инистерство образования Республики Беларусь Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе №3: По теме

«Стандартные типы данных, коллекции, функции, модули»

Выполнил: студент гр. 353502

Худницкий Александр Валерьевич

Проверил: Преподаватель

Жвакина Анна Васильевна

Минск 2025

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

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.

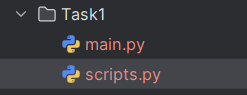
Предусмотреть максимальное количество итераций, равное 500.

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





Структура задания:



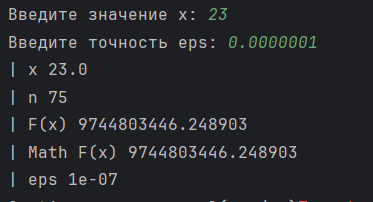
Содержание main.py:

# main.py  
# Purpose: Working with string  
# Lab: 3  
# Task: 1  
# Version: 1.0  
# Developer: Hudnitskii A.V. gr.353502  
# Date: 17-03-2025  
  
import math  
from scripts import calculate\_exp  
  
def main():  
 while True:  
 x = float(input("Введите значение x: "))  
 eps = float(input("Введите точность eps: "))  
 n, result = calculate\_exp(x, eps)  
  
 print(f"| x {x}")  
 print(f"| n {n}")  
 print(f"| F(x) {result}")  
 print(f"| Math F(x) {math.exp(x)}")  
 print(f"| eps {eps}")  
  
 continue\_program = input("Continue our program?(yes/no)")  
 if continue\_program != "yes":  
 break  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

Содержание scripts.py:

def calculate\_exp(x, eps):  
 *"""  
 Calculate exp.  
  
 Args: x (float), eps (float) : x and eps  
 Returns: n (int), result (float): Count of iterations, Result of func.  
 """* sum\_series = 0.0  
 n = 0  
 term = 1.0  
  
 while n < 500:  
 sum\_series += term  
 if abs(term) < eps:  
 break  
 n += 1  
 term = term \* x / n  
  
 return n, sum\_series

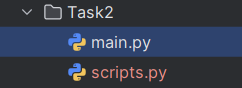
Результат выполенения программы:



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

**

Структура программы:



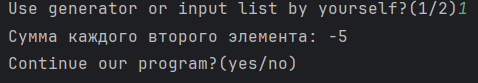
Содержание main.py:

# main.py  
# Purpose: Working with string  
# Lab: 3  
# Task: 2  
# Version: 1.0  
# Developer: Hudnitskii A.V. gr.353502  
# Date: 17-03-2025  
  
from scripts import sum\_of\_list  
from scripts import generate\_int\_list  
  
def main():  
 while True:  
 way\_to\_our\_text = ""  
 my\_list = []  
 while way\_to\_our\_text != '1' or way\_to\_our\_text != '2':  
 way\_to\_our\_text = input("Use generator or input list by yourself?(1/2)")  
 if way\_to\_our\_text == '1':  
 my\_list = generate\_int\_list(mode = 'auto')  
 break  
 elif way\_to\_our\_text == '2':  
 my\_list = generate\_int\_list(mode = 'by\_yourself')  
 break  
  
 print(f"Сумма каждого второго элемента: {sum\_of\_list(my\_list)}")  
 continue\_program = input("Continue our program?(yes/no)")  
 if continue\_program != "yes":  
 break  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

Содержание scripts.py:

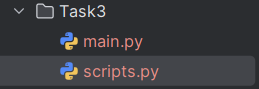
import random  
  
def sum\_of\_list(my\_list):  
 *"""  
 Counts the sum of list.  
  
 Args: my\_list (List) : Given the int list  
 Returns: sum (int) : The sum of the list.  
 """* return sum(my\_list)  
  
def generate\_int\_list(mode):  
 *"""  
 Generate int list.  
  
 Args: mode (string) : Mode to create list  
 Returns: my\_list (int) : Our int list.  
 """* my\_list = []  
 if mode == 'auto':  
 prev = 0  
 while prev != 1:  
 prev = random.randint(-5, 5)  
 my\_list.append(prev)  
  
 my\_list.remove(prev)  
 return [value for i, value in enumerate(my\_list) if i % 2 != 0]  
  
 elif mode == "by\_yourself":  
 while True:  
 try:  
 num = input("Введите элемент списка (или '1' для завершения ввода): ")  
 if num == '1':  
 break  
 my\_list.append(int(num))  
 except ValueError:  
 print("Ошибка: введите корректное вещественное число.")  
 return [value for i, value in enumerate(my\_list) if i % 2 != 0]

Результат выполнения программы:



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

Индивидуальное задание (вариант 25): в строке, вводимой с клавиатуры, подсчитать количество слов, начинающихся со строчной согласной буквы.



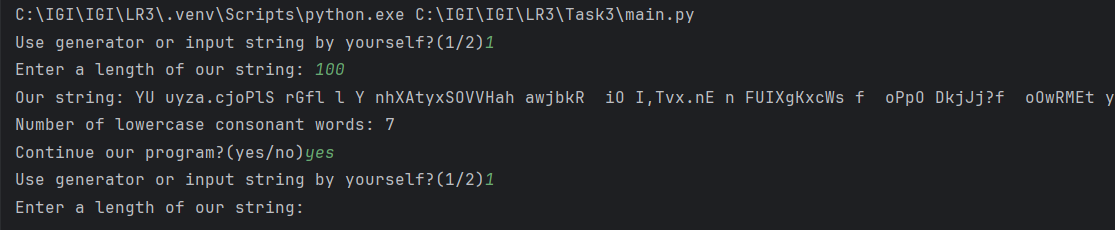
Содержание scritps.py:

import random  
  
def count\_lowercase\_consonant\_words(our\_text):  
 *"""  
 Counts the number of words that start with a lowercase consonant in the given text.  
  
 Args: our\_text (str) : Given string to find the number of specific words  
 Returns: count (int) : The count of words starting with a lowercase consonant.  
 """* vowels = {"a", "e", "i", "o", "u", "y"}  
 separators = {" ", ",", ".", ":", ";", "(", ")", "[", "]", "<", ">", "/", "|"}  
 count = 0  
 for i in range(len(our\_text) - 1):  
 if i == 0:  
 if 97 <= ord(our\_text[i]) <= 122 and our\_text[i] not in vowels:  
 count += 1  
 if our\_text[i] in separators and 97 <= ord(our\_text[i + 1]) <= 122 and our\_text[i + 1] not in vowels:  
 count += 1  
  
 return count  
  
  
def our\_text\_generator(length\_our\_text):  
 *"""  
 Generates a random string of a given length using English letters (both cases) and separators.  
  
 Args: length (int): The length of the string.  
 Returns: our\_text (string): A randomly generated string.  
 """* our\_chars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ ,.!?;:'  
 our\_text = ''.join(random.choice(our\_chars) for char in range(length\_our\_text))  
  
 return our\_text

Содержание main.py:

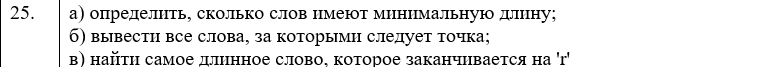
# main.py  
# Purpose: Count lowercase consonant words  
# Lab: 3  
# Task: 3  
# Version: 1.0  
# Developer: Hudnitskii A.V. gr.353502  
# Date: 13-03-2025  
  
from scripts import count\_lowercase\_consonant\_words  
from scripts import our\_text\_generator  
  
def main():  
 while True:  
 way\_to\_our\_text, our\_text = "", ""  
 while way\_to\_our\_text != '1' or way\_to\_our\_text != '2':  
 way\_to\_our\_text = input("Use generator or input string by yourself?(1/2)")  
 if way\_to\_our\_text == '1':  
 length\_our\_text = ""  
 while not (length\_our\_text.isdigit() and int(length\_our\_text) > 0):  
 length\_our\_text = input("Enter a length of our string: ")  
 length\_our\_text = int(length\_our\_text)  
 our\_text = our\_text\_generator(length\_our\_text)  
 print(f"Our string: {our\_text}")  
 break  
 elif way\_to\_our\_text == '2':  
 our\_text = input("Enter a string: ")  
 break  
  
 count = count\_lowercase\_consonant\_words(our\_text)  
 print(f"Number of lowercase consonant words: {count}")  
  
 continue\_program = input("Continue our program?(yes/no)")  
 if continue\_program != "yes":  
 break  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

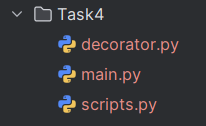
Результат выполнения программы:



Задание 4. 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.»





Содержание decorator.py:

def my\_decorator(func):  
 def wrapper():  
 print("Task started")  
 func()  
 print("Task ended")  
 return wrapper()

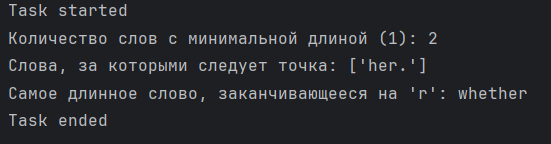
Содержание scripts.py:

def min\_length\_count(input\_string):  
 *"""  
 Counts the number of words with minimal length in the given text.  
  
 Args: input\_string (str) : Given string  
 Returns: count (int) : The count of words starting with minimal length.  
 """* words\_in\_string = input\_string.split()  
 min\_length = min(len(word) for word in words\_in\_string)  
 count = 0  
 for word in words\_in\_string:  
 if len(word) == min\_length:  
 count += 1  
 print(f"Количество слов с минимальной длиной ({min\_length}): {count}")  
  
  
  
def point\_before(input\_string):  
 *"""  
 Find all words after point.  
  
 Args: input\_string (str) : Given string  
 Returns: count (int) : The list of words starting after point.  
 """* words\_in\_string = input\_string.split()  
 words\_before\_point = []  
  
 for i in range(len(words\_in\_string) - 1):  
 if words\_in\_string[i + 1].endswith("."):  
 words\_before\_point.append(words\_in\_string[i + 1])  
  
 print("Слова, за которыми следует точка:", words\_before\_point)  
  
  
def max\_length\_with\_r(input\_string):  
 *"""  
 Find the longest word that ends on "r".  
  
 Args: input\_string (str) : Given string  
 Returns: count (int) : The longest word that ends on "r"t.  
 """* words\_in\_string = input\_string.split()  
  
 words\_ending\_with\_r = []  
 for word in words\_in\_string:  
 if word.lower().endswith('r'):  
 words\_ending\_with\_r.append(word)  
  
 longest\_word = max(words\_ending\_with\_r, key=len)  
 print("Самое длинное слово, заканчивающееся на 'r':", longest\_word)

Содержание main.py:

# main.py  
# Purpose: Working with string  
# Lab: 3  
# Task: 4  
# Version: 1.0  
# Developer: Hudnitskii A.V. gr.353502  
# Date: 17-03-2025  
  
from Task4.decorator import my\_decorator  
from Task4.scripts import min\_length\_count  
from Task4.scripts import point\_before  
from Task4.scripts import max\_length\_with\_r  
  
@my\_decorator  
def main():  
 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."  
 )  
 min\_length\_count(input\_string)  
 point\_before(input\_string)  
 max\_length\_with\_r(input\_string)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

Вывод программы:



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;

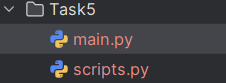
2) проверка корректности вводимых данных;

3) реализация основного задания с выводом результатов;

4) вывод списка на экран.



Структура задания:



Содержание main.py:

# main.py  
# Purpose: Working with string  
# Lab: 3  
# Task: 5  
# Version: 1.0  
# Developer: Hudnitskii A.V. gr.353502  
# Date: 17-03-2025  
  
from scripts import find\_min\_element  
from scripts import sum\_between\_first\_last\_positive  
from scripts import generate\_float\_list  
  
def main():  
 while True:  
 my\_list = []  
 way\_to\_our\_text = ""  
 while way\_to\_our\_text != '1' or way\_to\_our\_text != '2':  
 way\_to\_our\_text = input("Use generator or input list by yourself?(1/2)")  
 if way\_to\_our\_text == '1':  
 length\_our\_list = ""  
 while not (length\_our\_list.isdigit() and int(length\_our\_list) > 0):  
 length\_our\_list = input("Enter a length of our list: ")  
 length\_our\_list = int(length\_our\_list)  
 my\_list = generate\_float\_list(mode='auto', length\_our\_list = length\_our\_list)  
 break  
 elif way\_to\_our\_text == '2':  
 my\_list = generate\_float\_list(mode='manual', length\_our\_list = 0)  
 break  
  
 print(f"Список: {my\_list}")  
 print(f"Минимальный по модулю элемент: {find\_min\_element(my\_list)}")  
 print(f"Сумма элементов между первым и последним положительными элементами: {sum\_between\_first\_last\_positive(my\_list)}")  
  
 continue\_program = input("Continue our program?(yes/no)")  
 if continue\_program != "yes":  
 break  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

Содержание scripts.py:

import random  
from email.contentmanager import raw\_data\_manager  
  
  
def find\_min\_element(my\_list):  
 *"""  
 Find min abs element in float list.  
  
 Args: my\_list (List) : Given list  
 Returns: element (float) : Minimal abs element.  
 """* return min(my\_list, key=abs)  
  
  
def sum\_between\_first\_last\_positive(my\_list):  
 *"""  
 Counts the sum between first and last positive in the list.  
  
 Args: my\_list (List) : Given float list  
 Returns: sum (float) : The sum between first and last positive in the list.  
 """* first\_positive\_index = None  
 for i in range(len(my\_list)):  
 if my\_list[i] > 0:  
 first\_positive\_index = i  
 break  
  
 last\_positive\_index = None  
 for i in range(len(my\_list) - 1, -1, -1):  
 if my\_list[i] > 0:  
 last\_positive\_index = i  
 break  
  
 if first\_positive\_index is None or last\_positive\_index is None:  
 return 0  
 else:  
 return sum(my\_list[first\_positive\_index + 1:last\_positive\_index])  
  
def generate\_float\_list(mode, length\_our\_list):  
 *"""  
 Generate float list.  
  
 Args: mode (string), length\_our\_list (int) : Mode to create list, length of list  
 Returns: my\_list (float) : Our float list.  
 """* if mode == 'auto':  
 return [random.uniform(-100.0, 100.0) for \_ in range(length\_our\_list)]  
 else:  
 my\_list = []  
 while True:  
 try:  
 num = input("Введите элемент списка (или 'q' для завершения ввода): ")  
 if num.lower() == 'q':  
 break  
 my\_list.append(float(num))  
 except ValueError:  
 print("Ошибка: введите корректное вещественное число.")  
 return my\_list

Результат выполнения программы:

