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 проектування СУ»

Тема: "Розробка структурованих програм з розгалуженням та повтореннями"

ХАІ.301. 173. 3-91АВ/С. 2 ЛР

Виконав студент гр. _____3-91АВ/С_____

_____Опанасюк Олександр___
(підпис, дата) (П.І.Б.)

Перевірив
_____ к.т.н., доц. О. В. Гавриленко
(підпис, дата) (П.І.Б.)

2024

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 представлено в табл.1.

Завдання 2. Дано дійсні числа (xi, yi), i = 1,2, ... n, – координати точок на площині. Визначити кількість точок, що потрапляють в геометричну область заданого кольору (або групу областей). Варіанти геометричних областей представлені в табл.2.

Завдання 3. Дослідити ряд на збіжність. Умова закінчення циклу обчислення суми прийняти у вигляді: | un | <E або | un | > G де e – мала величина для переривання циклу обчислення суми сходиться ряду (e = 10-5 ... 10-20); g – величина для переривання циклу обчислення суми розходиться ряду (g = 102 ...105). Варіанти представлено в табл.3.

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. Перевірка числа

Вхідні дані:

int_num – ціле число, яке вводить користувач;

Тип: int; допустимі значення: будь-які цілі числа

Вихідні дані

int_num результат після перевірки та обробки введеного числа;

Тип: int.

Текстове повідомлення – результат обчислення, або сповіщення про помилку вводу.

Алгоритм вирішення показано нижче:

- Користувач вводить ціле число.
- Програма перевіряє, чи є введене число додатним.
- Якщо так, віднімається 8.
- Інакше число залишається незмінним.
- Результат виводиться на екран.
- Якщо введені дані некоректні, програма виводить текстове повідомлення про помилку.

```
4   def task_if1():
5       """
6       Завдання 1: Перевірка цілого числа.
7       Якщо число додатне, відняти 8; інакше залишити без змін.
8       """
9       try:
10          int_num = int(input("Введіть ціле число: "))
11          if int_num > 0:
12              int_num -= 8
13          print("Результат:", int_num)
14      except ValueError:
15          print("Помилка: потрібно ввести ціле число.")
```

Рисунок 1 – Алгоритм вирішення завдання 1

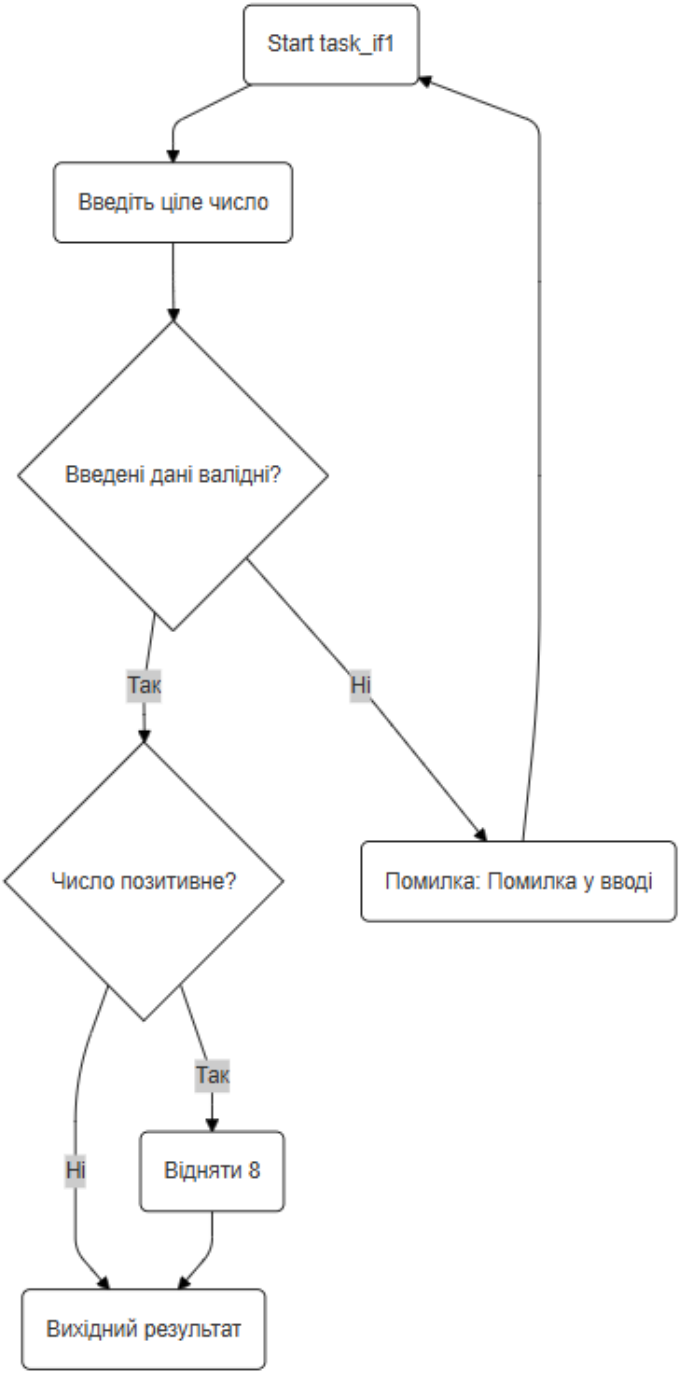

Рисунок 1.1 - Діаграма діяльності завдання 1

Завдання 2. Визначення точок у заданій області.

Вхідні дані:

points – список координат точок на площині;

Тип: list[tuple[float, float]]; допустимі значення: будь-які дійсні числа.

a – висота прямокутника;

Тип: float; допустимі значення: a > 0.

b – ширина прямокутника;

Тип: float; допустимі значення: b > 0.

Вихідні дані:

count_variant1 – кількість точок, які належать першій області (коричнева частина);

Тип: int.

count_variant2 – кількість точок, які належать другій області (салатова частина);

Тип: int.

Текстові повідомлення – результат перевірки кількості точок у кожній області.

Алгоритм вирішення показано нижче:

- Програма приймає список координат точок та параметри області (a, b).
- Для кожної точки перевіряється її належність до:
- Області 1: точка лежить у прямокутнику та в колі з центром (b/2, 0) і радіусом b/2.
- Області 2: точка лежить у прямокутнику та за межами кола з тим самим центром і радіусом.
- Рахуються кількість точок у кожній області.
- Результати виводяться на екран.

```python
def task_points_in_area(points, a, b):
    """
    Завдання 2: Визначення точок у заданій області.
    :param points: список координат точок [(x1, y1), (x2, y2), ...]
    :param a: висота прямокутника
    :param b: ширина прямокутника
    :return: кількість точок у заданій області
    """
    count_variant1 = 0
    count_variant2 = 0

    for x, y in points:
        # Перевірка на належність до варіанту 1 (коричнева область)
        if 0 <= x <= b and 0 <= y <= a and (x - b / 2) ** 2 + y ** 2 <= (b / 2) ** 2:
            count_variant1 += 1

        # Перевірка на належність до варіанту 2 (салатова область)
        if 0 <= x <= b and 0 <= y <= a and (x - b / 2) ** 2 + y ** 2 >= (b / 2) ** 2:
            count_variant2 += 1

    print("Кількість точок у варіанті 1:", count_variant1)
    print("Кількість точок у варіанті 2:", count_variant2)
```

Рисунок 2 – Визначення точок у заданій області

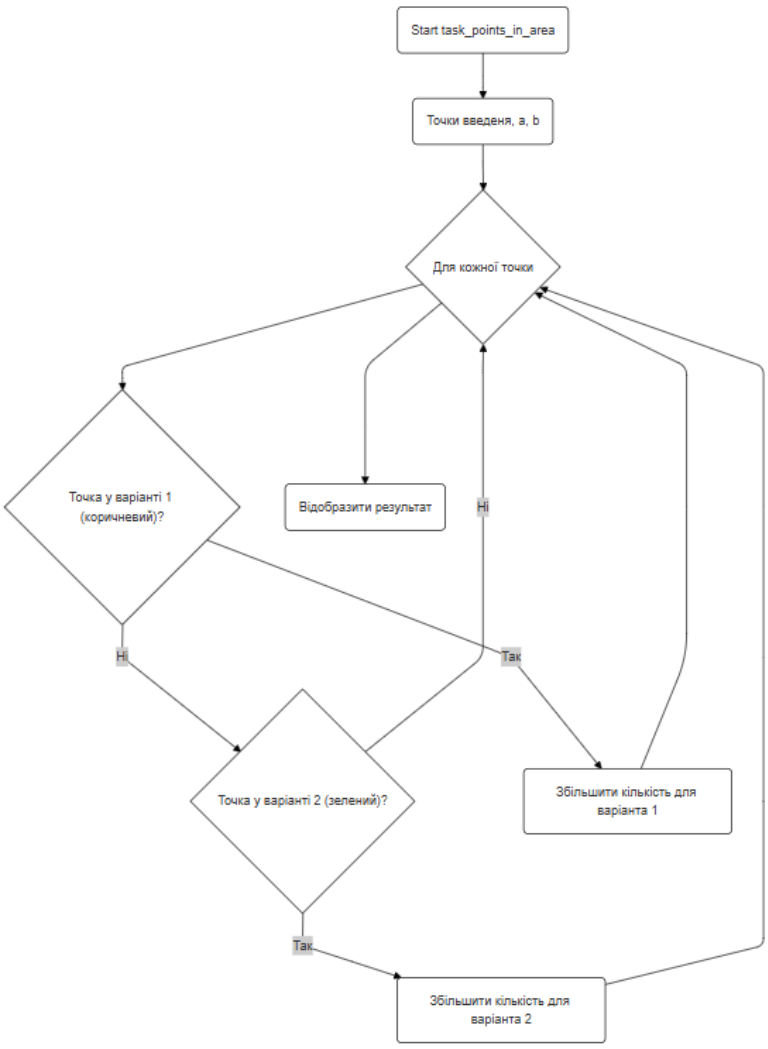

Рисунок 2.1 – Діграмма діяльності завдання 2

Завдання 3. Дослідження ряду на збіжність.

Вхідні дані:
e – мала величина для зупинки обчислення збіжного ряду;
Тип: float; допустимі значення: e = 10^-5 ... 10^-20.
g – велика величина для зупинки обчислення розбіжного ряду;
Тип: float; допустимі значення: g = 10^2 ... 10^5.

Вихідні дані:
s – сума ряду після виконання обчислень;
Тип: float.
Текстові повідомлення:
"Ряд збігається. Сума: ..." – якщо ряд збігся.
"Ряд розходиться. Часткова сума: ..." – якщо ряд розійшовся.
"Ділення на нуль." – якщо виникла помилка при обчисленнях.

Алгоритм вирішення показано нижче:
- Ініціалізація значень: n = 1, s = 0.
- Виконується обчислення членів ряду за формулою:

$$u_n \frac{n\sqrt{n}}{(n/2)^n}$$

- Перевіряється умова зупинки:
  Якщо |u_n| < e, ряд вважається збіжним.
  Якщо |u_n| > g, ряд вважається розбіжним.
- Значення суми (s) оновлюється з кожною ітерацією.
- У разі помилки (наприклад, ділення на нуль), виводиться текстове повідомлення.

```python
def task_series1(e=1e-5, g=1e5):
    """
    Завдання 3: Дослідження ряду на збіжність.
    |un| < e або |un| > g для зупинки.
    """
    n = 1
    s = 0  # Сума ряду
    while True:
        try:
            u = (n * math.sqrt(n)) / ((n / 2) ** n)
            if abs(u) < e:
                print("Ряд збігається. Сума:", s)
                break
            if abs(u) > g:
                print("Ряд розходиться. Часткова сума:", s)
                break
            s += u
            n += 1
        except ZeroDivisionError:
            print("Ділення на нуль.")
            break
```

Рисунок 3 – Дослідження ряду на збіжність

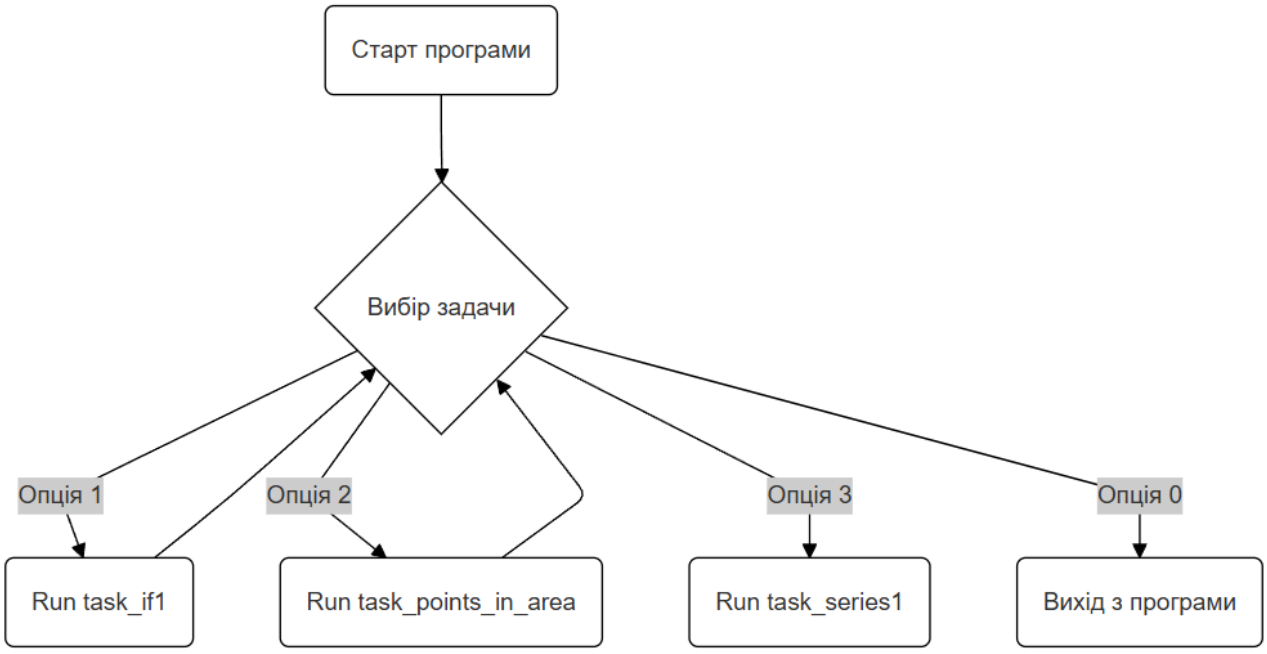

Рисунок 4 – Діаграма діяльності всієї програми

ВИСНОВКИ

У ході виконання роботи було вивчено основні алгоритми для вирішення задач із розгалуженням, перевірки належності точок до геометричних областей та дослідження збіжності рядів. Закріплено на практиці навички розробки функцій у Python з обробкою винятків, а також організації циклічного меню для багаторазового виконання задач. У результаті роботи відпрацьовано вміння структурувати код, використовувати модулі та забезпечувати коректність введення й обчислень.

Додаток А

Лістинг коду програми до задач №1: Перевірка числа, №2: Визначення точок у заданій області, №3: Дослідження ряду на збіжність, №4: Циклічний алгоритм організації меню

main.py
<
```
import m

def main():
    while True:
        print("\nМеню:")
        print("1. Завдання 1: Перевірка числа")
        print("2. Завдання 2: Точки в області")
        print("3. Завдання 3: Дослідження ряду")
        print("0. Вихід")
        try:
            choice = int(input("Виберіть задачу (0 для виходу): "))
            if choice == 0:
                print("До побачення!")
                break
            elif choice == 1:
                m.task_if1()
            elif choice == 2:
                # Задати координати точок та параметри області
                points = [(1, 1), (2, 2), (3, 3)]  # Введіть свої точки
                a = float(input("Введіть висоту прямокутника a: "))
                b = float(input("Введіть ширину прямокутника b: "))
                m.task_points_in_area(points, a, b)
            elif choice == 3:
                e = float(input("Введіть значення e (мала величина): "))
                g = float(input("Введіть значення g (величина для розходження): "))
                m.task_series1(e, g)
            else:
                print("Невірний вибір. Спробуйте ще раз.")
        except ValueError:
            print("Помилка: потрібно ввести число.")

if __name__ == "__main__":
    main()
```
>

m.py
<

```
import math


def task_if1():
    """
    Завдання 1: Перевірка цілого числа.
    Якщо число додатне, відняти 8; інакше залишити без змін.
    """
    try:
        int_num = int(input("Введіть ціле число: "))
        if int_num > 0:
            int_num -= 8
        print("Результат:", int_num)
    except ValueError:
        print("Помилка: потрібно ввести ціле число.")


def task_points_in_area(points, a, b):
    """
    Завдання 2: Визначення точок у заданій області.
    :param points: список координат точок [(x1, y1), (x2, y2), ...]
    :param a: висота прямокутника
    :param b: ширина прямокутника
    :return: кількість точок у заданій області
    """
    count_variant1 = 0
    count_variant2 = 0

    for x, y in points:
        # Перевірка на належність до варіанту 1 (коричнева область)
        if 0 <= x <= b and 0 <= y <= a and (x - b / 2) ** 2 + y ** 2 <= (b / 2) **
2:
            count_variant1 += 1

        # Перевірка на належність до варіанту 2 (салатова область)
        if 0 <= x <= b and 0 <= y <= a and (x - b / 2) ** 2 + y ** 2 >= (b / 2) **
2:
            count_variant2 += 1

    print("Кількість точок у варіанті 1:", count_variant1)
    print("Кількість точок у варіанті 2:", count_variant2)


def task_series1(e=1e-5, g=1e5):
    """
    Завдання 3: Дослідження ряду на збіжність.
    |un| < e або |un| > g для зупинки.
    """
    n = 1
```

```
    s = 0  # Сума ряду
    while True:
        try:
            u = (n * math.sqrt(n)) / ((n / 2) ** n)
            if abs(u) < e:
                print("Ряд збігається. Сума:", s)
                break
            if abs(u) > g:
                print("Ряд розходиться. Часткова сума:", s)
                break
            s += u
            n += 1
        except ZeroDivisionError:
            print("Ділення на нуль.")
            break
>
```

ДОДАТОК Б

Скрін-шоти вікна виконання програми

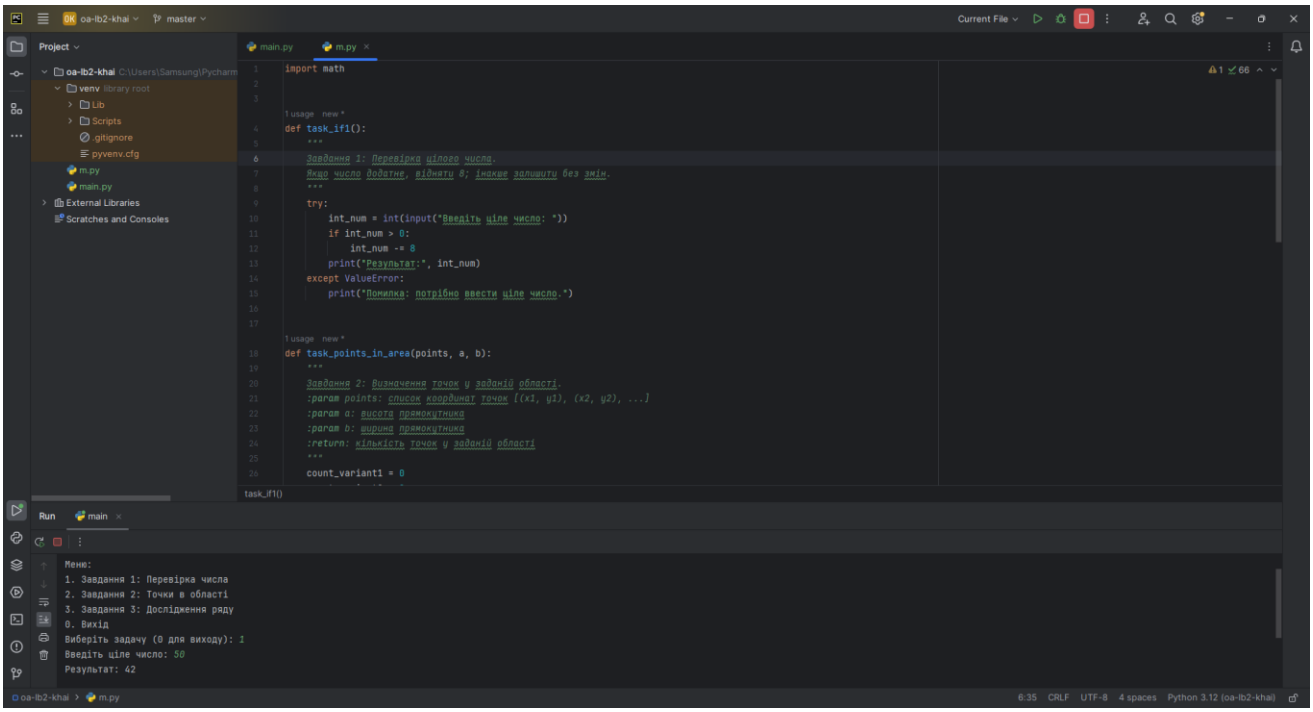

Рисунок Б.1 – Екран виконання програми для вирішення завдання №1: Перевірка числа

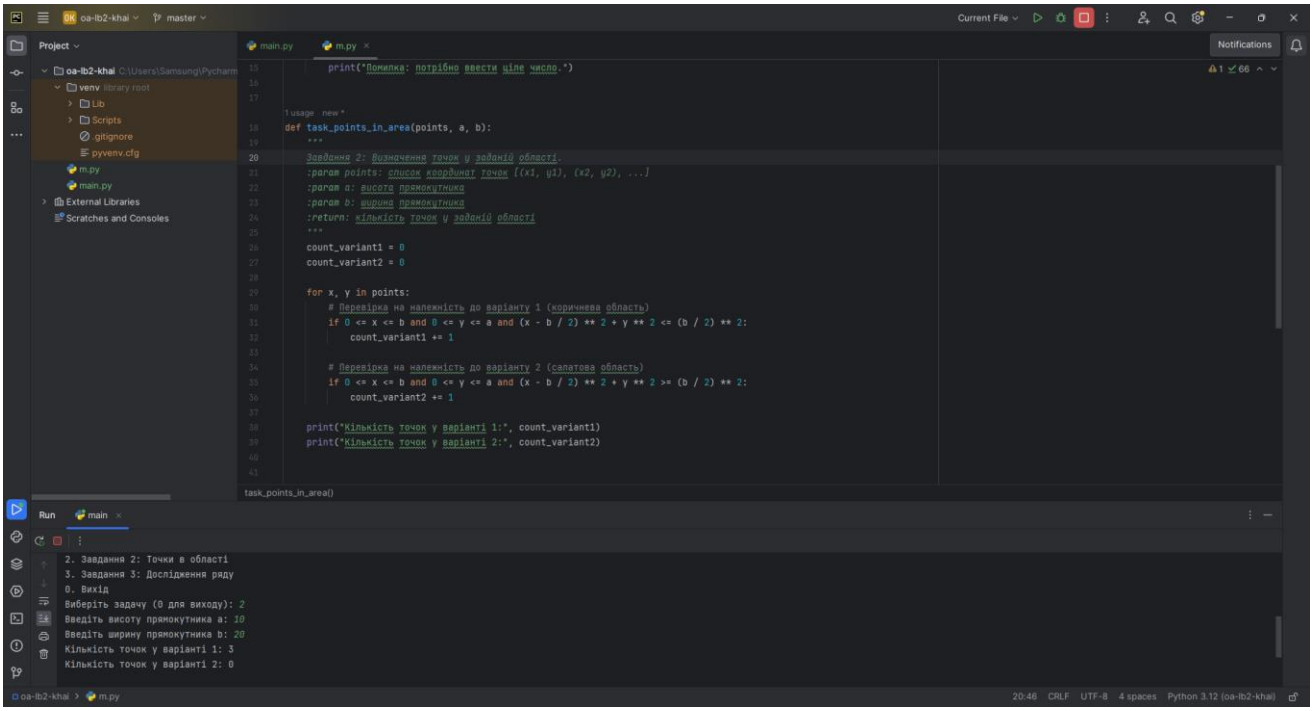

Рисунок Б.2 – Екран виконання програми для вирішення завдання №2: Визначення точок у заданій області

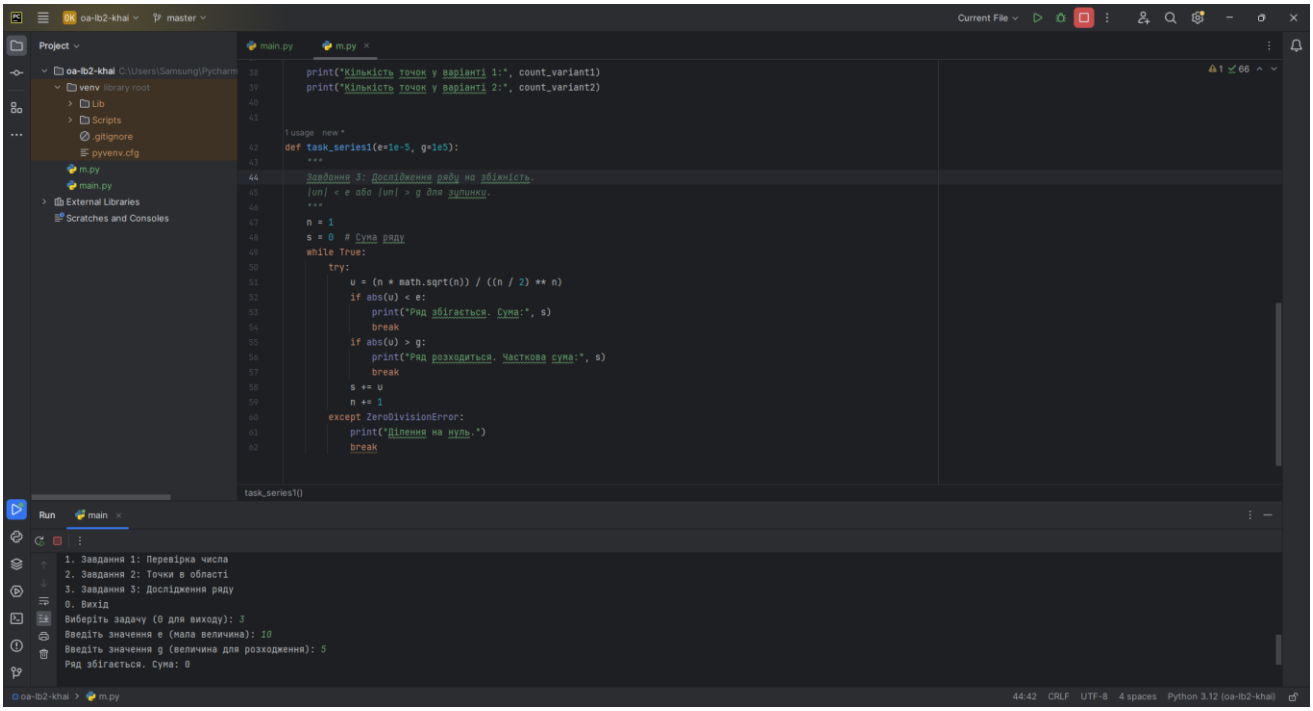

Рисунок Б.3 – Екран виконання програми для вирішення завдання №3: Дослідження ряду на збіжність

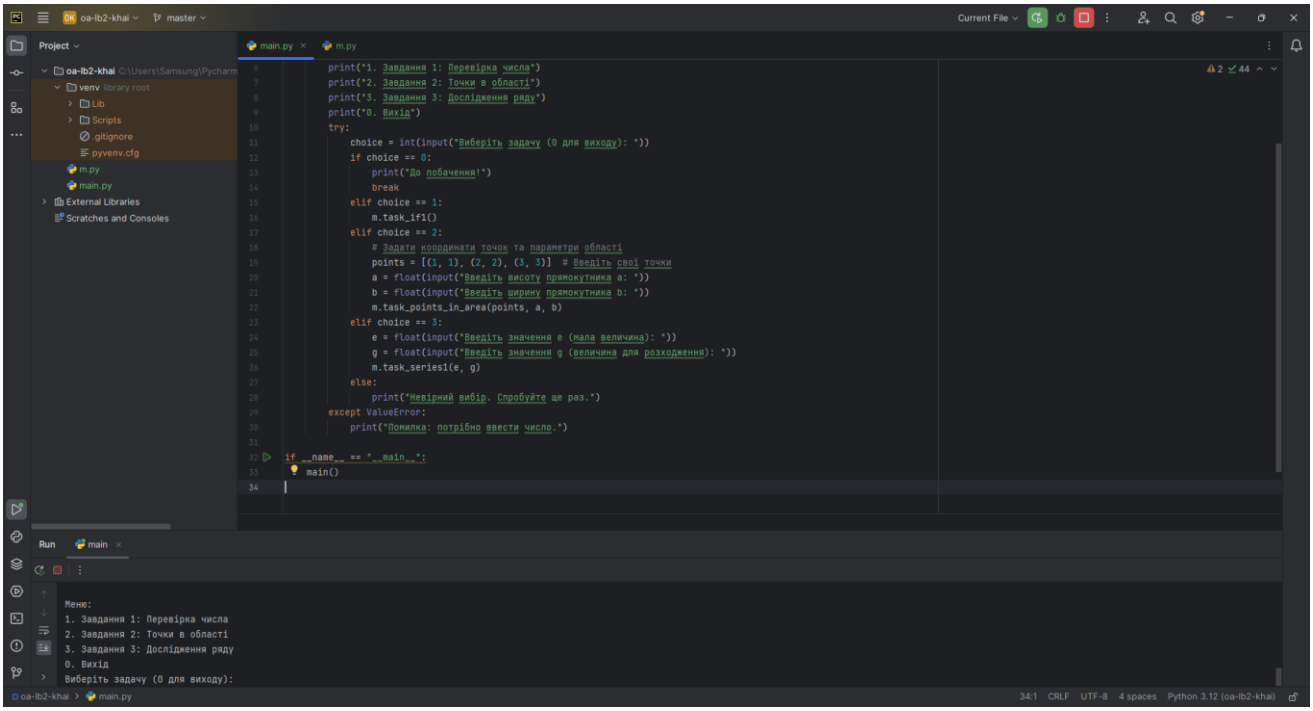

Рисунок Б.4 – Екран виконання програми для вирішення завдання №4: Циклічний алгоритм організації меню