Лабораторная работа №3.
Гимпель Кирилл гр. 253505
Вариант №5

Меню с выбором запускаемой задачи:

main.py > ...

```python
# Lab: 3
# Author: Gimpel Kirill
# Date: 10.04.2024
# Version: 1.0

from task1 import task1
from task2 import task2
from task3 import task3
from task4 import task4
from task5 import task5
from secure_input import secure_input


if __name__ == '__main__':
    while True:
        choice = secure_input(int, "Enter task number or press 0 to exit: ", lambda x: 0 <= x <= 5)
        match choice:
            case 0:
                break
            case 1:
                task1()
            case 2:
                task2()
            case 3:
                task3()
            case 4:
                task4()
            case 5:
                task5()

```

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.

5.	$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \dots$

```python

def exp_x(x: float, eps: float = 1e-5):
    """
        A function that finds e^x using a power series expansion.

        Args:
            x (float): The value of x for which the sum of the series is calculated.
            eps (float): The accuracy with which the value of e^x is to be obtained.

        Returns:
            float: e raised to the power of x.
            int: The number of iterations.
    """
    max_operation = 500
    n = 0
    term = 1.0
    sum = term
    while n <= max_operation and term > eps:
        n += 1
        term *= x / n
        sum += term
    return sum, n

```

```python

@define_task("Задание №1. Посчитать значение функции e^x с помощью \
разложения ее в ряд Тейлора")
def task1():
    """
        The function asks the user to input x and eps values. Calculates
        the value of the function e^x with eps precision using the function
        exp_x(x, eps) and compares it with the result of the function math.exp(x).
        Displays the result on the screen.
    """
    x = secure_input(float, "Enter x: ")
    eps = secure_input(float, "Enter eps: ", lambda val: val < 1)
    f = exp_x(x, eps)
    math_f = math.exp(x)
    print("x | n | F(x) | Math F(x) | eps")
    print(f"{x} | {f[1]} | {f[0]} | {math_f} | {eps}")
```

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

5.	Организовать цикл, который принимает целые числа с клавиатуры и подсчитывает количество неотрицательных чисел. Окончание цикла – ввод числа, меньшего – 100

```python
from secure_input import secure_input
from define_task import define_task


@define_task("Задание №2. Организовать цикл, который принимает целые \
числа с клавиатуры и подсчитывает количество неотрицательных чисел. \
Окончание цикла — ввод числа, меньшего –100")
def task2():
    """
        Function for calculation of the amount of non-negative numbers.

        The function asks the user to enter integers until a number less
        than -100 is entered. And then displays the result on the screen.
    """
    count = 0
    while True:
        val = secure_input(int, "Enter an integer. To complete, enter a number less than -100: ")
        if val < -100:
            break
        count += val >= 0
    print(f"The number of non-negative numbers among the data is {count}")

```

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

5.	В строке, вводимой с клавиатуры, подсчитать количество слов, начинающихся со строчной буквы

task3.py > ...

```python
from secure_input import secure_input
from define_task import define_task


@define_task("Задание №3. В строке, вводимой с клавиатуры, подсчитать \
количество слов, начинающихся со строчной буквы")
def task3():
    """
        Function to perform task 3.

        Task 3: In the string entered from the keyboard, count the number
        of words beginning with a lowercase letter.
    """
    source_string = input("Enter the string: ")
    words = source_string.split()
    lowercase_count = 0
    for word in words:
        lowercase_count += word[0].islower()
    print(f"The number of words that begin with a lowercase letter is equal to {lowercase_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.»

5.	а) определить, сколько слов имеют минимальную длину; б) вывести все слова, за которыми следует запятая; в) найти самое длинное слово, которое заканчивается на 'у'

```python
def count_min_words(text):
    """
        A function that determines how many words in the text have a minimum length.

        Args:
            text (str): Input text

        Returns:
            int: Number of words with minimum length.
    """
    text = text.replace(',', '')
    words = text.split()
    min_length = len(min(words, key=len))
    count = len(list(filter(lambda word: len(word) == min_length, words)))
    return count
```

```python
def get_all_words_followed_by_comma(text):
    """
        A function that finds all words followed by a comma in the text.

        Args:
            text (str): Input text

        Returns:
            list[str]: Words followed by a comma.
    """
    words = text.split()
    lst = list(filter(lambda word: word[-1] == ',', words))
    lst = list(map(lambda word: word.rstrip(','), lst))
    return lst
```

```python
def get_all_min_words_that_end_symbol(text, symbol):
    """
        A function that finds the longest word in the text ending with
        the given character.

        Args:
            text (str): Input text
            symbol (str): What symbol should the words end with?

        Returns:
            list[str]: Words followed by a symbol.
    """
    text = text.replace(',', '')
    words = text.split()
    words_end_symbol = list(filter(lambda word: word[-1] == symbol, words))
    min_length = len(min(words_end_symbol, key=len))
    return list(filter(lambda word: len(word) == min_length, words_end_symbol))
```

```python
@define_task("Задание №4.\n\
а) определить, сколько слов имеют минимальную длину;\n\
б) вывести все слова, за которыми следует запятая;\n\
в) найти самое длинное слово, которое заканчивается на 'y'")
def task4():
    """
        Function to perform task 4.

        Task 4 includes calling get_all_words_followed_by_comma,
        count_min_words, get_all_min_words_that_end_symbol functions
        on a certain input text.
    """
    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."

    print(f"{count_min_words(text)} words have a minimum length")
    print(f"All words followed by a comma: {get_all_words_followed_by_comma(text)}")
    print(f"Words that end in 'y': {get_all_min_words_that_end_symbol(text, 'y')}")
```

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.

5.	Найти максимальный по модулю элемент списка и сумму элементов списка, расположенных между первым и вторым положительными элементами

```python
def generator_number(size: int):
    """
        A function that asks the user for 'size' numbers of type float.

        Args:
            size: How many numbers to request from the user?
    """
    for i in range(size):
        yield secure_input(float, "Enter number: ")
```

```python
def max_abs_element(lst: list[float]):
    """
        A function that finds the maximum modulo element of the list.

        Args:
            lst: A list in which to search for the maximum modulo element.

        Returns:
            float: Maximum modulo element of the list.
    """
    return max(lst, key=lambda x: abs(x))
```

```python
def sum_between_first_and_second_positive(lst: list[float]):
    """
        A function that finds the sum of the list elements located between
        the first and second positive elements.

        If the second positive element is not found it will return 0.

        Args:
            lst: A list in which to search for the maximum modulo element.

        Returns:
            float: The sum of the elements of the list located between the
            first and second positive elements.
    """
    start = -1
    end = -1
    res = 0
    for i, num in enumerate(lst):
        if num > 0:
            if start == -1:
                start = i
                continue
            else:
                end = i
                break
        if start != -1:
            res += num
    if start == -1 or end == -1:
        res = 0
    return res
```

```python
@define_task("Задание №5. Найти максимальный по модулю элемент списка \
и сумму элементов списка, расположенных между первым и вторым положительными \
элементами")
def task5():
    """
        Function to perform task 5.

        Task 5 includes calling sum_between_first_and_second_positive,
        max_abs_element functions on a certain input list.
    """
    size_list = secure_input(int, "Enter list size: ", lambda x: x > 0)
    lst = list(generator_number(size_list))

    print(f"The maximum modulus element is equal to {max_abs_element(lst)}")
    print(f"Sum between first and second positive elements is equal to {sum_between_first_and_second_positive(lst)}")
```

Проверка на ввод:

```python
def secure_input(data_type: type, message: str, func: Callable[[type], bool] = None) -> type:
    """
        This function ensures the user inputs data of the correct type.
        If the user inputs data of the wrong type, it will keep prompting
        the user for input until they input data of the correct type.

        Args:
            data_type (type): The type of the data that the user is supposed to input.
            message (str): The message that is displayed to the user when prompting for input.
            func (function, optional): If this function returns False, the user will be prompted
            for input again.

        Returns:
            data_type: The user's input, converted to the specified data type.
    """
    again = True
    while again:
        try:
            data = data_type(input(message))
            again = not func(data) if func else False
        except ValueError:
            pass
        if again:
            print("Incorrect input!!!")
    return data
```

Декоратор:

define_task.py > ...

```python
def define_task(text_task: str):
    """
        A decorator for displaying the task description on the screen.
    """
    def actual_decorator(func):
        def task():
            print("====================")
            print(text_task)
            func()
            print("====================\n")
        return task
    return actual_decorator
```