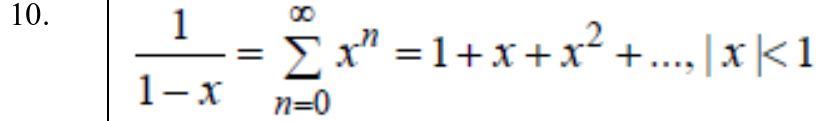
**Задание 1.** В соответствии с заданием своего варианта составить программу для вычисления значения функции c помощью разложения функции в степенной ряд. Задать точность вычислений eps.

Предусмотреть максимальное количество итераций, равное 500.

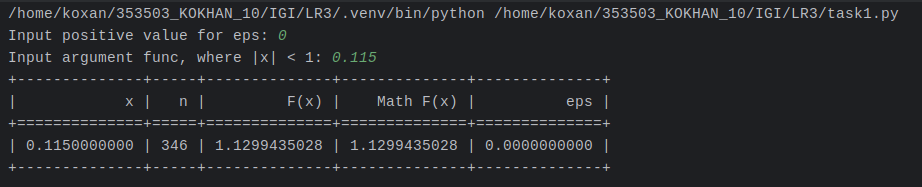
Вывести количество членов ряда, необходимых для достижения указанной точности вычислений. Результат получить в виде:



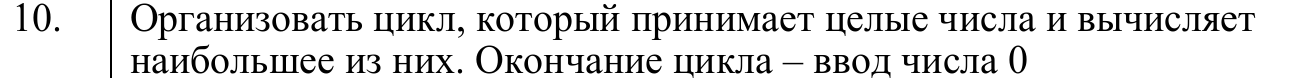
Здесь x – значение аргумента, F(x) – значение функции, n – количество просуммированных членов ряда, Math F(x) – значение функции, вычисленное с помощью модуля math.



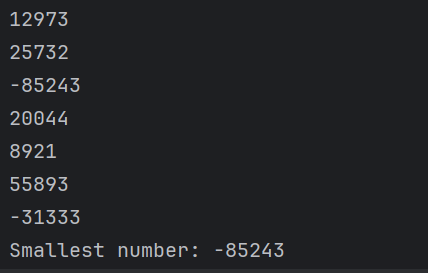
Код позже

Результат работы:  


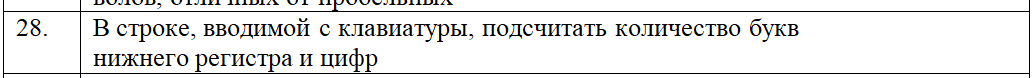
**Задание 2.** В соответствии с заданием своего варианта составить программу для нахождения суммы последовательности чисел.



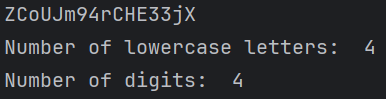
import Input\_data  
  
def Task2():  
 *"""  
 Function to find the smallest number among user-provided or randomly generated integers.  
  
 This function prompts the user for manual or automatic input of integers.  
 It then finds and displays the smallest number among the provided integers.  
  
 Args: None  
  
 Returns: None  
 """* minimum\_number = None  
 choice = Input\_data.Input\_data("Write 1 for manual input, 2 for automatic input: ", int, 1, 2)  
 if choice == 1:  
 while True:  
 num = Input\_data.Input\_data("Enter an integer (enter 1 to finish): ", int, None, None)  
  
 if num == 1:  
 break  
  
 if minimum\_number is None:  
 minimum\_number = num  
 else:  
 if num < minimum\_number:  
 minimum\_number = num  
 elif choice == 2:  
 x = Input\_data.Random\_Input(int, 1, 25)  
 while x > 0:  
 num = Input\_data.Random\_Input(int, -100000, 100000)  
 if num == 1:  
 break  
 print(num)  
 if minimum\_number is None:  
 minimum\_number = num  
 else:  
 if num < minimum\_number:  
 minimum\_number = num  
 x -= 1  
  
 print("Smallest number:", minimum\_number)



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



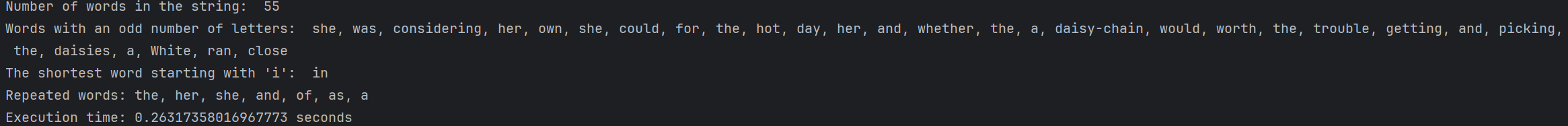
import Input\_data  
  
def Task3():  
 *"""  
 Function to count the number of lowercase letters and digits in a string.  
  
 This function prompts the user for manual or automatic input of a string.  
 It then counts and displays the number of lowercase letters and digits in the string.  
  
 Args: None  
  
 Returns: None  
 """* global user\_input  
 choice = Input\_data.Input\_data("Write 1 for manual input, 2 for automatic input: ", int, 1, 2)  
 if choice == 1:  
 user\_input = Input\_data.Input\_data("Enter a string: ", str, None, None)  
 elif choice == 2:  
 user\_input = Input\_data.Random\_Input(str, 10, 50)  
 print(user\_input)  
  
 lowercase\_count = 0  
 digit\_count = 0  
  
 for char in user\_input:  
 if char.islower():  
 lowercase\_count += 1  
 elif char.isdigit():  
 digit\_count += 1  
  
 print("Number of lowercase letters: ", lowercase\_count)  
 print("Number of digits: ", digit\_count)



**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.»

import time  
  
def measure\_time(func):  
 *"""  
 Decorator function to measure the execution time of a function.  
  
 This decorator wraps the input function and measures the time taken for its execution.  
 It then prints the execution time in seconds.  
  
 Args:  
 - func (function): The function to be decorated.  
  
 Returns:  
 - wrapper (function): The wrapper function.  
 """* def wrapper(\*args, \*\*kwargs):  
 start\_time = time.time()  
 result = func(\*args, \*\*kwargs)  
 end\_time = time.time()  
 print("Execution time:", end\_time - start\_time, "seconds")  
 return result  
 return wrapper  
  
@measure\_time  
def Task4():  
 *"""  
 Function to process text and perform various operations.  
  
 This function counts the number of words in a text, identifies words with an odd number  
 of letters, finds the shortest word starting with 'i', and identifies repeated words.  
 It also measures the execution time of the function.  
  
 Args: None  
  
 Returns: None  
 """* 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.")  
  
 words = text.replace(',', '').split()  
 odd\_length\_words = [word for word in words if len(word) % 2 != 0]  
 print("Number of words in the string: ", len(words))  
 print("Words with an odd number of letters: ", ", ".join(odd\_length\_words))  
  
 i\_words = [word for word in words if word.startswith('i')]  
 shortest\_i\_word = min(i\_words, key=len, default=None)  
 print("The shortest word starting with 'i': ", shortest\_i\_word)  
  
 unique\_words = set(words)  
 repeated\_words = {word for word in unique\_words if words.count(word) > 1}  
 print("Repeated words:", ", ".join(repeated\_words))  
  
 for \_ in range(10000000):  
 pass



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

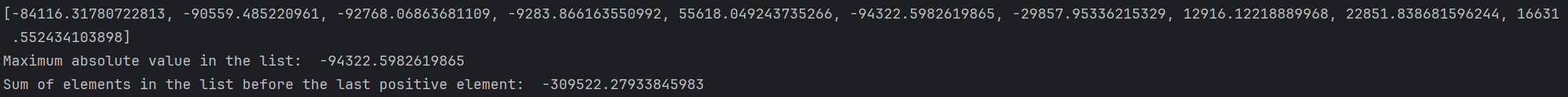
1) ввод элементов списка пользователем;

2) проверка корректности вводимых данных;

3) реализация основного задания с выводом результатов;

4) вывод списка на экран.

import time  
  
def measure\_time(func):  
 *"""  
 Decorator function to measure the execution time of a function.  
  
 This decorator wraps the input function and measures the time taken for its execution.  
 It then prints the execution time in seconds.  
  
 Args:  
 - func (function): The function to be decorated.  
  
 Returns:  
 - wrapper (function): The wrapper function.  
 """* def wrapper(\*args, \*\*kwargs):  
 start\_time = time.time()  
 result = func(\*args, \*\*kwargs)  
 end\_time = time.time()  
 print("Execution time:", end\_time - start\_time, "seconds")  
 return result  
 return wrapper  
  
@measure\_time  
def Task4():  
 *"""  
 Function to process text and perform various operations.  
  
 This function counts the number of words in a text, identifies words with an odd number  
 of letters, finds the shortest word starting with 'i', and identifies repeated words.  
 It also measures the execution time of the function.  
  
 Args: None  
  
 Returns: None  
 """* 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.")  
  
 words = text.replace(',', '').split()  
 odd\_length\_words = [word for word in words if len(word) % 2 != 0]  
 print("Number of words in the string: ", len(words))  
 print("Words with an odd number of letters: ", ", ".join(odd\_length\_words))  
  
 i\_words = [word for word in words if word.startswith('i')]  
 shortest\_i\_word = min(i\_words, key=len, default=None)  
 print("The shortest word starting with 'i': ", shortest\_i\_word)  
  
 unique\_words = set(words)  
 repeated\_words = {word for word in unique\_words if words.count(word) > 1}  
 print("Repeated words:", ", ".join(repeated\_words))  
  
 for \_ in range(10000000):  
 pass



Функция ввода

import random  
import string  
  
def Input\_data(prompt, data\_type, min\_value=None, max\_value=None):  
 *"""  
 Function to get user input with data type validation and optional value range constraints.  
  
 Args:  
 - prompt (str): The prompt message for the user.  
 - data\_type (type): The expected data type for user input (int, float, or str).  
 - min\_value (int/float, optional): The minimum allowed value (inclusive). Defaults to None.  
 - max\_value (int/float, optional): The maximum allowed value (inclusive). Defaults to None.  
  
 Returns:  
 - user\_input (int/float/str): The validated user input.  
  
 Raises:  
 - ValueError: If the user input does not match the specified data type or falls outside the specified range.  
 """* while True:  
 try:  
 user\_input = data\_type(input(prompt))  
 if min\_value is not None and user\_input < min\_value:  
 raise ValueError(f"Value must be at least {min\_value}")  
 if max\_value is not None and user\_input > max\_value:  
 raise ValueError(f"Value must be at most {max\_value}")  
 return user\_input  
 except ValueError as e:  
 print(f"Error: {e}. Please enter a valid value.")  
  
def Random\_Input(data\_type, min\_value=None, max\_value=None):  
 *"""  
 Function to generate random data based on the specified data type and optional value range constraints.  
  
 Args:  
 - data\_type (type): The data type for the generated value (int, float, or str).  
 - min\_value (int/float, optional): The minimum allowed value (inclusive). Defaults to None.  
 - max\_value (int/float, optional): The maximum allowed value (inclusive). Defaults to None.  
  
 Returns:  
 - generated\_value (int/float/str): The randomly generated value.  
  
 Raises:  
 - ValueError: If the specified data type is not supported (supported types: int, float, str).  
 """* if data\_type == str:  
 if min\_value is None:  
 min\_value = 1  
 if max\_value is None:  
 max\_value = 10  
  
 generated\_value = ''.join(  
 random.choices(string.ascii\_letters + string.digits, k=random.randint(min\_value, max\_value)))  
 return generated\_value  
  
 if min\_value is None:  
 min\_value = float('-inf')  
 if max\_value is None:  
 max\_value = float('inf')  
  
 if data\_type == int:  
 generated\_value = random.randint(min\_value, max\_value)  
 elif data\_type == float:  
 generated\_value = random.uniform(min\_value, max\_value)  
 else:  
 raise ValueError("Unsupported data type. Supported types are int, float, and str.")  
  
 return generated\_value

Декоратор:

def measure\_time(func):  
 *"""  
 Decorator function to measure the execution time of a function.  
  
 This decorator wraps the input function and measures the time taken for its execution.  
 It then prints the execution time in seconds.  
  
 Args:  
 - func (function): The function to be decorated.  
  
 Returns:  
 - wrapper (function): The wrapper function.  
 """*

Меню:

import Task1  
import Task2  
import Task3  
import Task4  
import Task5  
import Input\_data  
  
def main\_menu():  
 *"""  
 Function to display the main menu and execute the selected task.  
  
 The function displays a menu with options for each task and prompts the user for input.  
 Based on the user's choice, it executes the corresponding task or exits the program.  
  
 Args: None  
  
 Returns: None  
 """* while True:  
 print("MENU")  
 print("1. Task 1")  
 print("2. Task 2")  
 print("3. Task 3")  
 print("4. Task 4")  
 print("5. Task 5")  
 print("0. Exit")  
  
 choice = Input\_data.Input\_data("Enter task number: ", int, 0, 5)  
  
 if choice == 1:  
 Task1.Task1()  
 elif choice == 2:  
 Task2.Task2()  
 elif choice == 3:  
 Task3.Task3()  
 elif choice == 4:  
 Task4.Task4()  
 elif choice == 5:  
 Task5.Task5()  
 elif choice == 0:  
 print("Exiting program...")  
 break