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.

10. $$\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n = 1 + x + x^2 + \ldots, |x| < 1$$

```python
from tabulate import tabulate

import inputValidate

DESCRIPTION = """Write a program to calculate the value of the function 1 / (1 - x)
    using its power series expansion: 1 / (1 - x) = 0∑N x^n"""
MAX_INTERATION = 500
TITLE_TABLE = ["x", "n", "F(x)", "Math F(x)", "eps"]


def get_validate_inputs():  1 usage  Artko06 *
    """Function to receive input from the user"""
    eps = inputValidate.input_data_with_random( description: "Input positive value for eps: ",
                                                float,  min_value: 0, is_generate_random=True)
    x = inputValidate.input_data_with_random( description: "Input argument func, where |x| < 1: ",
                                              float, -0.9999999999999999,  max_value: 0.9999999999999999,
                                              is_generate_random=True)

    return eps, x


def calculate(eps, x):  1 usage  Artko06
    """Function for calculating the sum of a series until a specified accuracy is reached"""
    math_fx = 1 / (1 - x)
    term = 1  # init first term of series(x^0)
    fx = term  # start value: x^0 = 1
    count_interation = 1

    while count_interation < MAX_INTERATION and abs(term) > eps:
        count_interation += 1
        term *= x  # now member of series
        fx += term

    return count_interation, fx, math_fx


def print_table(x, count_interation, fx, math_fx, eps):  1 usage  Artko06
    """Function for outputting results in a table"""
    value_table = [[x, count_interation, fx, math_fx, eps]]
    print(tabulate(value_table, headers=TITLE_TABLE, tablefmt="grid", floatfmt=".10f"))

```

```python
def print_description():  1 usage  Artko06
    """Function for describing the task condition"""
    print(DESCRIPTION, end="\n\n")


def task1():  1 usage  Artko06
    """The main function to start the whole process"""
    print_description()
    eps, x = get_validate_inputs()
    count_interation, fx, math_fx = calculate(eps, x)
    print_table(x, count_interation, fx, math_fx, eps)
```

Результат работы:

```
/home/koxan/353503_KOKHAN_10/IGI/LR3/.venv/bin/python /home/koxan/353503_KOKHAN_10/IGI/LR3/main.py

1. Run task1
2. Run task2
3. Run task3
4. Run task4
5. Run task5
0. Exit

Input number of task: 1
Запуск задачи: run_task1
Write a program to calculate the value of the function 1 / (1 - x)
    using its power series expansion: 1 / (1 - x) = 0∑N x^n

Input positive value for eps: 0.1
Input argument func, where |x| < 1: random
+---------------+-----+--------------+--------------+--------------+
|             x |   n |         F(x) |    Math F(x) |          eps |
+===============+=====+==============+==============+==============+
| -0.1874241224 |   3 | 0.8477036793 | 0.8421590746 | 0.1000000000 |
+---------------+-----+--------------+--------------+--------------+
Задача run_task1 выполнена за 12.7349 секунд.
```

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

10. Организовать цикл, который принимает целые числа и вычисляет наибольшее из них. Окончание цикла – ввод числа 0

```python
import inputValidate

DESCRIPTION = """Create a loop that takes integers and calculates the largest of them.
    The loop ends when the number 0 is entered"""


def get_numbers():  1 usage  Artko06
    """Function to get numbers from user"""
    list_numbers = []

    while True:
        num = inputValidate.input_data_with_random( description: "Input integer num or 0 for finishing input: ", int,
                                                   is_generate_random=True, is_printing_generate_value=True)
        if num != 0:
            list_numbers.append(num)
        else:
            break

    return list_numbers


def print_max_number(numbers):  1 usage  Artko06
    """Function to display the result"""
    if numbers:
        print(f"The largest number entered is: {max(numbers)}")
    else:
        print("The list is empty")


def print_description():  1 usage  Artko06
    """Function for describing the task condition"""
    print(DESCRIPTION, end="\n\n")


def task2():  1 usage  Artko06
    """The main function to start the whole process"""
    print_description()
    numbers = get_numbers()
    print_max_number(numbers)
```

Результат работы:

```
/home/koxan/353503_KOKHAN_10/IGI/LR3/.venv/bin/python /home/koxan/353503_KOKHAN_10/IGI/LR3/main.py

1. Run task1
2. Run task2
3. Run task3
4. Run task4
5. Run task5
0. Exit

Input number of task: 2
Запуск задачи: run_task2
Create a loop that takes integers and calculates the largest of them.
    The loop ends when the number 0 is entered

Input integer num or 0 for finishing input: random

Generated value: -1635566198

Input integer num or 0 for finishing input: 3
Input integer num or 0 for finishing input: -1
Input integer num or 0 for finishing input: 4
Input integer num or 0 for finishing input: 6
Input integer num or 0 for finishing input: 2
Input integer num or 0 for finishing input: 0
The largest number entered is: 6
Задача run_task2 выполнена за 16.2530 секунд.
```

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

10. | В строке, вводимой с клавиатуры, подсчитать количество символов, лежащих в диапазоне от 'g' до 'o'

```python
import inputValidate

DESCRIPTION = """In a line entered from the keyboard,
    count the number of characters lying in the range from 'g' to 'o'"""


def get_input_string():  1 usage  Artko06
    """Function to get a string from the user"""
    return inputValidate.input_data_with_random( description: "Input string: ", str,
                                                 is_generate_random=True,
                                                 is_printing_generate_value=True)  # g, h, i, j, k, l, m, n, o


def count_characters_in_range(string_value):  1 usage  Artko06
    """Function to count characters in a string in a given range from 'g' to 'o'"""
    count_characters = 0

    for character in string_value:
        if 'g' <= character <= 'o':
            count_characters += 1

    return count_characters


def print_result(count_characters):  1 usage  Artko06
    """Function to display the result"""
    print(f"Count characters in range from 'g' before 'o': {count_characters}")


def print_description():  1 usage  Artko06
    """Function for describing the task condition"""
    print(DESCRIPTION, end="\n\n")


def task3():  1 usage  Artko06
    """The main function to start the whole process"""
    print_description()
    string_value = get_input_string()
    count_characters = count_characters_in_range(string_value)
    print_result(count_characters)

```

Результат работы:

```
/home/koxan/353503_KOKHAN_10/IGI/LR3/.venv/bin/python /home/koxan/353503_KOKHAN_10/IGI/LR3/main.py

1. Run task1
2. Run task2
3. Run task3
4. Run task4
5. Run task5
0. Exit

Input number of task: 3
Запуск задачи: run_task3
In a line entered from the keyboard,
    count the number of characters lying in the range from 'g' to 'o'

Input string: random

Generated value: C8QyNEb

Count characters in range from 'g' before 'o': 0
Задача run_task3 выполнена за 3.9383 секунд.
```

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.»

10.	а) определить число слов, ограниченных пробелами; б) определить, сколько раз повторяется каждая буква; в) вывести по алфавиту словосочетания, отделенные запятыми

```python
DESCRIPTION = """a) Determine the number of words delimited by spaces
b) Determine how many times each letter is repeated
c) List the phrases separated by commas in alphabetical order"""
CONST_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."
)


def count_words_in_text(text):  1 usage  Artko06
    """Function for counting the number of words in a text"""
    words = text.replace(',', '').replace('.', '').split()

    return len(words)


def count_letter_repeatability(text):  1 usage  Artko06
    """Function to count the frequency of each letter in the text"""
    letter_dict = {letter: 0 for letter in "abcdefghijklmnopqrstuvwxyz" + "ABCDEFGHIJKLMNOPQRSTUVWXYZ"}

    for character in text:
        if character.isalpha():
            letter_dict[character] += 1

    return letter_dict
```

```python
def sort_phrases_in_text(text):  1 usage  Artko06
    """Function for sorting words in each phrase of text"""
    phrases = text.replace('.', '').split(", ")

    for i in range(len(phrases)):
        sorted_words = sorted(phrases[i].split())
        phrases[i] = ' '.join(sorted_words)

    return phrases


def print_description():  1 usage  Artko06
    """Function for describing the task condition"""
    print(DESCRIPTION, end="\n\n")


def task4():  1 usage  Artko06
    """The main function to start the whole process"""
    print_description()

    # ---------------------------------- a ----------------------------------------------------------
    word_count = count_words_in_text(CONST_STRING)
    print("a) Count words in the text:", word_count, "\n")

    # ---------------------------------- b ----------------------------------------------------------
    letter_dict = count_letter_repeatability(CONST_STRING)
    print("b) Repeatability of each letter:", letter_dict, "\n")

    # ---------------------------------- c ----------------------------------------------------------
    sorted_phrases = sort_phrases_in_text(CONST_STRING)
    print("c) Sorted phrases:", sorted_phrases)

```

Результат работы:

```
/home/koxan/353503_KOKHAN_10/IGI/LR3/.venv/bin/python /home/koxan/353503_KOKHAN_10/IGI/LR3/main.py

1. Run task1
2. Run task2
3. Run task3
4. Run task4
5. Run task5
0. Exit

Input number of task: 4
Запуск задачи: run_task4
a) Determine the number of words delimited by spaces
b) Determine how many times each letter is repeated
c) List the phrases separated by commas in alphabetical order

a) Count words in the text: 55

b) Repeatability of each letter: {'a': 16, 'b': 5, 'c': 5, 'd': 13, 'e': 31, 'f': 4, 'g': 5, 'h': 17, 'i': 17, 'j': 0, 'k': 3, 'l': 10, 'm': 3

c) Sorted phrases: ['So considering her in mind own she was', 'as as could she well', 'and day feel for her hot made sleepy stupid the very',
Задача run_task4 выполнена за 0.0002 секунд.
```

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.

10. Найти минимальный положительный элемент списка и сумму элементов списка, расположенных между первым и последним положительными элементами

```python
import inputValidate

DESCRIPTION = """a) Find useful positive elements of the list
b) Find the composition of the list located between the first and last elements of the elements"""


def get_numbers():  1 usage  Artko06
    """Function to get numbers from user"""
    list_numbers = []

    while True:
        num = inputValidate.input_data_with_random( description: "Input float num or 0 for finishing input: ", float,
                                                    is_generate_random=True, is_printing_generate_value=True)
        if num != 0:
            list_numbers.append(num)
        else:
            break

    return list_numbers


def find_first_and_last_positive(numbers):  1 usage  Artko06
    """Function to find the indices of the first and last positive number"""
    index_first_positive = -1
    index_last_positive = -1

    for index in range(len(numbers)):
        if numbers[index] > 0:
            if index_first_positive == -1:
                index_first_positive = index
            index_last_positive = index

    return index_first_positive, index_last_positive


def find_min_positive(numbers):  1 usage  Artko06
    """Function to find the minimum positive number in a list"""
    return min(num for num in numbers if num > 0)
```

```python
def calculate_sum_between_first_and_last_positive(numbers, index_first_positive, index_last_positive):  1 usage  Artko06
    """Function to calculate the sum of elements between the first and last positive number"""
    if index_first_positive != index_last_positive:
        return sum(numbers[index_first_positive + 1:index_last_positive])
    else:
        return None


def print_description():  1 usage  Artko06
    """Function for describing the task condition"""
    print(DESCRIPTION, end="\n\n")


def task5():  1 usage  Artko06
    """The main function to start the whole process"""
    print_description()

    list_numbers = get_numbers()

    if list_numbers:
        index_first_positive, index_last_positive = find_first_and_last_positive(list_numbers)

        if index_first_positive != -1:
            print("Min positive number:", find_min_positive(list_numbers))

            sum_between = calculate_sum_between_first_and_last_positive(
                list_numbers, index_first_positive, index_last_positive
            )

            if sum_between is not None:
                print("Sum between first and last positive number:", sum_between)
            else:
                print("No elements between the first and last positive number")

        else:
            print("All numbers are negative")

    else:
        print("The list is empty")
```

```
/home/koxan/353503_KOKHAN_10/IGI/LR3/.venv/bin/python /home/koxan/353503_KOKHAN_10/IGI/LR3/main.py

1. Run task1
2. Run task2
3. Run task3
4. Run task4
5. Run task5
0. Exit

Input number of task: 5
Запуск задачи: run_task5
a) Find useful positive elements of the list
b) Find the composition of the list located between the first and last elements of the elements

Input float num or 0 for finishing input: random

Generated value: -4.902694482701124e+306

Input float num or 0 for finishing input: random

Generated value: 5.743538958096767e+306

Input float num or 0 for finishing input: 3
Input float num or 0 for finishing input: 4.3
Input float num or 0 for finishing input: q
Error: could not convert string to float: 'q'. Please enter a valid value.
Input float num or 0 for finishing input: 0
Min positive number: 3.0
Sum between first and last positive number: 3.0
Задача run_task5 выполнена за 13.9563 секунд.
```

Функции ввода

Обычная:

```python
import generateRandom


def input_data(description, data_type, min_value=None, max_value=None):  1 usage  Artko06
    """Prompts the user for input, validates type and value constraints, and returns the valid input"""
    while True:
        try:

            user_input = data_type(input(description))

            if min_value is not None and user_input < min_value:
                raise ValueError(f"The value have to be greater or = then {min_value}")

            if max_value is not None and user_input > max_value:
                raise ValueError(f"The value have to be less or = then {max_value}")

            return user_input

        except ValueError as e:
            print(f"Error: {e}. Please enter a valid value.")


```

С рандомом:

```python
def input_data_with_random(description, data_type, min_value=None, max_value=None,  5 usages  Artko06
                           is_generate_random=False, is_printing_generate_value=False):
    """Prompts the user for input, validates type and value constraints, and returns the valid input,
    but generate random value, if user enter word 'random' or 'RANDOM'"""
    while True:
        try:
            user_input = input(description)

            if is_generate_random and user_input.upper() == "RANDOM":
                random_value = generateRandom.generate_random(data_type=data_type, min_value=min_value,
                                                              max_value=max_value,
                                                              is_printing_generate_value=is_printing_generate_value)
                return random_value
            else:
                user_input = data_type(user_input)

            if min_value is not None and user_input < min_value:
                raise ValueError(f"The value have to be greater or = then {min_value}")

            if max_value is not None and user_input > max_value:
                raise ValueError(f"The value have to be less or = then {max_value}")

            return user_input

        except ValueError as e:
            print(f"Error: {e}. Please enter a valid value.")
```

```python
def print_generate_number(generate_value):  2 usages  Artko06
    """Function for outputting result generate value"""
    print("\nGenerated value:", generate_value, "\n")


def generate_random(data_type, min_value=None, max_value=None, is_printing_generate_value=False):  1 usage  Artko06
    """Function to generate random data based on the specified data type and optional value range constraints"""

    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)))

        if is_printing_generate_value:
            print_generate_number(generated_value)

        return generated_value

    if min_value is None and data_type == float:
        min_value = -1e307
    elif min_value is None and data_type == int:
        min_value = -2_147_483_648

    if max_value is None and data_type == float:
        max_value = 1e307
    elif max_value is None and data_type == int:
        max_value = 2_147_483_647

    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.")

    if is_printing_generate_value:
        print_generate_number(generated_value)
    return generated_value
```

Декоратор с меню выбора задачи:

```python
import time

import inputValidate
import task1
import task2
import task3
import task4
import task5


# ------------------------------------------------------
# Lab Work №3
# Topic: Standard Data Types, Collections, Functions, Modules
# Goal: Master the basic syntax of Python, gain skills working with standard data types,
# collections, functions, modules, and reinforce them by developing interactive applications.
# Version: 1.0
# Developer: Kokhan Artyom
# Date of Development: 12.03.2025
# ------------------------------------------------------

def log_task_execution(func):  5 usages  Artko06
    def wrapper(*args, **kwargs):  Artko06
        start_time = time.time()  # Засекаем время начала выполнения
        print(f"Запуск задачи: {func.__name__}")
        result = func(*args, **kwargs)
        end_time = time.time()  # Засекаем время окончания выполнения
        print(f"Задача {func.__name__} выполнена за {end_time - start_time:.4f} секунд.")
        return result

    return wrapper

```

```python
@log_task_execution  1 usage  Artko06
def run_task1():
    task1.task1()


@log_task_execution  1 usage  Artko06
def run_task2():
    task2.task2()


@log_task_execution  1 usage  Artko06
def run_task3():
    task3.task3()


@log_task_execution  1 usage  Artko06
def run_task4():
    task4.task4()


@log_task_execution  1 usage  Artko06
def run_task5():
    task5.task5()


while True:
    print("\n1. Run task1\n2. Run task2\n3. Run task3\n4. Run task4\n5. Run task5\n0. Exit\n")

    select = inputValidate.input_data( description: "Input number of task: ", int,  min_value: 0,  max_value: 5)

    if select == 1:
        run_task1()
    elif select == 2:
        run_task2()
    elif select == 3:
        run_task3()
    elif select == 4:
        run_task4()
    elif select == 5:
        run_task5()
    else:
        break
```