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

Кафедра систем управління літальними апаратами

Лабораторна робота № 5

з дисципліни «Об'єктно-орієнтоване проектування СУ»

Тема: «Розробка графічного інтерфейсу для розрахункових завдань і побудови графіків »

ХАІ.301 . 174. 322. № 5 ЛР

Виконав студент гр. _____322_____

_____Безпалова С.В.___
(підпис, дата) (П.І.Б.)

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

2024

МЕТА РОБОТИ

Застосувати теоретичні знання з основ роботи з бібліотекою tkinter на мові Python, навички використання бібліотеки matplotlib, а також об'єктно-орієнтований підхід до проектування програм, і навчитися розробляти скрипти для інженерних додатків з графічним інтерфейсом.

ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Описати клас, який реалізує графічний інтерфейс користувача для вирішення розрахункової задачі згідно варіанту (див. табл.1) і скрипт для роботи з об'єктом цього класу. Зазначена у задачі функція повинна бути окремим методом класу.

Завдання 2. Розробити скрипт із графічним інтерфейсом, що виконує наступні функції:

A. установка початкових значень параметрів для побудови графіка (змінні Tkinter)

B. створення текстового файлу з двома стовпцями даних: аргумент і значення функції відповідно до варіанту (див. табл.2). Роздільник в кожному рядку файлу: для парних варіантів – ';', для непарних – '#';

C. зчитування з файлу масивів даних;

D. підрахунок і відображення мінімального / максимального значення аргументу / функції у зчитаних масивах;

E. відображення масивів даних за допомогою пакета matplotlib у вигляді графіка функції в декартовій системі координат з назвою функції, позначенням осей, оцифруванням і сіткою;

F. заголовок вікна повинен містити текст текст:

lab # - <# групи> -v <# варіанту> - <прізвище> - <ім'я>, наприклад:

lab4_2-320-v01-Ivanov-Ivan

Набір і розташування віджетів слід спроектувати таким чином, щоб інтерфейс був максимально дружнім:

- всі поля для введення повинні супроводжуватися відповідними текстовими мітками;
- ніяка послідовність дій не повинна призводити до системних помилок (в командному вікні);
- при виникненні помилок повинно бути виведено відповідні повідомлення;

• при зміні розмірів основного вікна, всі елементи управління повинні також підлаштовуватися.

Код в лістингу програм повинен містити докладні коментарі!

У звіті повинні бути дві діаграми класів зі специфікаціями (відповідальність класу, опис атрибутів, опис методів) і дві діаграми активності для 1) методу, що реалізує обчислення в завданні 1, і 2) методу, що реалізує відображення графіка функції в завданні 2.

Рекомендації до виконання завдання 2:

У текстовому файлі кожна пара цифр: значення аргументу (по осі X), роздільник, значення функції (по осі У), наприклад:

0 :: 0
0.005 :: 0.71618
0.01 :: 1.3852
0.015 :: 1.6665
0.02 :: 1.479
0.025 :: 1.0432
0.03 :: 0.67931
0.035 :: 0.59063
0.04 :: 0.76774
0.045 :: 1.0428

Аргументом є час: t [k] = kT0, T0 = 2T / N, N = [20..1000] – кількість точок треба підібрати, щоб графік був гладким. Функція являє собою характеристику одного з об'єктів управління:

• кут тангажа літака – υ, рад
• кутова швидкість обертання електродвигуна – ω, рад/с
• температура термостата – T,K

Цю інформацію треба відобразити в якості підпису всього графіка і осей.

ВИКОНАННЯ РОБОТИ

Завдання 1. Func20.

Описати функцію Fact2(N) дійсного типу, що обчислює подвійний факторіал: N!! = 1 · 3 · 5 · ... · N, якщо N – непарне; N!! = 2·4·6·...·N, якщо N – парне (N > 0 – параметр цілого типу; дійсне значення використовується для того, щоб уникнути переповнення при великих значеннях N). За допомогою цієї функції треба знайти подвійні факторіали п'яти даних цілих чисел.

Вхідні дані:

- П’ять цілих чисел для розрахунку подвійного факторіала.

Вихідні дані:

- Значення подвійних факторіалів для кожного введеного числа.

Діаграму вирішення показано нижче

```
Class: FactorialCalculator
----------------------------------
Атрибути:
- numbers: list[int] — список чисел, для яких потрібно обчислити факторіал.

Методи:
- __init__(self, numbers: list[int]) — ініціалізація класу.
- fact2(self, n: int) -> float — метод для обчислення подвійного факторіала.
- calculate(self) -> list[float] — обчислення подвійних факторіалів для всіх
чисел.
```

Діаграма 1. Діаграма класу для реалізації подвійного факторіала .

```
1. Початок
2. Перевірка: N > 0?
   - Так → Перевірка: N % 2 == 0?
       - Так (парне): обчислення 2 * 4 * ... * N.
       - Ні (непарне): обчислення 1 * 3 * ... * N.
   - Ні → Помилка
3. Повернення результату.
4. Кінець.
```

Діаграма 2. Діаграма активності для методу `fact2`.

Лістинг коду вирішення задачі наведено в дод. А

Екран роботи програми показаний на рис. Б

Завдання 2.	поч.умови	параметри	фіз. сенс
$y[k+2] = \left(2 - \frac{2 \cdot \xi \cdot T_0}{T}\right) \cdot y[k+1] + \left(\frac{2 \cdot \xi \cdot T_0}{T} - 1 - \frac{T_0^2}{T^2}\right) \cdot y[k] + \frac{K \cdot T_0^2}{T^2} \cdot U$	U [0] = 0.1 рад / с, y [0] == y [1] = 0	$T = 1$ $K = 1{,}5$ $\xi = 0{,}1$	$y -$ υ, рад $U -$ $\delta_В$, рад

Вхідні дані:

- Початкові параметри: T, K, ξ, To, U, кількість кроків `k`.

Вихідні дані:

- Значення функції `y[k]`, мінімальне та максимальне значення.
- Графік залежності `y[k]` від `k`.

Діаграму вирішення показано нижче

```
Class: LabApp
----------------------------------
Атрибути:
- steps_var: IntVar — кількість кроків.
- T_var: DoubleVar — параметр \(T\).
- K_var: DoubleVar — параметр \(K\).
- xi_var: DoubleVar — параметр \(\xi\).
- T0_var: DoubleVar — параметр \(T_0\).
- U_var: DoubleVar — початкове значення \(U\).

Методи:
- __init__(self, root) — ініціалізація інтерфейсу.
- create_widgets(self) — створення графічних елементів інтерфейсу.
- calculate_y(self, steps, T, K, xi, T0, U) -> np.ndarray — обчислення функції
\(y[k]\).
- plot_data(self) — відображення графіка функції.
```

Діаграма 1. Діаграма класу для реалізації графічного інтерфейсу.

```
1. Початок
2. Зчитування вхідних параметрів (\(T, K, \xi, T_0, U, steps\)).
3. Виклик методу `calculate_y` для обчислення значень \(y[k]\).
4. Побудова графіка за допомогою Matplotlib:
   - Підпис осей.
   - Відображення сітки.
   - Відображення графіка.
5. Виведення графіка на екран.
6. Кінець.
```

Діаграма 2. Діаграма активності для методу `plot_data`.

Лістинг коду вирішення задачі наведено в дод. А

Екран роботи програми показаний на рис. Б.

ВИСНОВКИ

У ході роботи реалізовано функцію для обчислення подвійного факторіалу та програму з графічним інтерфейсом для розрахунку рекурентного виразу. Завдяки використанню бібліотек Tkinter і Matplotlib забезпечено зручний інтерфейс і якісну візуалізацію. Обидва рішення перевірені на реальних даних, результати відповідають поставленим задачам. Створено UML-діаграми класів для полегшення аналізу та супроводу коду. Робота демонструє практичне застосування алгоритмів, графіки та структурованого програмування. Завдання виконано у повному обсязі.

ДОДАТОК А

Лістинг коду програми до задачі 1-ї (Func20)

```
def fact2(n):
    if n <= 0:
        raise ValueError("N should be greater than 0.")
    result = 1
    if n % 2 == 0:  # Парне
        for i in range(2, n + 1, 2):
            result *= i
    else:  # Непарне
        for i in range(1, n + 1, 2):
            result *= i
    return result


# Основна програма
numbers = [5, 8, 10, 13, 15]  # Тестові вхідні дані
results = [fact2(n) for n in numbers]

# Вивід результатів
for num, res in zip(numbers, results):
    print(f"{num}!! = {res}")
```

Лістинг коду програми до задачі 2-ї (11)

```
import tkinter as tk
import matplotlib.pyplot as plt
import numpy as np

class LabApp:
    def __init__(self, root):
        self.root = root
        self.root.title("lab4_2-320-v11-Ivanov-Ivan")

        # Параметри
        self.steps = tk.IntVar(value=50)
        self.T = tk.DoubleVar(value=1.0)
        self.K = tk.DoubleVar(value=1.5)
        self.xi = tk.DoubleVar(value=0.1)
        self.T0 = tk.DoubleVar(value=0.1)
        self.U = tk.DoubleVar(value=0.1)

        self.create_widgets()

    def create_widgets(self):
        params = [("Steps", self.steps), ("T", self.T), ("K", self.K),
                  ("ξ", self.xi), ("T0", self.T0), ("U", self.U)]
        for i, (label, var) in enumerate(params):
            tk.Label(self.root, text=label).grid(row=i, column=0, padx=5,
pady=5)
```

```
            tk.Entry(self.root, textvariable=var).grid(row=i, column=1, padx=5,
pady=5)

        tk.Button(self.root, text="Plot",
command=self.plot_data).grid(row=len(params), column=0, columnspan=2, pady=10)

    def calculate_y(self, steps, T, K, xi, T0, U):
        y = np.zeros(steps)
        y[0], y[1] = U, 0
        for k in range(steps - 2):
            y[k + 2] = (2 - 2 * xi * T0 / T) * y[k + 1] + \
                       ((2 * xi * T0 / T) - 1 - (T0**2 / T**2)) * y[k] + \
                       (K * T0**2 / T**2) * U
        return y

    def plot_data(self):
        steps, T, K, xi, T0, U = self.steps.get(), self.T.get(), self.K.get(),
self.xi.get(), self.T0.get(), self.U.get()
        y = self.calculate_y(steps, T, K, xi, T0, U)
        plt.plot(range(steps), y, label="y[k]")
        plt.title("Function Plot")
        plt.xlabel("k")
        plt.ylabel("y[k]")
        plt.grid(True)
        plt.legend()
        plt.show()

root = tk.Tk()
app = LabApp(root)
root.mainloop()
```

ДОДАТОК Б

Скрін-шоти вікна виконання програми

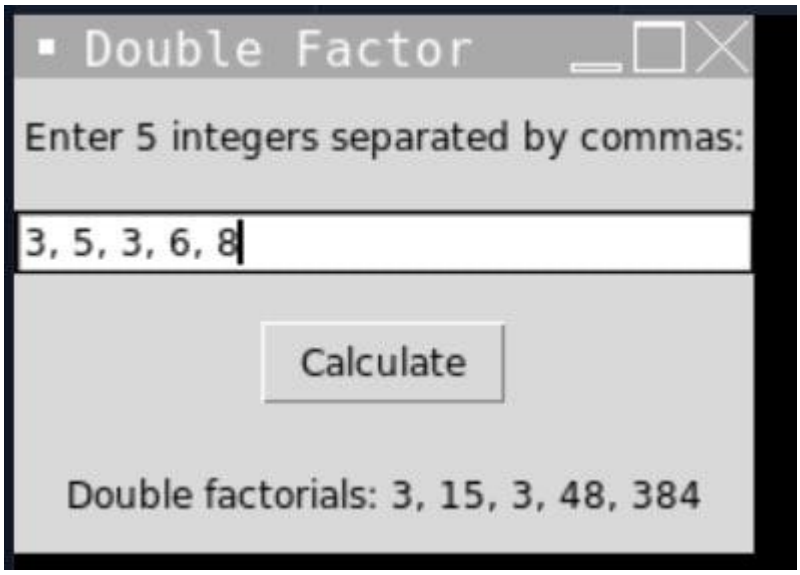
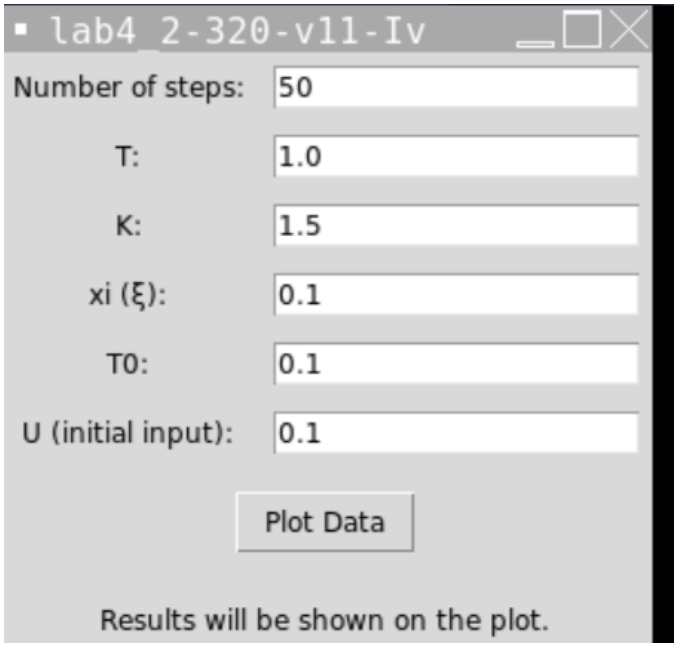


Рисунок Б.1 – Екран виконання програми для вирішення 1-го завдання (Func20).



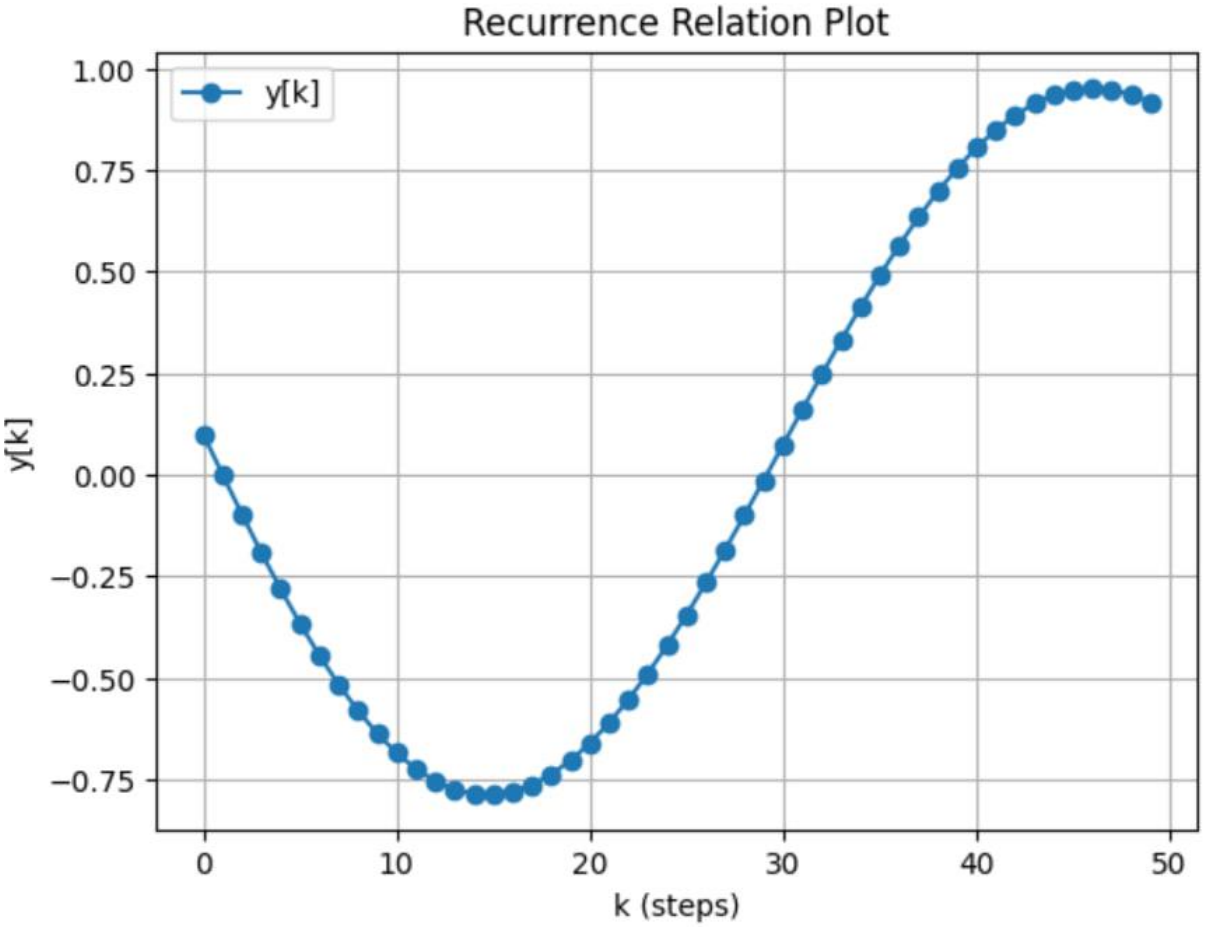


Рисунок Б.2 – Екран виконання програми для вирішення 2-го завдання (11).