Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Избранные главы информатики

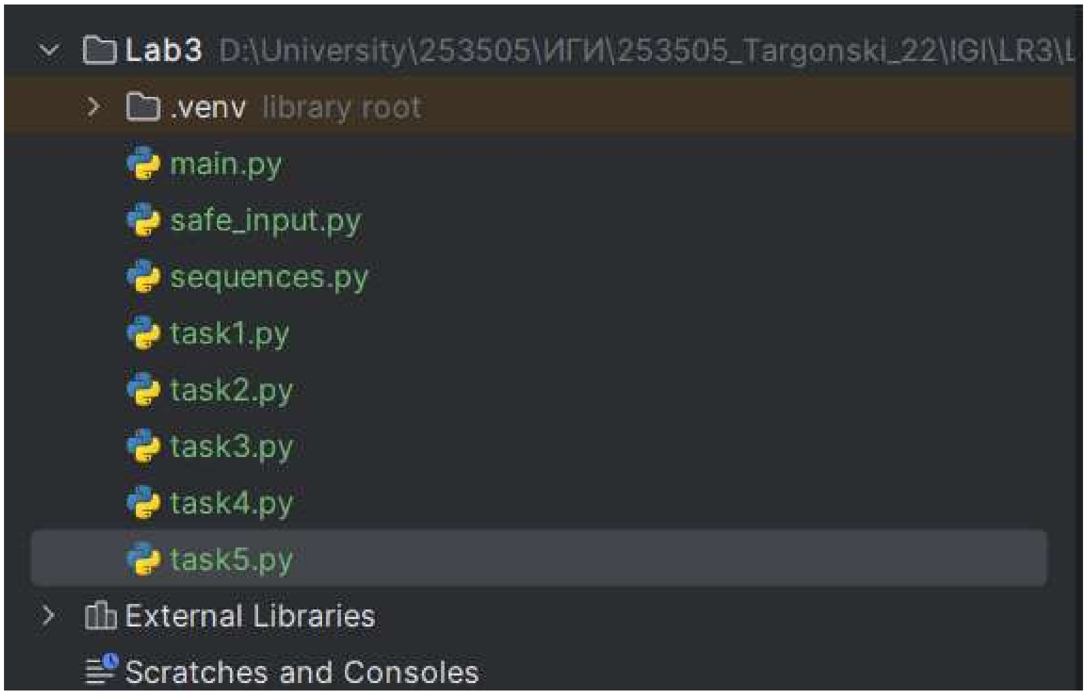
ОТЧЁТ
к лабораторной работе №1

Выполнил: студент группы 253505
Таргонский Дмитрий Андреевич

Минск 2022

Вариант №22

Структура проетка



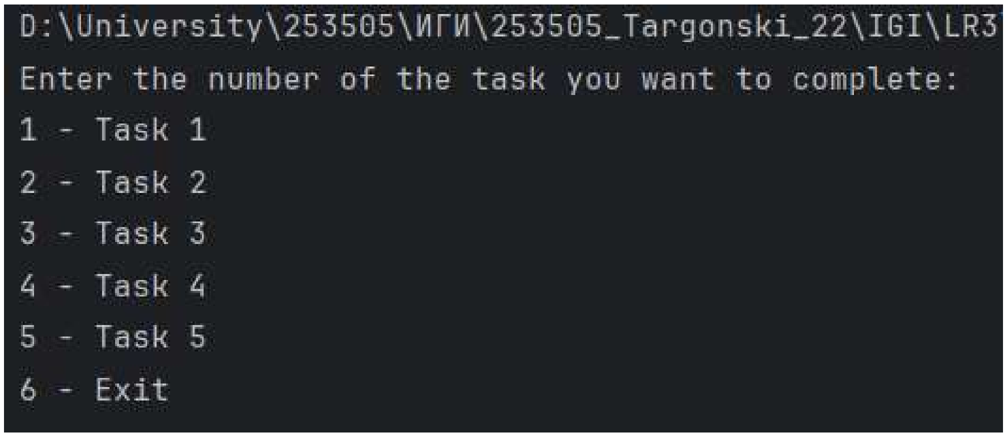
Lab3 D:\University\253505\ИГИ\253505_Targonski_22\IGI\LR3\L
.venv library root
main.py
safe_input.py
sequences.py
task1.py
task2.py
task3.py
task4.py
task5.py
External Libraries
Scratches and Consoles

Главный (исполняемый) файл

```
# This is the main file for running a series of Python programs as
part of a lab assignment.
# Each task is a separate function imported from its respective
module.
# The user can choose which task to run, and the program will
execute it and display the results.
# Lab Work Number: №3
# Program Version: 3.0
# Developer: Tarhonski Dzmitry
# Development Date: 28.03.2024
import os

# Import the module where the calculate_series function is defined
from task1 import task1
from task2 import task2
from task3 import task3
from task4 import task4
from task5 import task5

def main():
    """
    The main function of the program. It prompts the user for input,
    calls the calculate_series function, and prints the results.
    """
    print("Enter the number of the task you want to complete:")
    print("1 - Task 1")
    print("2 - Task 2")
    print("3 - Task 3")
    print("4 - Task 4")
    print("5 - Task 5")
    print("6 - Exit")
    task = input()
    exitTask=False
    exitLab=False
    while exitLab==False:
            match task:
                case "1":
                    while exitTask==False:
                        try:
                            print("Task №1 selected")
                            task1()
                            # Ask the user if they want to run the
program again
                            run_again = input("Do you want to run
task №1 again? (yes/no): ")
                            if run_again.lower() == 'yes':
                                print("Restarting the task")
                            if run_again.lower() == 'no':
                                print("Exiting task #1")
                                exitTask=True
                                main()
                            else:
                                while run_again.lower() != 'yes' and
run_again.lower() != 'no':
                                    print("Input error! Repeat
again:")
                                    run_again = input()
                        except ValueError:
                            print("An incorrect value has been
entered. Please enter a numeric value.")
                        except Exception as e:
                            print(f"Error occurred: {e}")
                case "2":
                    while exitTask == False:
                        try:
                            print("Task №2 selected")
                            task2()
                            # Ask the user if they want to run the
program again
                            run_again = input("Do you want to run
task №2 again? (yes/no): ")
```

```
                            if run_again.lower() == 'yes':
                                print("Restarting the task")
                            if run_again.lower() == 'no':
                                print("Exiting task №2")
                                exitTask = True
                                main()
                            else:
                                while run_again.lower() != 'yes' and
run_again.lower() != 'no':
                                    print("Input error! Repeat
again:")
                                    run_again = input()
                        except ValueError:
                            print("An incorrect value has been
entered. Please enter a numeric value.")
                        except Exception as e:
                            print(f"Error occurred: {e}")
                case "3":
                    while exitTask == False:
                        try:
                            print("Task №3 selected")
                            task3()
                            # Ask the user if they want to run the
program again
                            run_again = input("Do you want to run
task №1 again? (yes/no): ")
                            if run_again.lower() == 'yes':
                                print("Restarting the task")
                            if run_again.lower() == 'no':
                                print("Exiting task №3")
                                exitTask = True
                                main()
                            else:
                                while run_again.lower() != 'yes' and
run_again.lower() != 'no':
                                    print("Input error! Repeat
again:")
                                    run_again = input()
                        except ValueError:
                            print("An incorrect value has been
entered. Please enter a numeric value.")
                        except Exception as e:
                            print(f"Error occurred: {e}")
                case "4":
                    while exitTask == False:
                        try:
                            print("Task №4 selected")
                            task4()
                            # Ask the user if they want to run the
program again
                            run_again = input("Do you want to run
task №1 again? (yes/no): ")
                            if run_again.lower() == 'yes':
                                print("Restarting the task")
                            if run_again.lower() == 'no':
                                print("Exiting task №4")
                                exitTask = True
                                main()
                            else:
                                while run_again.lower() != 'yes' and
run_again.lower() != 'no':
                                    print("Input error! Repeat
again:")
                                    run_again = input()
                        except ValueError:
                            print("An incorrect value has been
entered. Please enter a numeric value.")
                        except Exception as e:
                            print(f"Error occurred: {e}")
                case "5":
                    while exitTask == False:
                        try:
                            print("Task №5 selected")
                            task5()
                            # Ask the user if they want to run the
program again
                            run_again = input("Do you want to run
task №1 again? (yes/no): ")
                            if run_again.lower() == 'yes':
                                print("Restarting the task")
                            if run_again.lower() == 'no':
                                print("Exiting task №5  ")
                                exitTask = True
                                main()
                            else:
                                while run_again.lower() != 'yes' and
run_again.lower() != 'no':
                                    print("Input error! Repeat
again:")
                                    run_again = input()
                        except ValueError:
                            print("An incorrect value has been
entered. Please enter a numeric value.")
                        except Exception as e:
                            print(f"Error occurred: {e}")
                case "6":
                    print("Program shutdown.")
                    exitLab = True
                case _:
                    print("Invalid value entered, try again!")
                    main()

if __name__ == "__main__":
    main()
```

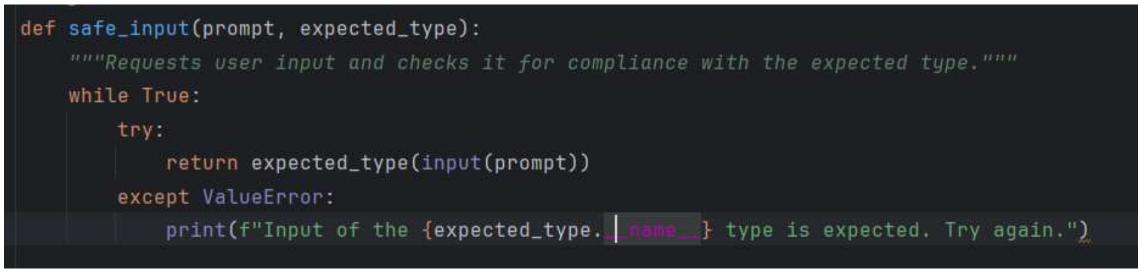
Результат запуска



```
D:\University\253505\ИГИ\253505_Targonski_22\IGI\LR3
Enter the number of the task you want to complete:
1 - Task 1
2 - Task 2
3 - Task 3
4 - Task 4
5 - Task 5
6 - Exit
```

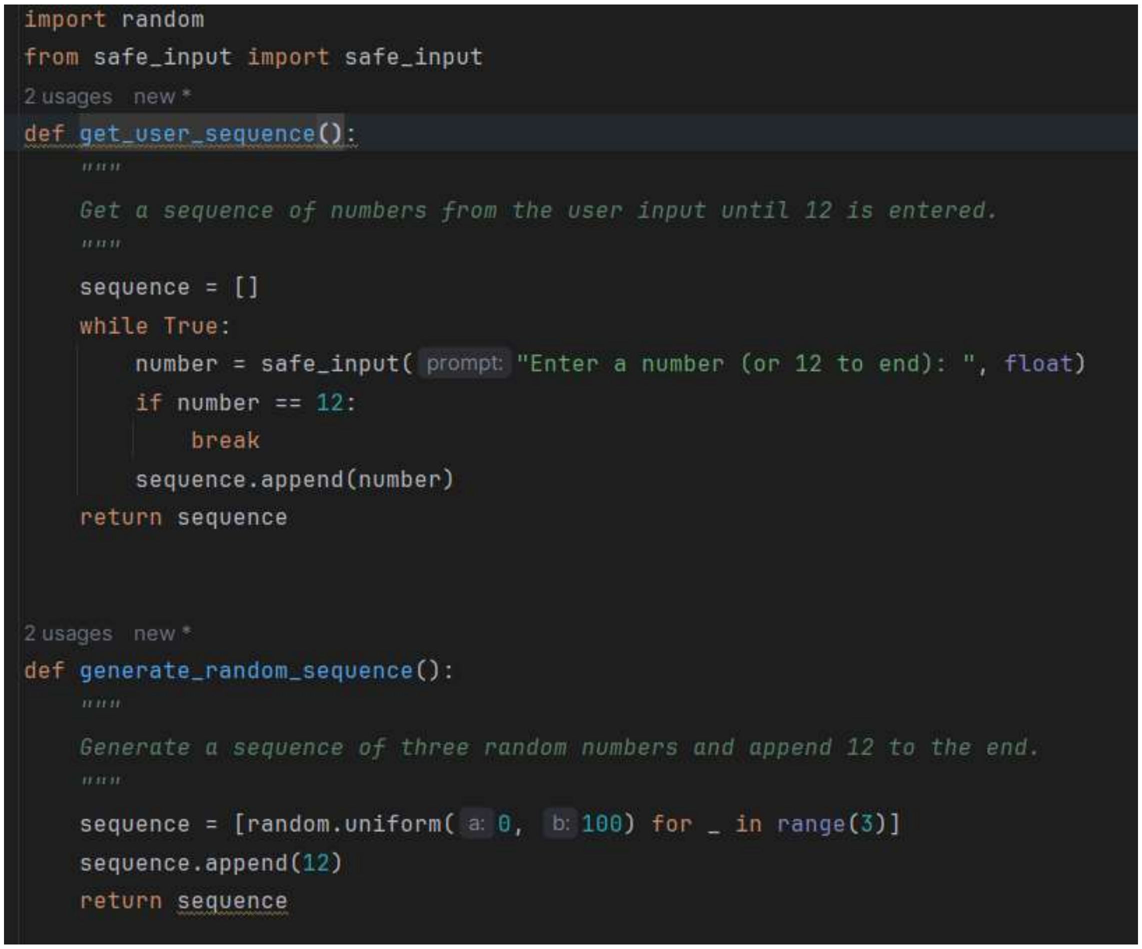
Вспомогательные функции

safe_input



```python
def safe_input(prompt, expected_type):
    """Requests user input and checks it for compliance with the expected type."""
    while True:
        try:
            return expected_type(input(prompt))
        except ValueError:
            print(f"Input of the {expected_type.__name__} type is expected. Try again.")
```

senquences



```
import random
from safe_input import safe_input
2 usages  new *
def get_user_sequence():
    """
    Get a sequence of numbers from the user input until 12 is entered.
    """
    sequence = []
    while True:
        number = safe_input( prompt: "Enter a number (or 12 to end): ", float)
        if number == 12:
            break
        sequence.append(number)
    return sequence


2 usages  new *
def generate_random_sequence():
    """
    Generate a sequence of three random numbers and append 12 to the end.
    """
    sequence = [random.uniform( a: 0,  b: 100) for _ in range(3)]
    sequence.append(12)
    return sequence
```

Задание 1. В соответствии с заданием своего варианта составить программу для вычисления значения функции с помощью разложения функции в степенной ряд. Задать точность вычислений eps.

Предусмотреть максимальное количество итераций, равное 500.

Вывести количество членов ряда, необходимых для достижения указанной точности вычислений. Результат получить в виде:

x	*n*	*F*(*x*)	*Math F*(*x*)	*eps*

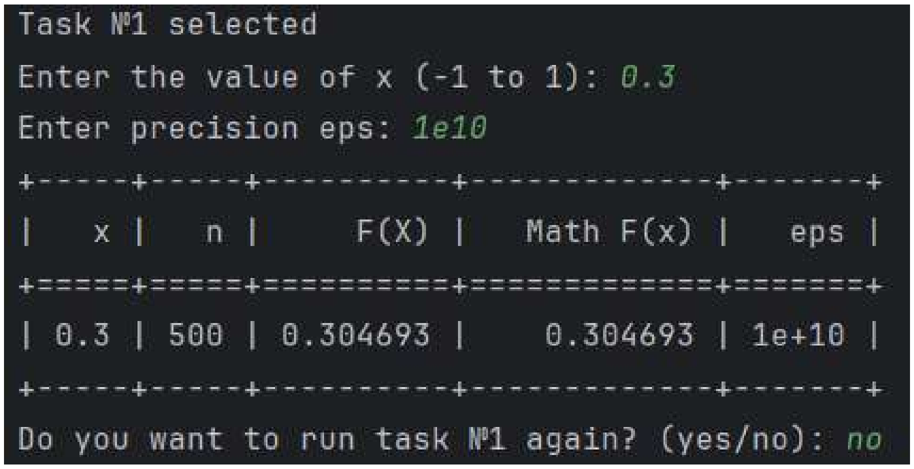
Здесь x – значение аргумента, F(x) – значение функции, n – количество просуммированных членов ряда, Math F(x) – значение функции, вычисленное с помощью модуля math.

$$\arcsin x = \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1} = x + \frac{x^3}{6} + \frac{3x^5}{40} + \ldots, |x| < 1$$

Листинг кода task1()

```
import math
from tabulate import tabulate
from safe_input import safe_input
# This function calculates the value of a function using a power
series expansion.
# It takes an argument 'x' and a precision 'eps', and returns the
function value
# and the number of terms summed to reach the specified precision.
# The function limits the number of iterations to 500.

def task1():
    def calculate_series(x, eps):
        """
        Calculate the value of the function using power series
expansion.

        Parameters:
        x (float): The argument value for which the function is
calculated.
        eps (float): The precision of the calculations.

        Returns:
        float: The value of the function.
        int: The number of terms summed in the series.
        """
        n = 0  # Counter for the number of terms summed in the
series
        term = x  # The first term of the series
        sum_series = term  # The sum of the series

        # Loop to sum the terms of the series until the specified
precision is reached
        # or the maximum number of iterations is exceeded
        while abs(term) < eps and n < 500:
            n += 1  # Increment the term counter
            term = (math.factorial(2 * n) / (2 ** (2 * n) *
(math.factorial(n) ** 2))) * (x ** (2 * n + 1) / (2 * n + 1))
# Calculate the next term
            sum_series += term  # Add the term to the sum

        return sum_series, n
    # Example of using the function
    while True:
        x =  safe_input("Enter the value of x (-1 to 1): ",float)  # 
Prompt the user to enter the value of 'x'
        if -1 <= x <= 1:
            break
        else:
            print("The value of x must be in the range from -1 to 1 
inclusive.")
    while True:
        eps = float(input("Enter precision eps: "))  # Prompt the 
user to enter the precision 'eps'
        if eps <=0:
            print("Input error. The accuracy must be greater than 
0!")
        else:
            break
    series_value, terms_count = calculate_series(x, eps)  # Call the 
function with user inputs

    # Print the results
    resultData=[
        [x,terms_count,series_value,math.asin(x),eps]
    ]
    headers=["x","n","F(X)","Math F(x)","eps"]
    print(tabulate(resultData, headers=headers, tablefmt="grid"))
```

Результат запуска



```
Task №1 selected
Enter the value of x (-1 to 1): 0.3
Enter precision eps: 1e10
+-----+-----+----------+-------------+-------+
|   x |   n |     F(X) |   Math F(x) |   eps |
+=====+=====+==========+=============+=======+
| 0.3 | 500 | 0.304693 |    0.304693 | 1e+10 |
+-----+-----+----------+-------------+-------+
Do you want to run task №1 again? (yes/no): no
```

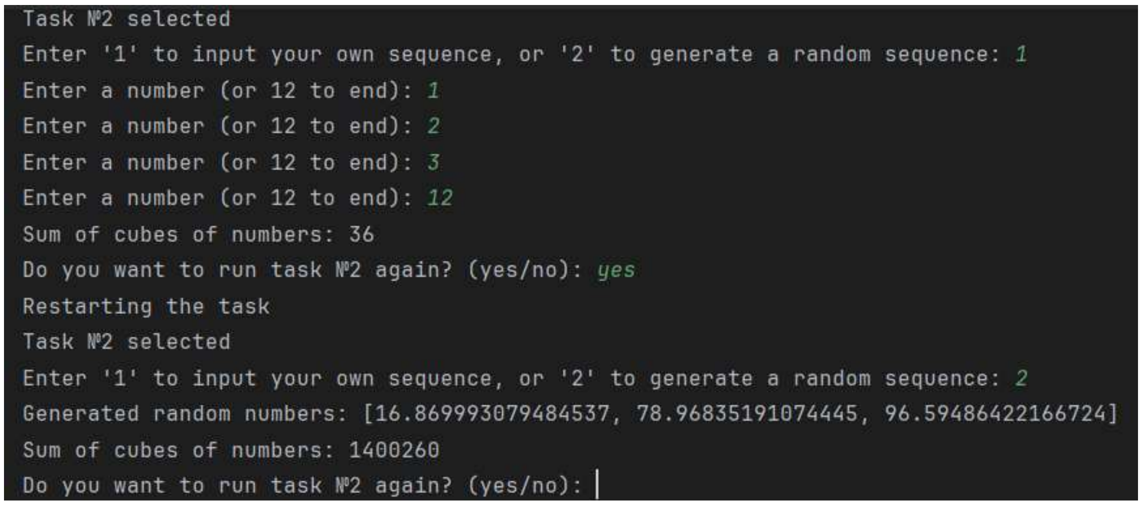
Задание 2. В соответствии с заданием своего варианта составить программу для нахождения суммы последовательности чисел.

Организовать цикл, принимающий числа и суммирующий их кубы. Окончание цикла – ввод числа 12

Листинг кода task2()

```
from safe_input import safe_input

from sequences import get_user_sequence, generate_random_sequence




def task2():

    def sum_cubes(numbers):

        """

        Sum the cubes of numbers in the provided sequence.


        Parameters:

        numbers (list): The sequence of numbers to sum the cubes of.


        Returns:

        int: The total sum of the cubes of the numbers.

        """

        return sum(number ** 3 for number in numbers)


    # Ask the user to choose the method of sequence initialization

    method = safe_input("Enter '1' to input your own sequence, or
'2' to generate a random sequence: ", int)


    if method == 1:
        numbers = get_user_sequence()

    elif method == 2:

        numbers = generate_random_sequence()

        print(f"Generated random numbers: {numbers[:-1]}")

    else:

        print("Invalid input. Exiting the program.")

        return


    # Calculate and print the sum of cubes

    print(f"Sum of cubes of numbers: {int(sum_cubes(numbers))}")
```

Результат запуска



```
Task №2 selected
Enter '1' to input your own sequence, or '2' to generate a random sequence: 1
Enter a number (or 12 to end): 1
Enter a number (or 12 to end): 2
Enter a number (or 12 to end): 3
Enter a number (or 12 to end): 12
Sum of cubes of numbers: 36
Do you want to run task №2 again? (yes/no): yes
Restarting the task
Task №2 selected
Enter '1' to input your own sequence, or '2' to generate a random sequence: 2
Generated random numbers: [16.869993079484537, 78.96835191074445, 96.59486422166724]
Sum of cubes of numbers: 1400260
Do you want to run task №2 again? (yes/no): 
```

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

Определить, является ли введенная с клавиатуры строка двоич-

ным числом

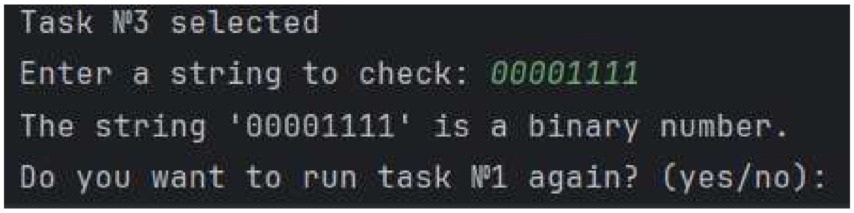
Листинг кода task3()

```
# Main program
```

```
def task3():

    user_input = input("Enter a string to check: ")  # Prompt the
user for a string

    is_binary_number(user_input)  # Call the function with the user
input

def binary_check_decorator(func):

    """

    A decorator that logs the result of checking if a string is a
binary number.

    """

    def wrapper(text):

        result = func(text)  # Call the function being checked

        if result:

            print(f"The string '{text}' is a binary number.")

        else:

            print(f"The string '{text}' is NOT a binary number.")

        return result

    return wrapper


# Applying the decorator to the is_binary_number function

@binary_check_decorator

def is_binary_number(text):

    """

            Checks if the entered string is a binary number.


            This function takes a string as input and analyzes it to
determine

            if it is composed exclusively of the digits 0 and 1,
which would
```

```
            make it a binary number. It iterates through each
character of the

            string and returns False if it finds any character other
than '0' or '1'.


            Parameters:

            text (str): The string to be checked.


            Returns:

            bool: True if the string is a binary number, False
otherwise.

            """

    # The body of the function remains unchanged

    # Iterate over each character in the string

    for char in text:

        # If the character is not '0' or '1', return False

        if char not in ('0', '1', ' '):

            return False

    # If the loop completes without returning False, it's a binary
number

    return True
```

Результат запуска



```
Task №3 selected
Enter a string to check: 00001111
The string '00001111' is a binary number.
Do you want to run task №1 again? (yes/no):
```

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.

а) определить количество строчных букв;

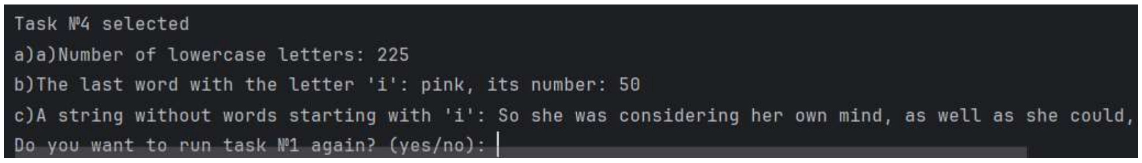
б) найти последнее слово, содержащее букву 'i' и его номер;

в) вывести строку, исключив из нее слова, начинающиеся с 'i'

Листинг кода task4

```
def task4():

    # The original text string

    text = (

        "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."

    )



    # Task a) Determine the number of lowercase letters

    # Count each character in the text that is a lowercase letter

    lowercase_count = sum(1 for char in text if char.islower())

    print(f"a)a)Number of lowercase letters: {lowercase_count}")
    # Task b) Find the last word containing the letter 'i' and its
number

    # Split the text into words and enumerate them starting from 1

    words = text.split()

    last_i_word = None  # Initialize the variable to store the last
word with 'i'

    last_i_index = None  # Initialize the variable to store the
index of the last word with 'i'

    # Iterate through the words and their indices

    for index, word in enumerate(words, start=1):

        # Check if the word contains the letter 'i'

        if 'i' in word.lower():

            last_i_word = word  # Update the last word with 'i'

            last_i_index = index  # Update the index of the last
word with 'i'

    print(f"b)The last word with the letter 'i': {last_i_word}, its
number: {last_i_index}")



    # Task c) Output the string excluding words starting with 'i'

    # Join words that do not start with 'i' into a new string

    filtered_text = ' '.join(word for word in words if not
word.lower().startswith('i'))

    print(f"c)A string without words starting with 'i':
{filtered_text}")
```

Результат запуска



```
Task №4 selected
a)a)Number of lowercase letters: 225
b)The last word with the letter 'i': pink, its number: 50
c)A string without words starting with 'i': So she was considering her own mind, as well as she could,
Do you want to run task №1 again? (yes/no): 
```

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;

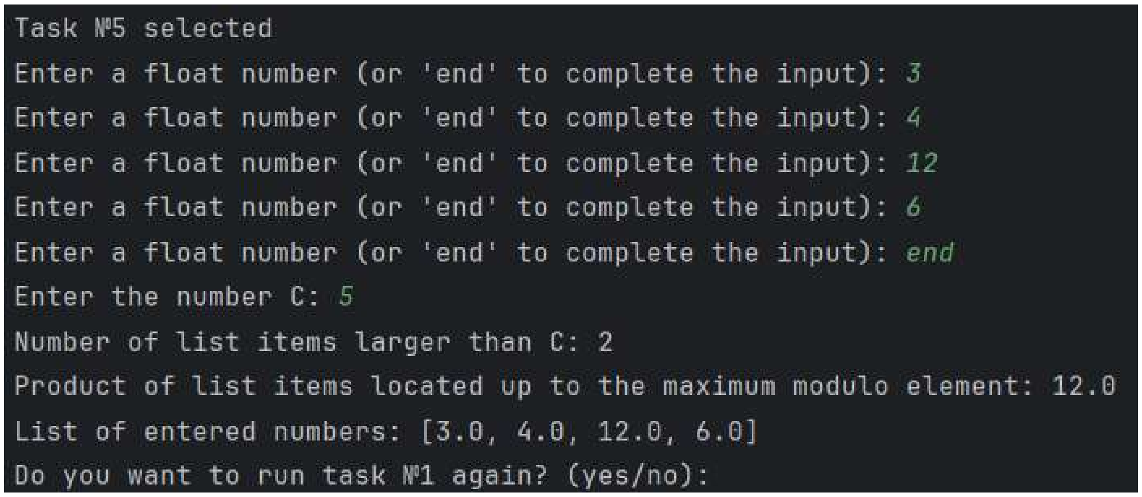
4) вывод списка на экран.

Найти количество элементов списка, больших числа С (параметр С вводится с клавиатуры пользователем) и произведение элементов списка, расположенных до максимального по модулю элемента

Листинг кода task5()

```
from safe_input import safe_input
def task5():
    # The main program
    # User input for the list
    user_float_list = input_float_list()
    # User input for the parameter C
    c = safe_input("Enter the number C: ",float)
    # Display the results
    print(f"Number of list items larger than C: {count_greater_than_c(user_float_list, c)}")
    print(f"Product of list items located up to the maximum modulo element:
{product_before_max(user_float_list)}")
    # Display the list on the screen
    print(f"List of entered numbers: {user_float_list}")

def input_float_list():
    """
    Prompts the user to enter real numbers to create a list.
    Validates the input and returns the list of numbers.
    """
    float_list = []  # Initialize an empty list to store the real numbers
    while True:  # Start an infinite loop for user input
        number = input("Enter a float number (or 'end' to complete the input): ")
        if number == 'end':  # Check if the user wants to end the input
            break
        try:
            float_number = float(number)  # Attempt to convert the input to a real number
            float_list.append(float_number)  # Add the number to the list
        except ValueError:  # Handle the error if the input is not a real number
            print("A non-float number has been entered. Try again.")
    return float_list  # Return the list of real numbers


def count_greater_than_c(float_list, c):
    """
    Counts the number of elements in the list that are greater than the number C.
    """
    return sum(1 for number in float_list if number > c)  # Use a generator expression to count


def product_before_max(float_list):
    """
    Calculates the product of elements in the list located before the maximum absolute value element.
    """
    max_value = max(float_list, key=abs)  # Find the max value by absolute value
    max_index = float_list.index(max_value)  # Get the index of the max value
    if max_index == 0:  # Check if the max value is the first element
        return None
    product = 1  # Initialize the product variable
    for number in float_list[:max_index]:  # Iterate over elements before the max value
        product *= number  # Multiply the elements to get the product
    return product  # Return the product
```

Результат запуска



```
Task №5 selected
Enter a float number (or 'end' to complete the input): 3
Enter a float number (or 'end' to complete the input): 4
Enter a float number (or 'end' to complete the input): 12
Enter a float number (or 'end' to complete the input): 6
Enter a float number (or 'end' to complete the input): end
Enter the number C: 5
Number of list items larger than C: 2
Product of list items located up to the maximum modulo element: 12.0
List of entered numbers: [3.0, 4.0, 12.0, 6.0]
Do you want to run task №1 again? (yes/no):
```