

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Кафедра систем управління літальними апаратами

Лабораторна робота № 2

з дисципліни «Об'єктно-орієнтоване програмування СУ»

Тема: «Розробка структурованих програм з розгалуженням та
повтореннями»

ХАІ.301 .173. 310ст.2 ЛР

Виконав студент гр. _____310ст_____

_____Возвишаєв Олексій Андрійович_____

(підпис, дата)

(П.І.Б.)

Перевірив

_____к.т.н., доц. О. В. Гавриленко

_____ас. В. О. Білозерський

(підпис, дата)

(П.І.Б.)

Лабораторна робота №2

з дисципліни «Об'єктно-орієнтоване
програмування авіаційно-транспортних систем»

Тема: "Розробка структурованих програм з розгалуженням та повтореннями"

МЕТА РОБОТИ

Вивчити теоретичний матеріал щодо синтаксису на мові Python і поданням у вигляді UML діаграм діяльності алгоритмів з розгалуження та циклами, а також навчитися використовувати функції, інструкції умовного переходу і циклів для реалізації інженерних обчислень.

ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Вирішити завдання на алгоритми з розгалуженням.

Завдання 2. Дано дійсні числа (x_i, y_i) , $i = 1, 2, \dots, n$, – координати точок на площині. Визначити кількість точок, що потрапляють в геометричну область заданого кольору (або групу областей).

Завдання 3. Дослідити ряд на збіжність. Умова закінчення циклу обчислення суми прийняти у вигляді: $|u_n| < \epsilon$ або $|u_n| > G$ де ϵ – мала величина для переривання циклу обчислення суми сходиться ряду ($\epsilon = 10^{-5} \dots 10^{-20}$); g – величина для переривання циклу обчислення суми розходиться ряду ($g = 10^2 \dots 10^5$).

Завдання 4. Для багаторазового виконання будь-якого з трьох зазначених вище завдань на вибір розробити циклічний алгоритм організації меню в командному вікні.

ВИКОНАННЯ РОБОТИ

Завдання 1. Вирішення задачі (If) 19

Вхідні дані:

num1 перше ціле число int ціле число

num2 друге ціле число int ціле число

num3 третє ціле число int ціле число

num4 четверте ціле число int ціле число

Вихідні дані:

output Порядковий номер числа, яке відрізняється від трьох інших (1, 2, 3 або 4). Тип int

error_msg Повідомлення про помилку, якщо введені дані не є цілими числами. Тип str

Алгоритм вирішення показано на рис. 1

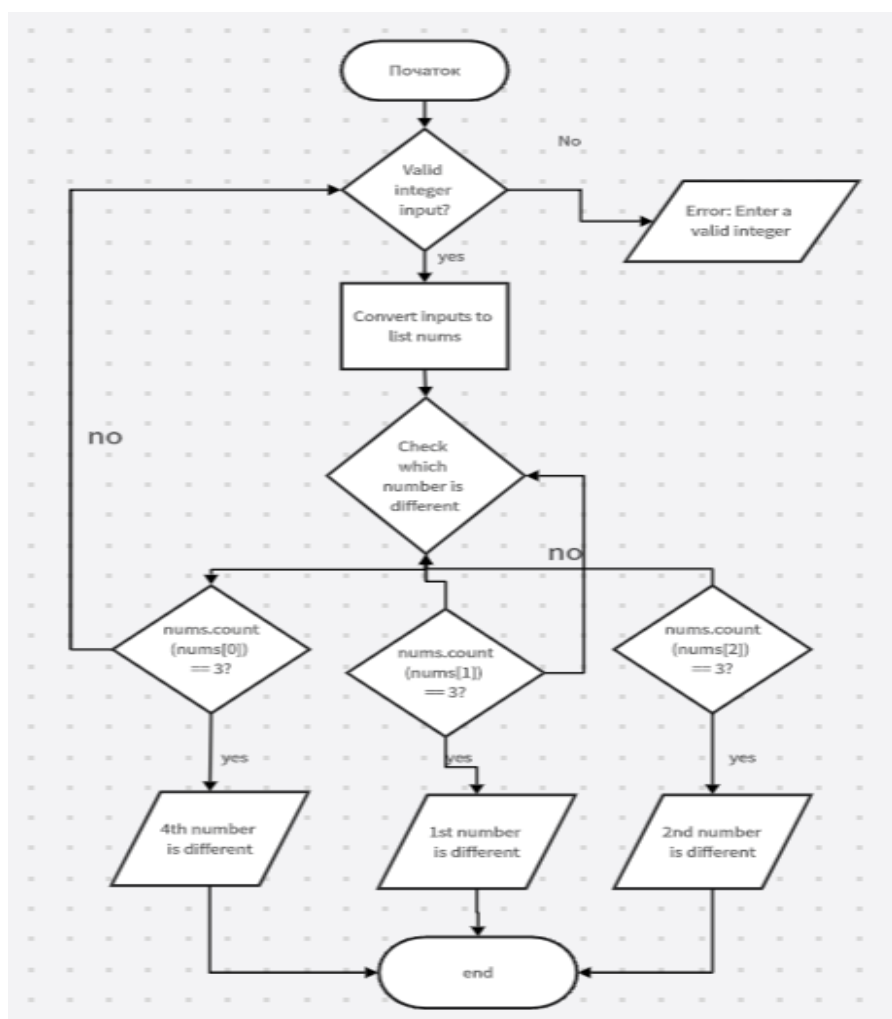


Рисунок 1- Блок схема алгоритма вирішення до Завдання 1

Лістинг коду вирішення задачі наведено в дод. А (стор. 7). Екран роботи програми показаний на рис. Б.11.

Завдання 2. Вирішення задачі Т.2.геом.обл. 12

Вхідні дані: Радіус: 1.0 Точки: [(0.5, 1.5), (1.2, 0.8), (2.0, 2.0)]

Вихідні дані: Кількість точок, що потрапляють у жовту область: 2

Алгоритм вирішення показано на рис. 2

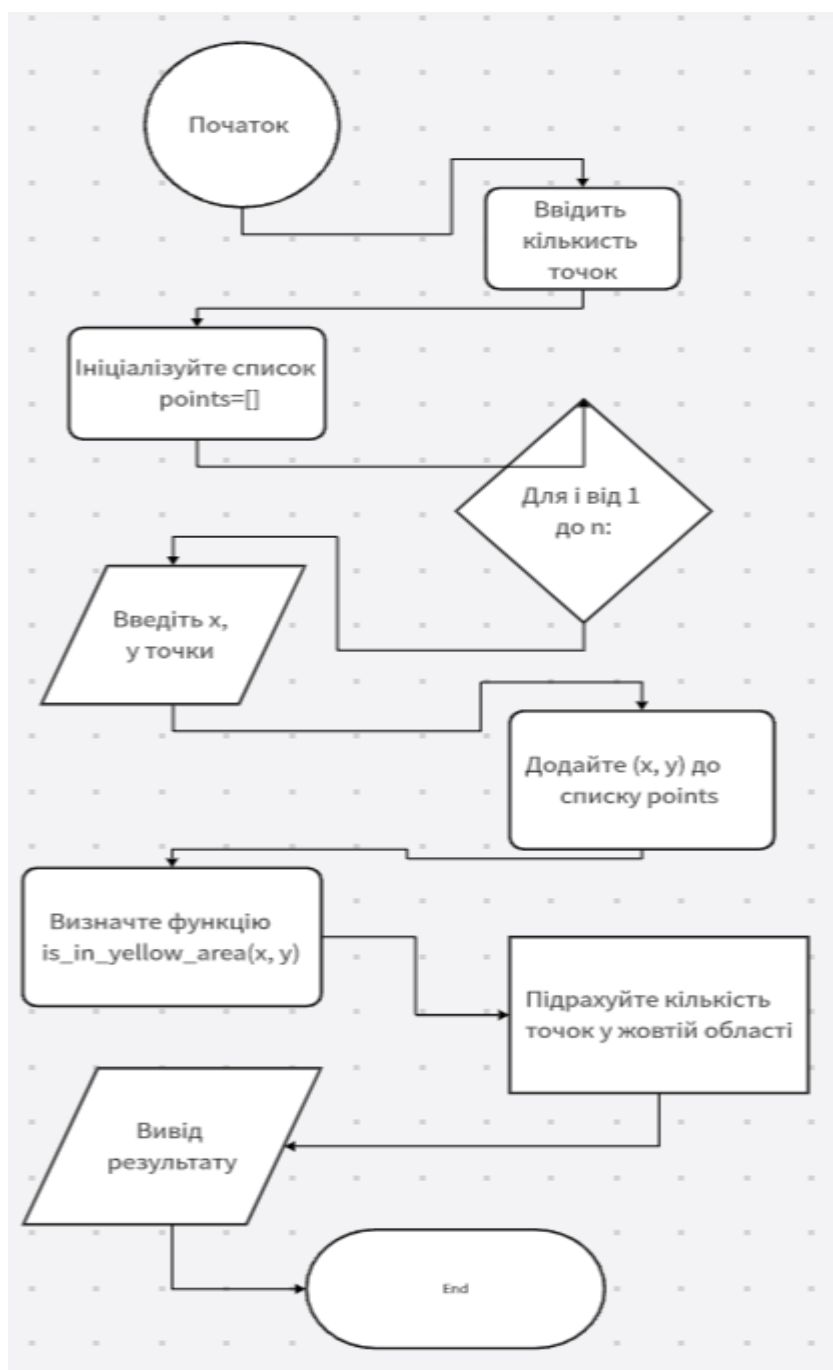


Рисунок 2- Блок схема алгоритма вирішення до Завдання 2

Лістинг коду вирішення задачі наведено в дод. А (стор.7). Екран роботи програми показаний на рис. Б.<номер>.

Завдання 3. Вирішення задачі (Т.3.ряди) 11

Вхідні дані

epsilon – межа збіжності (float, $10^{-5} \leq \epsilon \leq 10^{-20}$)

g – межа розбіжності (float, $10^2 \leq g \leq 10^5$)

Вихідні дані

total_sum – сума ряду (float).

n – номер ітерації зупинки (int).

message – текстове повідомлення ("Ряд збігається" або "Ряд розбігається").

Алгоритм вирішення показано нижче

ініціалізація параметрів:

Початкове значення для лічильника $n = 1$.

Ініціалізуємо суму $total_sum = 0$.

Встановлюємо максимальну кількість ітерацій max_iter .

Основний цикл обчислення:

Повторюємо обчислення для кожного члена ряду до тих пір, поки кількість ітерацій не перевищить max_iter або поки ряд не збіжиться або розбіжиться.

Обчислення чисельника:

Для кожного кроку n , обчислюємо чисельник члена ряду: факторіал числа $(2n+1)$

Обчислення знаменника:

Обчислення поточного члена ряду:

Перевірка умов збіжності або розбіжності:

Завершення обчислень:

Лістинг коду вирішення задачі наведено в дод. А (стор. 8) Екран роботи програми показаний на рис. Б.12.

Завдання 4.

Вхідні дані:

epsilon: Точність для перевірки збіжності. Тип: float Обмеження: 10^{-5} до 10^{-20}

g: Параметр для перевірки розбіжності. Тип: float Обмеження: 10^2 до 10^5

max_iter: Максимальна кількість ітерацій для обчислень. Тип: int
Обмеження: 1000 до 10000

Вихідні дані:

total_sum: Сума ряду. Тип: float

Алгоритм вирішення показано нижче

Введення даних:

Отримати значення epsilon, g та max_iter від користувача.

Ініціалізація:

Встановити $n = 1$ та $total_sum = 0$.

Цикл обчислень:

Обчислювати термін ряду:

Визначити чисельник як факторіал $(2n + 1)$.

Визначити знаменник за формулою.

Перевірити умови:

Якщо $|\text{термін}| < \text{epsilon}$, вивести суму та завершити.

Якщо $|\text{термін}| > g$, вивести повідомлення про розбіжність і завершити.

Додати термін до total_sum та інкрементувати n.

Перевірка ітерацій:

Якщо n перевищує max_iter, вивести повідомлення про досягнення максимального ліміту.

Завершення:

Вивести результати обчислень або повідомлення про стан ряду.

Лістинг коду вирішення задачі наведено в дод. А (стор.9). Екран роботи програми показаний на рис. Б.14.

Висновок

У цій лабораторній роботі були реалізовані алгоритми для вирішення завдань з розгалуженнями та циклами, що включають перевірку точок на потрапляння в геометричну область, дослідження рядів на збіжність, а також реалізацію меню для багаторазового виконання програм. Було успішно

застосовано структури розгалуження та цикли для вирішення інженерних завдань на Python, а також перевірено коректність введених даних і обробку помилок.

Додаток А

Лістинг коду програми до задачі (If) 19

```
def task_if19():
    """
    Завдання 1 (If19): Дано чотири цілих числа, одне з яких відрізняється від
    трьох інших рівних між собою.
    Визначити порядковий номер числа, відмінного від інших.
    """
    try:
        nums = [int(input(f"Введіть число {i+1}: ")) for i in range(4)]
        if nums.count(nums[0]) == 3:
            print("Відрізняється число 4 (останнє)")
        elif nums.count(nums[1]) == 3:
            print("Відрізняється число 1 (перше)")
        elif nums.count(nums[2]) == 3:
            print("Відрізняється число 2 (друге)")
        else:
            print("Відрізняється число 3 (третє)")
    except ValueError:
        print("Помилка: Введіть ціле число!")
```

Лістинг коду програми до задачі (Т.2.геом.обл.) 12

```
def is_in_yellow_area(x, y):
    return 0 <= x <= 1 and 0 <= y <= 1 # Задайте межі жовтої області

def task_geom_area(points):
    count_in_yellow_area = sum(1 for x, y in points if is_in_yellow_area(x, y))
    return count_in_yellow_area

try:
    n = int(input("Введіть кількість точок: ")) # Кількість точок
    points = []

    for i in range(n):
        x, y = map(float, input(f"Введіть координати точки {i+1} (x y): ").split())
        points.append((x, y))

    result = task_geom_area(points)
    print(f"Кількість точок, що потрапляють у жовту область: {result}")

except ValueError as e:
```

```
print(f"Помилка: {e}")
```

Лістинг коду програми до задачі Завдання 3 (Т.3.ряди) 11

```
import math

def factorial_series(epsilon=1e-5, g=1e5, max_iter=10000):
    n = 1
    total_sum = 0

    while n <= max_iter:
        # Обчислення чисельника (2n+1)!
        numerator = math.factorial(2 * n + 1)

        # Обчислення знаменника 1 * 4 * ... * (3n - 2)
        denominator = 1
        for i in range(1, n + 1):
            denominator *= (3 * i - 2)

        # Обчислюємо поточний член ряду
        term = numerator / denominator

        # Перевіряємо умови збіжності або розбіжності
        if abs(term) < epsilon:
            print(f'Ряд збігається на кроці {n}, терм: {term}')
            break
        if abs(term) > g:
            print(f'Ряд розбігається на кроці {n}, терм: {term}')
            break

        # Додаємо поточний член до суми
        total_sum += term
        n += 1

    return total_sum

# Приклад виклику функції
epsilon = 1e-10 # Можна змінити, наприклад, на 1e-20
g = 1e3         # Межа для розбіжності
result = factorial_series(epsilon, g)
print(f'Сума ряду: {result}')
```

Лістинг коду програми до задачі Завдання 4

```
from math import factorial

def calculate_series(epsilon, g, max_iter):
    n = 1
    total_sum = 0
```



```

while True:
    # Обчислення терміна ряду
    numerator = factorial(2 * n + 1)
    denominator = 1
    for i in range(1, n + 1):
        denominator *= (3 * i - 2)
    term = numerator / denominator

    # Умова зупинки
    if abs(term) < epsilon:
        print(f"Ряд збігається. Сума: {total_sum}")
        break
    if abs(term) > g:
        print("Ряд розбігається.")
        break

    total_sum += term
    n += 1

    if n > max_iter:
        print("Достигнуто максимального числа ітерацій.")
        break

def menu():
    while True:
        print("\nМеню:")
        print("1. Обчислити суму ряду")
        print("2. Вихід")

        choice = input("Оберіть опцію (1-2): ")

        if choice == '1':
            epsilon = float(input("Введіть epsilon (10^-5 до 10^-20): "))
            g = float(input("Введіть g (10^2 до 10^5): "))
            max_iter = int(input("Введіть максимальну кількість ітерацій (1000 до 10000): "))
            calculate_series(epsilon, g, max_iter)
        elif choice == '2':
            print("Вихід з програми.")
            break
        else:
            print("Неправильний вибір, будь ласка, спробуйте ще раз.")

# Викликаємо меню
menu()

```

ДОДАТОК Б

Скрін-шоти вікна виконання програми

```
task_if19()
Введіть число 1: 2
Введіть число 2: 3
Введіть число 3: 4
Введіть число 4: 5
Відрізняється число 3 (третє)
```

Рисунок Б.1 – Екран виконання програми для вирішення завдання
(If) 19

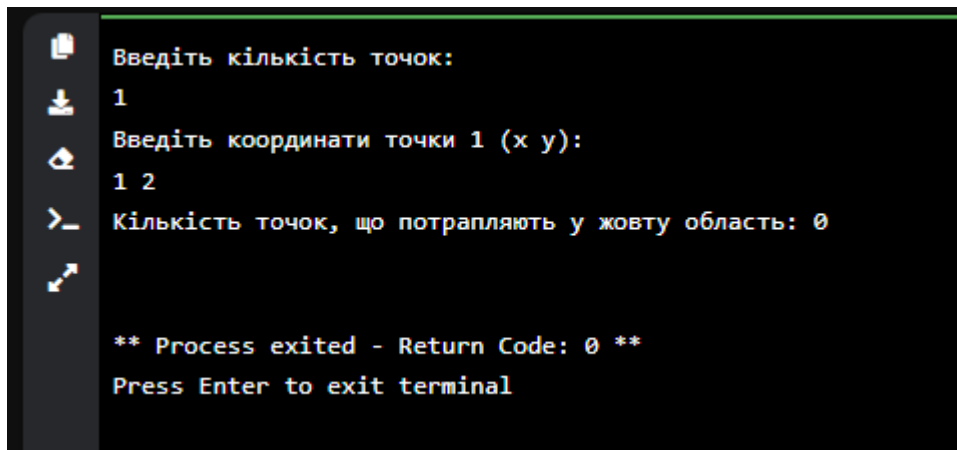


Рисунок Б.2 – Екран виконання програми для вирішення завдання
(Т.2.геом.обл.) 12

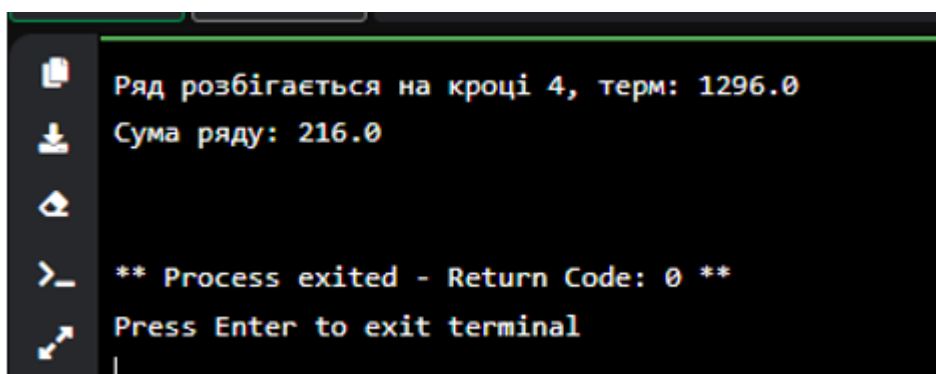
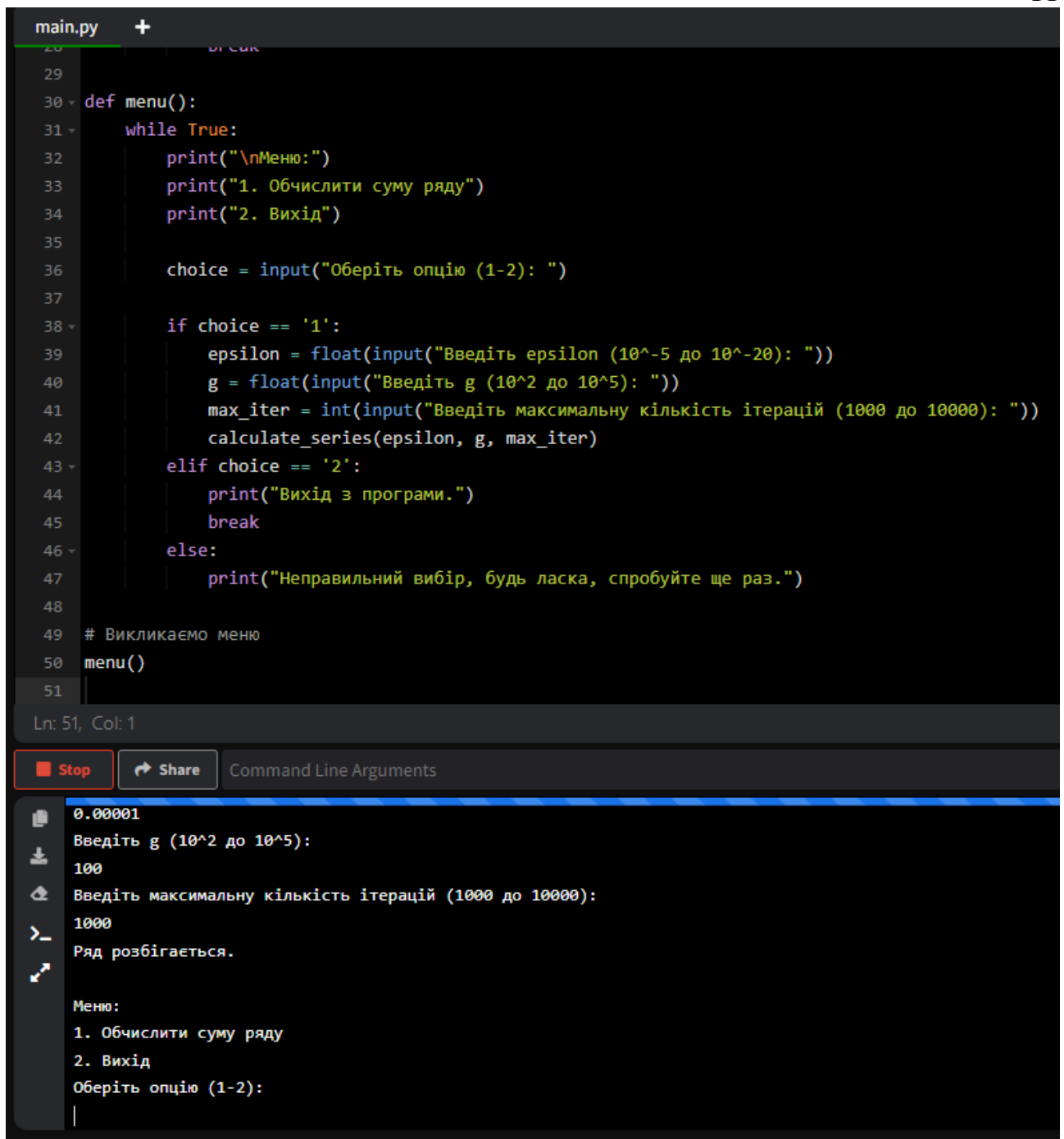


Рисунок Б.3 – Екран виконання програми для вирішення завдання
(Т.3.ряди) 11



```

main.py +
28
29
30 def menu():
31     while True:
32         print("\nМеню:")
33         print("1. Обчислити суму ряду")
34         print("2. Вихід")
35
36         choice = input("Оберіть опцію (1-2): ")
37
38         if choice == '1':
39             epsilon = float(input("Введіть epsilon (10^-5 до 10^-20): "))
40             g = float(input("Введіть g (10^2 до 10^5): "))
41             max_iter = int(input("Введіть максимальну кількість ітерацій (1000 до 10000): "))
42             calculate_series(epsilon, g, max_iter)
43         elif choice == '2':
44             print("Вихід з програми.")
45             break
46         else:
47             print("Неправильний вибір, будь ласка, спробуйте ще раз.")
48
49 # Викликаємо меню
50 menu()
51
Ln: 51, Col: 1
[Stop] [Share] Command Line Arguments
0.00001
Введіть g (10^2 до 10^5):
100
Введіть максимальну кількість ітерацій (1000 до 10000):
1000
Ряд розбігається.
Меню:
1. Обчислити суму ряду
2. Вихід
Оберіть опцію (1-2):
|

```

Рисунок Б.4 – Екран виконання програми для вирішення завдання 4