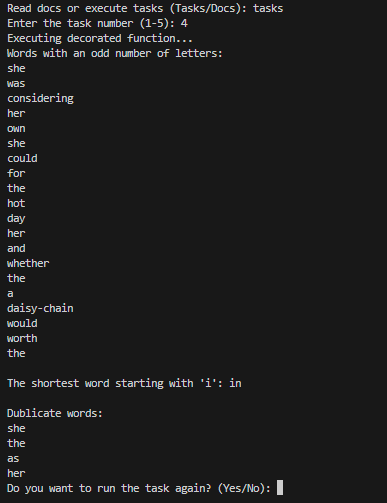












**Задание 5.** В соответствии с заданием своего варианта составить программу для обработки вещественных списков. Программа должна содержать следующие базовые функции:

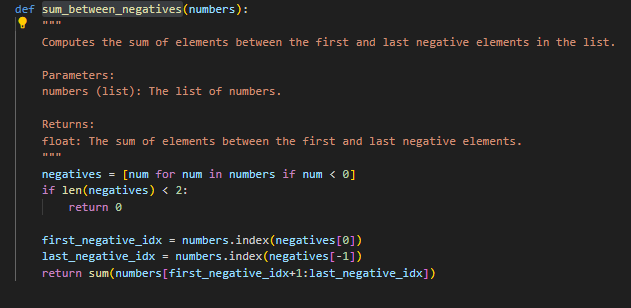
1) ввод элементов списка пользователем;

2) проверка корректности вводимых данных;

3) реализация основного задания с выводом результатов;

4) вывод списка на экран.





##!Generator block

def simple\_generator(left\_lim, right\_lim):

    i = left\_lim

    while i <= right\_lim:

        yield i

        i += 1

def init\_from\_generator(size):

    """

    Initializes a list of numbers using a generator function.

    Parameters:

    size (int): The size of the list.

    generator\_func (function): The generator function.

    Returns:

    list: The initialized list of numbers.

    """

    numbers = []

    for x in simple\_generator(-size, size):

        numbers.append(x)

    return numbers

##!User input

def init\_from\_user(size, input\_validator):

    """

    Initializes a list of numbers using user input.

    Parameters:

    size (int): The size of the list.

    input\_validator (function): The input validator function.

    Returns:

    list: The initialized list of numbers.

    """

    numbers = []

    print("Enter the numbers:")

    for i in range(size):

        valid\_input = False

        while not valid\_input:

            num = input(f"Number {i+1}: ")

            if input\_validator(num):

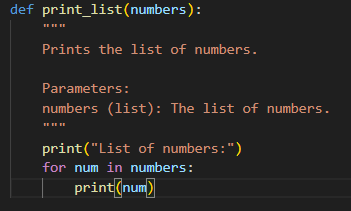
                numbers.append(float(num))

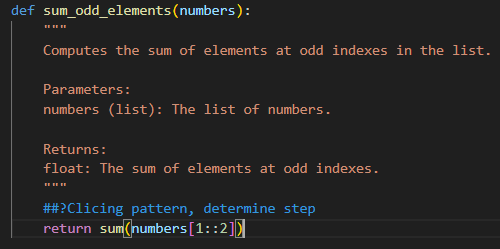
                valid\_input = True

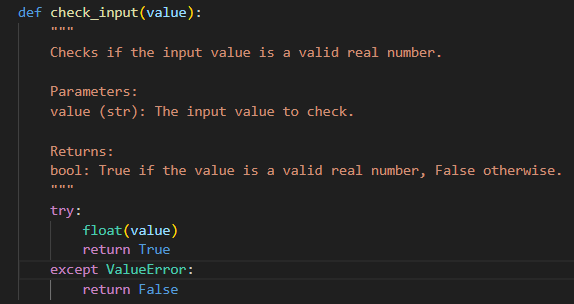
            else:

                print("Invalid input. Please enter a valid real number.")

    return numbers







def run\_5():

    run\_task = True

    while(run\_task):

        ##!Choose the way

        print("Choose an input method:")

        print("1. Manual input")

        print("2. Generator input")

        choice = input("Enter your choice (1 or 2): ")

        if choice == '1':

            size = input("Enter the size of the list: ")

            if not size.isdigit():

                print("Invalid input. Please enter a valid integer.")

                continue

            size = int(size)

            numbers = init\_from\_user(size, check\_input)

        elif choice == '2':

            size = input("Enter the size of the list: ")

            if not size.isdigit():

                print("Invalid input. Please enter a valid integer.")

                continue

            size = int(size)

            numbers = init\_from\_generator(size)

        else:

            print("Invalid choice. Please enter 1 or 2.")

            continue

        if not numbers:

            print("No valid numbers entered. Please try again.")

            continue

        ##!Calling tasks components

        print\_list(numbers)

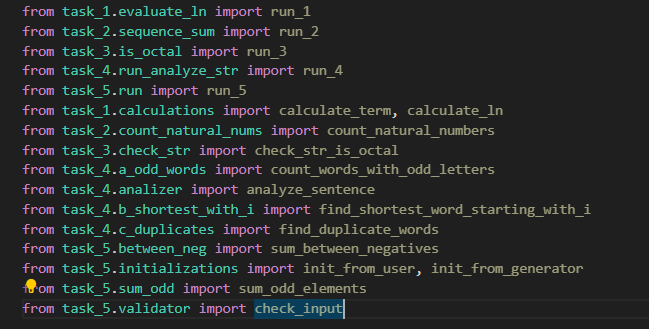
        sum\_odd = sum\_odd\_elements(numbers)

        sum\_between = sum\_between\_negatives(numbers)

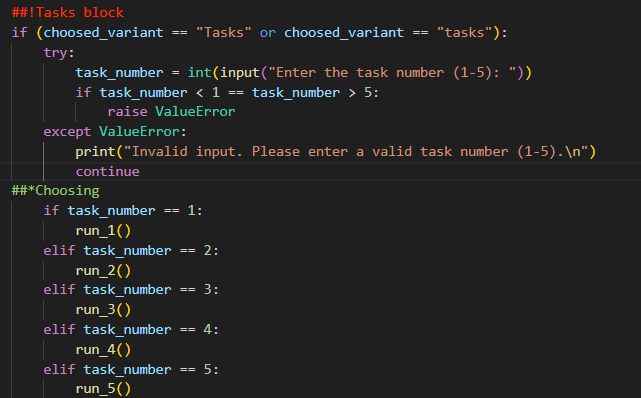
        print("Sum of elements at odd indexes:", sum\_odd)

        print("Sum of elements between the first and last negative elements:", sum\_between)

Все импорты:



Запуск заданий:



Запуск документации:

