Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина «Избранные главы информатики»

**ОТЧЕТ**

к лабораторной работе №3

на тему:

**«Стандартные типы данных, коллекции, функции, модули»**

БГУИР 6-05-0612-02 01

|  |
| --- |
| Выполнила студентка группы 353504  АНТОНОВА Лидия Сергеевна |
|  |
| (дата, подпись студента) |
| Проверила  ЖВАКИНА Анна Васильевна |
|  |
| (дата, подпись преподавателя) |

Минск 2025

# Цель работы

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

Вариант 1.

# Выполнение работы

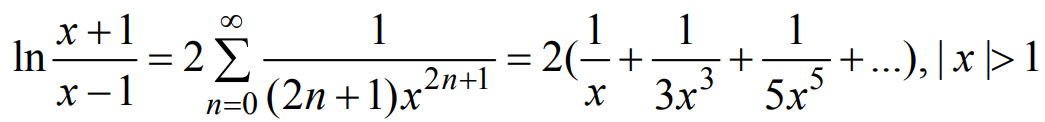
Задание 1. В соответствии с заданием своего варианта составить программу для вычисления значения функции c помощью разложения функции в степенной ряд. Задать точность вычислений eps.

Предусмотреть максимальное количество итераций, равное 500.

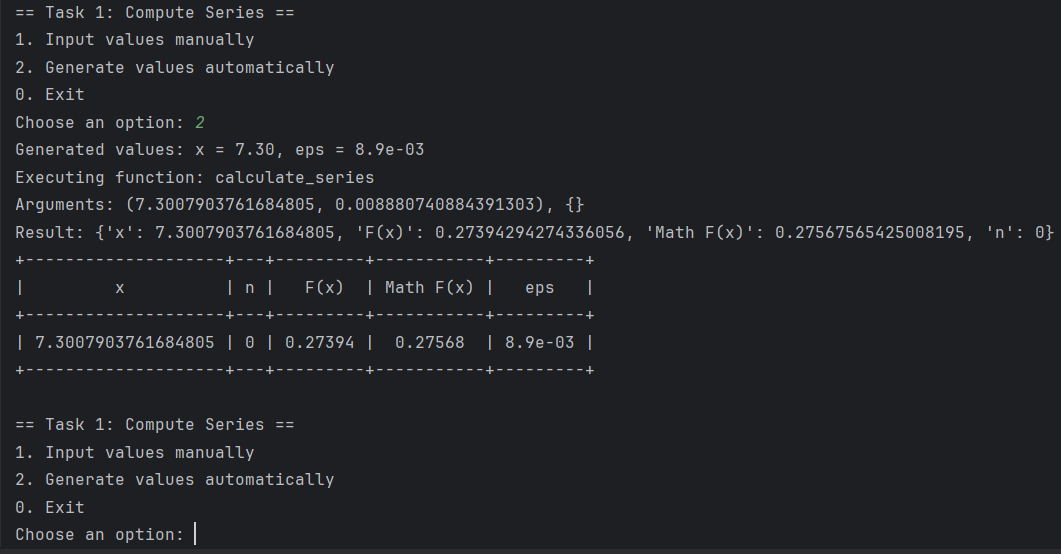
Вывести количество членов ряда, необходимых для достижения указанной точности вычислений. Результат получить в виде:



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



import math  
from Decorator import log\_execution  
from Input import \*  
from prettytable import PrettyTable  
  
@log\_execution  
def calculate\_series(x, eps, max\_iterations=500):  
 *"""  
 Function to compute ln(x+1)/(x-1) approximation using Taylor series expansion by 500 iterations or less and with math funciton  
 Args:  
 x (float): Argument value.  
 eps (float): Desired accuracy.  
 max\_iterations (int): Maximum number of terms.  
 Returns:  
 dict: Result including F(x), Math F(x), n (terms needed).  
 """* if abs(x) <= 1:  
 raise ValueError("x must satisfy |x| > 1")  
  
 n = 0  
 term = 1 / x  
 result = term  
 mathF = math.log((x + 1) / (x - 1))  
 while abs(mathF - result \* 2) > eps and n < max\_iterations:  
 n += 1  
 term = 1 / ((2 \* n + 1) \* (x \*\* (2 \* n + 1)))  
 result += term  
 return {  
 "x": x,  
 "F(x)": 2 \* result,  
 "Math F(x)": mathF,  
 "n": n  
 }  
  
def display\_results(result, eps):  
 *"""  
 Displays the results in a formatted table.  
 Args:  
 result (dict): Result of the series calculation.  
 eps (float): Precision used in calculations.  
 """* table = PrettyTable()  
 table.field\_names = ["x", "n", "F(x)", "Math F(x)", "eps"]  
 table.add\_row([  
 result["x"],  
 result["n"],  
 f"{result['F(x)']:.5f}",  
 f"{result['Math F(x)']:.5f}",  
 f"{eps:.1e}"  
 ])  
 print(table)  
  
def task1():  
 *"""  
 Task 1: compute computing series with input or automatic generation.  
 """* while True:  
 print("\n== Task 1: Compute Series ==")  
 print("1. Input values manually")  
 print("2. Generate values automatically")  
 print("0. Exit")  
 choice = input("Choose an option: ").strip()  
  
 if choice == '1':  
 try:  
 x = input\_float("Enter x, |x| > 1: ", 1.1)  
 eps = input\_float("Enter precision (eps): ", 0)  
 except ValueError as ve:  
 print(f"Invalid input: {ve}")  
 continue  
 elif choice == '2':  
 x = generate\_value(1.1, 10)  
 eps = generate\_value(1e-5, 1e-2)  
 print(f"Generated values: x = {x:.2f}, eps = {eps:.1e}")  
 elif choice == '0':  
 print("Exiting Task 1...")  
 break  
 else:  
 print("Invalid choice. Please choose 1, 2, or 0.")  
 continue  
  
 try:  
 result = calculate\_series(x, eps)  
 display\_results(result, eps)  
 except ValueError as ve:  
 print(f"Invalid input: {ve}")  
 except Exception as e:  
 print(f"An error occurred: {e}")



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

Организовать цикл, который принимает целые числа с клавиатуры и суммирует их квадраты. Окончание цикла – ввод числа 0.

from Decorator import log\_execution  
from Input import \*  
  
@log\_execution  
def sum\_of\_squares(sequence):  
 """  
 Calculates the sum of squares of integers entered by the user until 0 is entered.  
 Args:  
 sequence (list): Sequence of integers.  
 Returns:  
 int: Sum of squares.  
 """  
 return sum(num \*\* 2 for num in sequence)  
  
def task2():  
 """  
 Task 2: Sum of squares with sequence initialization and length customization.  
 """  
 while True:  
 print("\n== Task 2: Sum of Sequence Squares ==")  
 print("1. Input sequence manually")  
 print("2. Generate sequence automatically")  
 print("0. Exit")  
 choice = input("Choose an option: ").strip()  
  
 if choice == '1':  
 sequence = input\_sequence()  
 print("User input sequence:", sequence)  
 elif choice == '2':  
 length = input\_int("Enter the length of the sequence: ", 1)  
 sequence = list(generate\_sequence(length))  
 print("Generated sequence:", sequence)  
 elif choice == '0':  
 print("Exiting Task 2...")  
 break  
 else:  
 print("Invalid choice. Please choose 1, 2, or 0.")  
 continue  
  
 total\_squares = sum\_of\_squares(sequence)  
 print(f"Sum of squares: {total\_squares}")

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

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

В строке, вводимой с клавиатуры, подсчитать количество заглавных английских букв.

from Decorator import log\_execution  
from Input import \*  
  
@log\_execution  
def count\_uppercase\_letters(input\_string):  
 *"""  
 Counts the number of uppercase English letters in the input string.  
 Args:  
 input\_string (str): The input text.  
 Returns:  
 int: Number of uppercase English letters.  
 """* return sum(1 for char in input\_string if 'A' <= char <= 'Z')  
  
def task3():  
 *"""  
 Task 3: Analyzes a string entered by the user to count uppercase English letters.  
 Allows user to input a string or generate a random one.  
 """* while True:  
 print("\n== Task 3: Count Uppercase English Letters ==")  
 print("1. Input string manually")  
 print("2. Generate random string")  
 print("0. Exit")  
  
 choice = input("Choose an option: ").strip()  
  
 if choice == '1':  
 input\_string = input("Enter a string (or type 'exit' to quit): ")  
  
 elif choice == '2':  
 try:  
 length = input\_int("Enter the desired length of the string: ", 1)  
 input\_string = generate\_random\_string(length)  
 print(f"Generated string: {input\_string}")  
 except ValueError:  
 print("Invalid input. Please enter a positive integer.")  
 continue  
  
 elif choice == '0':  
 print("Exiting Task 3...")  
 break  
  
 else:  
 print("Invalid choice. Please choose 1, 2, or 0.")  
 continue  
  
 uppercase\_count = count\_uppercase\_letters(input\_string)  
 print(f"Number of uppercase English letters: {uppercase\_count}")

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

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.»

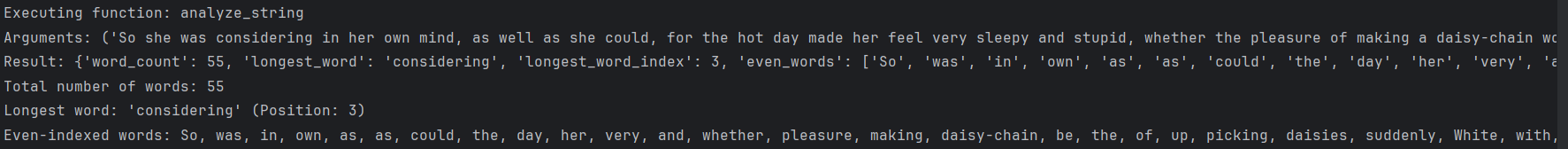
Если не оговорено иное, то регистр букв при решении задачи не имеет значения.

а) определить количество слов в строке;

б) найти самое длинное слово и его порядковый номер;

в) вывести каждое четное слово.

from Decorator import log\_execution  
  
@log\_execution  
def analyze\_string(input\_string):  
 *"""  
 Analyze a string to find word count, longest word, and even-indexed words.  
 Args:  
 input\_string (str): Text to analyze.  
 Returns:  
 dict: Analysis results (word count, longest word, even words).  
 """* words = [word.strip(",") for word in input\_string.split()]  
 word\_count = len(words)  
 longest\_word = max(words, key=len)  
 longest\_word\_index = words.index(longest\_word)  
 even\_words = [word for i, word in enumerate(words) if i % 2 == 0]  
 return {  
 "word\_count": word\_count,  
 "longest\_word": longest\_word,  
 "longest\_word\_index": longest\_word\_index,  
 "even\_words": even\_words  
 }  
  
def task4():  
 *"""  
 Task 4: Analyze a predefined string and display results.  
 """* print("\n== Task 4: Analyze string ==")  
  
 input\_string = (  
 "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."  
 )  
  
 results = analyze\_string(input\_string)  
  
 print(f"Total number of words: {results['word\_count']}")  
 print(f"Longest word: '{results['longest\_word']}' (Position: {results['longest\_word\_index']})")  
 print("Even-indexed words:", ", ".join(results["even\_words"]))



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 сумму отрицательных элементов списка и произведение элементов, расположенных между максимальным и минимальным элементами.

from Decorator import log\_execution  
from Input import \*  
  
@log\_execution  
def calculate\_product\_between\_min\_max(float\_list):  
 *"""  
 Calculate the product of elements between the minimum and maximum elements.  
 Args:  
 float\_list (list): List of floats.  
 Returns:  
 float: Product of elements between min and max.  
 """* min\_index = float\_list.index(min(float\_list))  
 max\_index = float\_list.index(max(float\_list))  
  
 if min\_index > max\_index:  
 min\_index, max\_index = max\_index, min\_index  
  
 product = 1  
 for x in float\_list[min\_index + 1:max\_index]:  
 product \*= x  
  
 return product  
  
def display\_list(float\_list):  
 *"""  
 Function to display the float list on the screen.  
 Args:  
 float\_list (list): The list of floats to display.  
 """* print("List of floats:", float\_list)  
  
@log\_execution  
def calculate\_negatives\_sum(float\_list):  
 *"""  
 Calculate the sum of negative elements in the list.  
 Args:  
 float\_list (list): List of floats.  
 Returns:  
 float: Sum of negative elements.  
 """* return sum(x for x in float\_list if x < 0)  
  
def task5():  
 *"""  
 Task 5: Process a float list with user input or automatic generation and display results.  
 Includes:  
 1) Manual input or automatic generation of float list.  
 2) Displaying the list.  
 3) Calculating sum of negative elements.  
 4) Calculating product of elements between min and max.  
 """* while True:  
 print("\n== Task 5: Process Float List ==")  
 print("1. Input list manually")  
 print("2. Generate list automatically")  
 print("0. Exit")  
  
 choice = input("Choose an option: ").strip()  
  
 if choice == '1':  
 float\_list = input\_float\_list()  
 elif choice == '2':  
 try:  
 size = input\_int("Enter the size of the list: ", 1)  
 min\_value = input\_float("Enter the minimum value for the list: ")  
 max\_value = input\_float("Enter the maximum value for the list: ")  
 if min\_value >= max\_value:  
 print("Minimum value must be less than maximum value. Try again.")  
 continue  
 float\_list = generate\_float\_list(size, min\_value, max\_value)  
 except ValueError:  
 print("Invalid input. Please enter valid numbers.")  
 continue  
 elif choice == '0':  
 print("Exiting Task 5...")  
 break  
 else:  
 print("Invalid choice. Please choose 1, 2, or 0.")  
 continue  
  
 display\_list(float\_list)  
  
 negatives\_sum = calculate\_negatives\_sum(float\_list)  
 product\_between\_min\_max = calculate\_product\_between\_min\_max(float\_list)  
  
 print(f"Sum of negative elements: {negatives\_sum}")  
 print(f"Product of elements between min and max: {product\_between\_min\_max}")

Изображение выглядит как текст, снимок экрана, меню

Контент, сгенерированный ИИ, может содержать ошибки.