

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний аерокосмічний університет  
«Харківський авіаційний інститут»

Кафедра систем управління літальних апаратів

## Практична робота №5

з дисципліни «Об'єктно-орієнтоване проєктування СУ»

Тема: «Реалізація класу і робота з об'єктами»

XAI.301. 175. 328. 08 ПР

Виконав студент гр. \_\_\_\_\_ 328

Глєбова Каріна

(підпис, дата)

(П.І.Б.)

Перевірив

к.т.н., доц. О. В. Гавриленко

(підпис, дата)

(П.І.Б.)

2025

## 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.

**Funk 29.** Описати функцію `AddRightDigit(D, K)`, яка додає до цілого додатного числа K праворуч цифру D і повертає отримане число (D і K – параметри цілого типу, причому D лежить у діапазоні 0–9). За допомогою цієї функції послідовно додати до даного числа K праворуч дані цифри D1 і D2, виводячи результат кожного додавання.

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

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

Б. 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 – '#';

С. зчитування з файлу масивів даних;

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

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

Ф. заголовок вікна повинен містити текст текст:

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

`lab4_2-320-v01-Ivanov-Ivan`

Task 2. На рис. 1 показані дані для побудови графіка для варіанту 2.

$y[k + 2] = \left(2 - \frac{2 \cdot \xi \cdot T_0}{T}\right) \cdot y[k + 1] + \left(\frac{2 \cdot \xi \cdot T_0}{T} - 1 - \frac{T_0^2}{T^2}\right) \cdot y[k] + \frac{K \cdot T_0^2}{T^2} U$	$U[0] = 0.1 \text{ рад / с,}$ $y[0] = y[1] = 0$	$T = 0,1$ $K = 3$ $\xi = 0,2$	$y - v, \text{рад}$ $U - U, \text{рад}$ $\delta_B, \text{рад}$
--	--	-------------------------------------	--

Рисунок 1 – Дані для побудови графіка для варіанту 2

Завдання 3. Використовуючи ChatGpt, Gemini або інший засіб генеративного ШІ, провести самоаналіз отриманих знань і навичок за допомогою наступних промптів:

1. «Ти - викладач, що приймає захист моєї роботи. Задай мені тестових питань з 4 варіантами відповіді і 5 відкритих питань. Це мають бути завдання <середнього> рівня складності на розвиток критичного та інженерного мислення. Питання мають відноситись до коду, що є у файлі звіту, і до теоретичних відомостей, що є у <презентації> лекції»
2. «Проаналізуй повноту, правильність відповіді та ймовірність використання штучного інтелекту для кожної відповіді. Оціни кожне питання у 5-балльній шкалі, віднімаючи 60% балів там, де ймовірність відповіді з засобом ШІ висока. Обчисли загальну середню оцінку»
3. «Проаналізуй код у звіті, і додай опис і приклади коду з питань, які є в теоретичних відомостях, але не реалізовано у коді при вирішенні завдань»

Проаналізуйте задані питання, коментарі і оцінки, надані ШІ. Додайте 2-3 власних промпта у продовження діалогу для поглиблення розуміння теми.

## ВИКОНАННЯ РОБОТИ

Завдання 1.

Вирішення задачі Func 29.

Опис функції AddRightDigit(D, K).

Вхідні дані (ім'я, опис, тип, обмеження):

K – перше число для вводу, до якого будуть додаватися цифри; тип int, ціле число;

D1 – перша цифра, яка буде додана праворуч до числа K; тип int, ціле число; може бути лише в межах від 0 до 9;

D2 – друга цифра, яка буде додана праворуч до результату після додавання D1; тип int, ціле число; може бути лише в межах від 0 до 9.

Вихідні дані (ім'я, опис, тип):

res1 – результат додавання цифри D1 праворуч до числа K;

res2 – результат додавання цифри D2 праворуч до res1.

Діаграма активності для завдання task1.py показана на рис. 2.

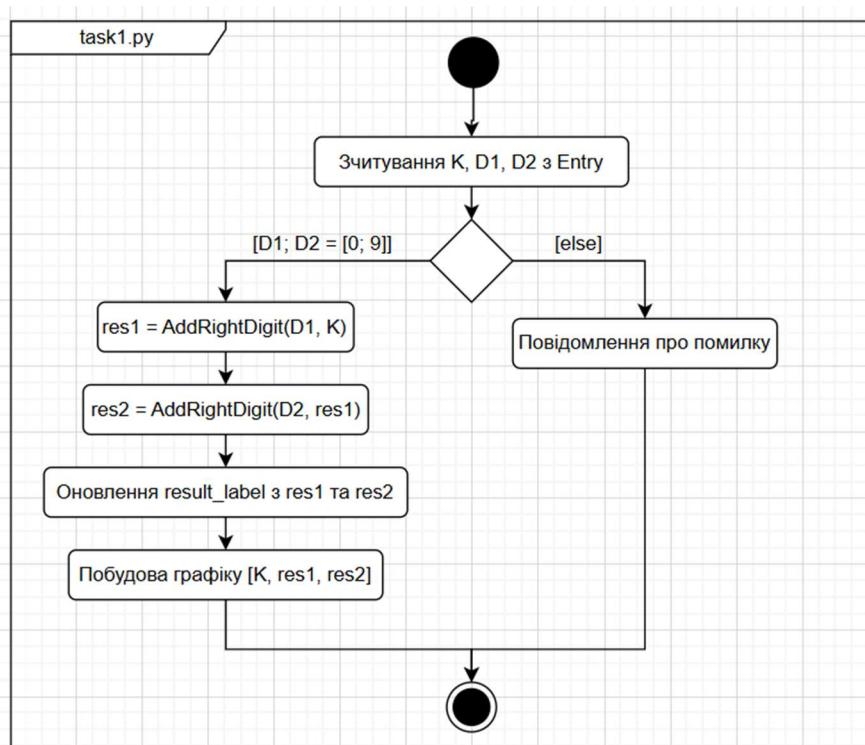


Рисунок 2 – Діаграма активності для завдання task1.py

Лістинг коду вирішення задачі task1.py наведено в дод. А (стор. 9-12).

Екран роботи програми показаний на рис. Б.1 (дод. Б, стор. 13-15)

Завдання 2.

Вирішення задачі Task 2.

Розроблення скрипту із графічним інтерфейсом.

Вихідні дані (ім'я, опис, тип):

N – кількість введених точок для побудови графіка; тип int, ціле число; може бути в межах від 20 до 1000;

Вихідні дані:

lab5\_data.txt – текстовий файл, де позначені час та значення через крапку з комою (бо варіант парний);

min\_x, max\_x – мінімальне та максимальне значення часу, автоматично визначається з масиву x; тип float, число з плаваючою крапкою.

min\_y, max\_y – мінімальне та максимальне значення функції y, залежить від графіку; тип float, число з плаваючою крапкою;

plt.plot(x, y) – графік залежності кута тангажу літака від часу, будується лише після вдалих обчислень та не зберігається автоматично.

Діаграма активності завдання task2.py показана на рис. 3.

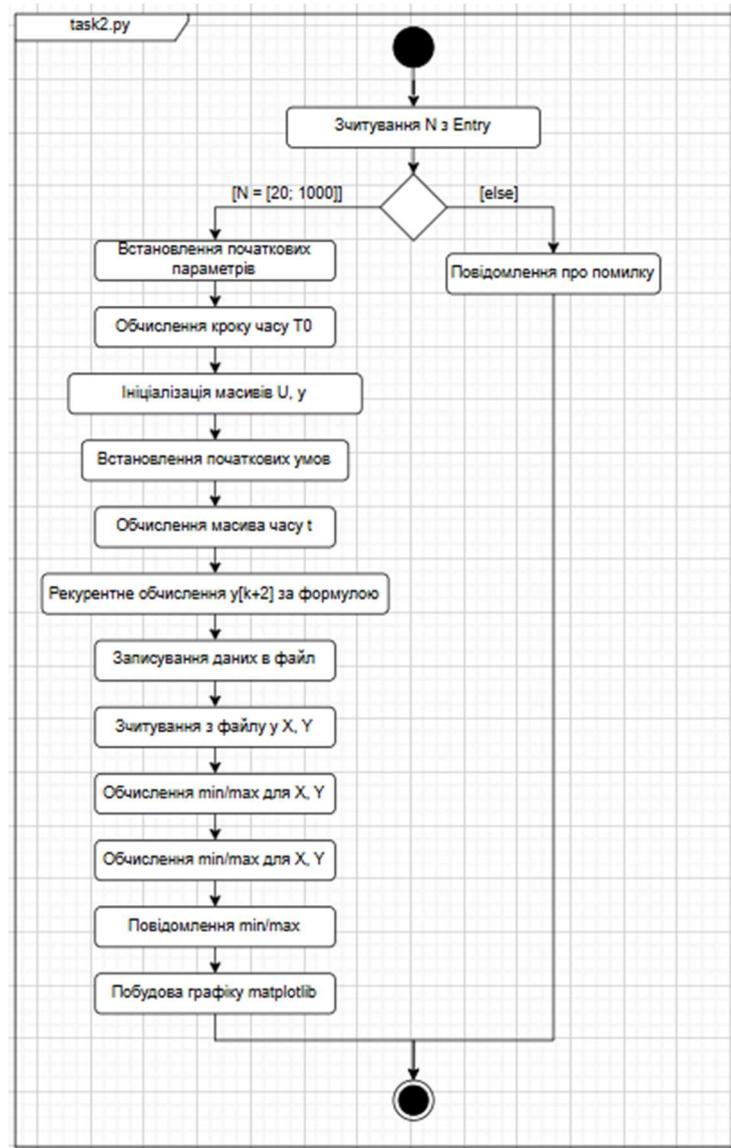


Рисунок 3 – Діаграма активності головної функції task2.py

Лістинг коду вирішення задачі task2.py наведено в дод. А (стор. 9-12).

Екран роботи програми показаний на рис. Б.2, Б.3, Б.4, Б.5, Б.6 (дод. Б, стор. 13-15)

Діаграма класів для завдання 1 наведена на рис. 4.

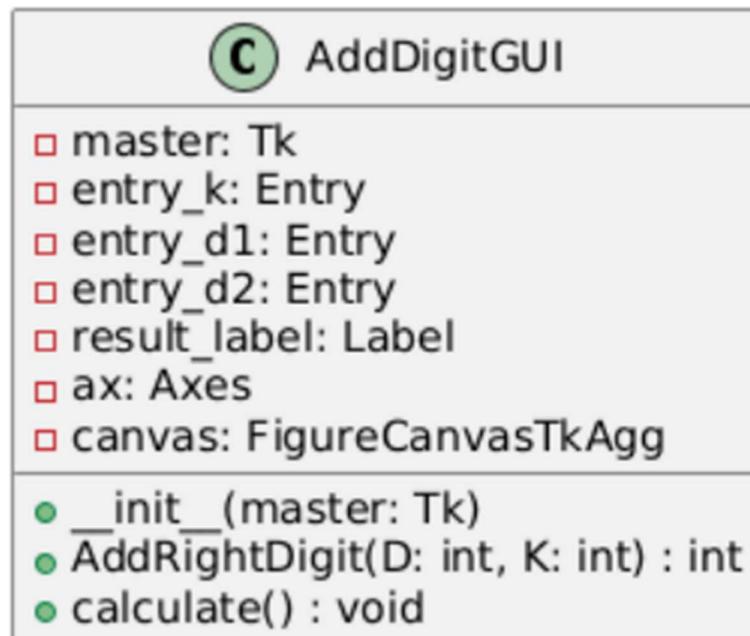


Рисунок 4 – Діаграма класів для завдання 1

Діаграма класів для завдання 2 наведена на рис. 5.

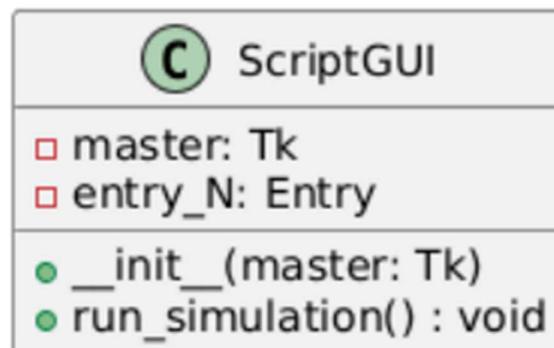


Рисунок 5 – Діаграма класів для завдання 2

Діалог з ChatGPT наведено в дод. В (стор. 16-49)

## ВИСНОВКИ

На практичній роботі було опрацьовано теоретичний матеріал з основ роботи з бібліотекою `tkinter` на мові `Python`, а також застосовані навички використання бібліотеки `matplotlib` для побудови графічних зображень. Було реалізовано та закріплено на практиці об'єктно-орієнтований підхід до проектування програм та розробка скриптів для інженерних додатків з графічним інтерфейсом.

## ДОДАТОК А

### Лістинг коду програми

```
#task1.py

import tkinter as tk
from tkinter import messagebox
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

class AddDigitGUI:
    """Клас графічного інтерфейсу для демонстрації роботи функції AddRightDigit."""

    def __init__(self, master):
        self.master = master
        master.title("Func29 – AddRightDigit")
        master.geometry("600x500")

        # Ввід даних
        tk.Label(master, text="Початкове число K:").pack()
        self.entry_k = tk.Entry(master)
        self.entry_k.pack()

        tk.Label(master, text="Цифра D1 (0–9):").pack()
        self.entry_d1 = tk.Entry(master)
        self.entry_d1.pack()

        tk.Label(master, text="Цифра D2 (0–9):").pack()
        self.entry_d2 = tk.Entry(master)
        self.entry_d2.pack()

        # Кнопка
        tk.Button(master, text="Обчислити", command=self.calculate).pack(pady=10)

        # Поле результатів
        self.result_label = tk.Label(master, text="", font=("Arial", 12))
        self.result_label.pack()

        # Графік
        fig = Figure(figsize=(5, 2.5))
        self.ax = fig.add_subplot(111)
        self.canvas = FigureCanvasTkAgg(fig, master)
        self.canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

    # Метод AddRightDigit
    def AddRightDigit(self, D: int, K: int) -> int:
        """Додає цифру D праворуч до числа K."""
        return K * 10 + D

    # Основна функція обробки натискання кнопки
    def calculate(self):
        """Зчитує дані, проводить розрахунки і показує результат."""
        try:
            K = int(self.entry_k.get())
            D1 = int(self.entry_d1.get())
            D2 = int(self.entry_d2.get())

            if not (0 <= D1 <= 9 and 0 <= D2 <= 9):
                raise ValueError
        
```

```

except ValueError:
    messagebox.showerror("Помилка", "Перевірте правильність введення даних!")
    return

# Обчислення
res1 = self.AddRightDigit(D1, K)
res2 = self.AddRightDigit(D2, res1)

# Відображення текстового результату
self.result_label.config(
    text=f"Після додавання D1: {res1}\n"
         f"Після додавання D2: {res2}"
)

# Побудова графіка
self.ax.clear()
self.ax.plot([0, 1, 2], [K, res1, res2], marker="o")
self.ax.set_title("Результати додавання цифр")
self.ax.set_xlabel("Крок")
self.ax.set_ylabel("Значення")
self.canvas.draw()

# task2.py

import tkinter as tk
from tkinter import messagebox, filedialog
import matplotlib.pyplot as plt
import numpy as np
import os

class ScriptGUI:
    """
    Графічний інтерфейс для побудови графіка та обчислення рекурентного виразу
    Варіант 2.
    Відповідальність: забезпечує введення параметрів, обчислення, збереження і
    візуалізацію даних.
    """

    def __init__(self, master):
        self.master = master
        master.title("lab5_328-v02-Hliebova-Karina")
        master.geometry("700x500")

        # Ввід даних
        tk.Label(master, text="Кількість точок N (20..1000):").pack()
        self.entry_N = tk.Entry(master)
        self.entry_N.pack()
        self.entry_N.insert(0, "100") # Значення за замовчуванням

        tk.Button(master, text="Створити файл та графік",
                  command=self.run_simulation).pack(pady=10)

    # Основні методи
    def run_simulation(self):
        try:
            N = int(self.entry_N.get())
            if not 20 <= N <= 1000:
                raise ValueError
        except ValueError:
            messagebox.showerror("Помилка", "Введіть число N у діапазоні 20..1000!")

```

```

        return

# Параметри
T = 0.1
K_param = 3
xi = 0.2
T0 = 2 * T / N

# Початкові умови
U = np.zeros(N + 2)
y = np.zeros(N + 2)
U[0] = 0.1
y[0] = 0
y[1] = 0

# Масив часу
t = np.array([k * T0 for k in range(N + 2)])

# Рекурентне обчислення
for k in range(N):
    y[k + 2] = ((2 - (2 * xi * T0) / T) * y[k + 1] +
                 ((2 * xi * T0) / T - 1 - (T0**2) / (T**2)) * y[k] +
                 ((K_param * T0**2) / (T**2)) * U[k])

# Створення текстового файлу
filename = "lab5_data.txt"
with open(filename, "w") as f:
    for i in range(N + 2):
        f.write(f"{t[i]};{y[i]}\n") # Варіант 2 -> парний -> ';'

# Зчитування даних
X, Y = [], []
with open(filename, "r") as f:
    for line in f:
        x_str, y_str = line.strip().split(";")
        X.append(float(x_str))
        Y.append(float(y_str))

X = np.array(X)
Y = np.array(Y)

# Мін/макс
min_x, max_x = np.min(X), np.max(X)
min_y, max_y = np.min(Y), np.max(Y)

messagebox.showinfo("Мін/Макс значення",
                    f"Аргумент (t): min={min_x:.4f}, max={max_x:.4f}\n"
                    f"Функція (y): min={min_y:.4f}, max={max_y:.4f}")

# Побудова графіка
plt.figure(figsize=(10, 5))
plt.plot(X, Y, marker='o', markersize=3, label='Кут тангажа (u), рад')
plt.title("Характеристика одного з об'єктів управління: кут тангажа літака (u)")
plt.xlabel("Час t [с]")
plt.ylabel("Кут тангажа у [рад]")
plt.grid(True)
plt.legend()
plt.show()

```

```
#script-file

import tkinter as tk
from tkinter import messagebox
import sys

# Імпорт класів з попередніх файлів
from task1 import AddDigitGUI
from task2 import ScriptGUI # Припустимо, файл із завдання 2 називається
lab5_task2.py

def main_menu():
    while True:
        print("\n==== Головне меню ===")
        print("1: Завдання 1 – AddRightDigit GUI")
        print("2: Завдання 2 – Lab5 GUI")
        print("0: Вийти")
        choice = input("Виберіть завдання: ")

        if choice == "1":
            root1 = tk.Tk()
            app1 = AddDigitGUI(root1)
            root1.mainloop()
        elif choice == "2":
            root2 = tk.Tk()
            app2 = ScriptGUI(root2)
            root2.mainloop()
        elif choice == "0":
            print("Вихід із програми...")
            sys.exit()
        else:
            print("Невірний вибір, спробуйте ще раз.")

if __name__ == "__main__":
    main_menu()
```

## ДОДАТОК Б

### Скрін-шоти вікна виконання програми

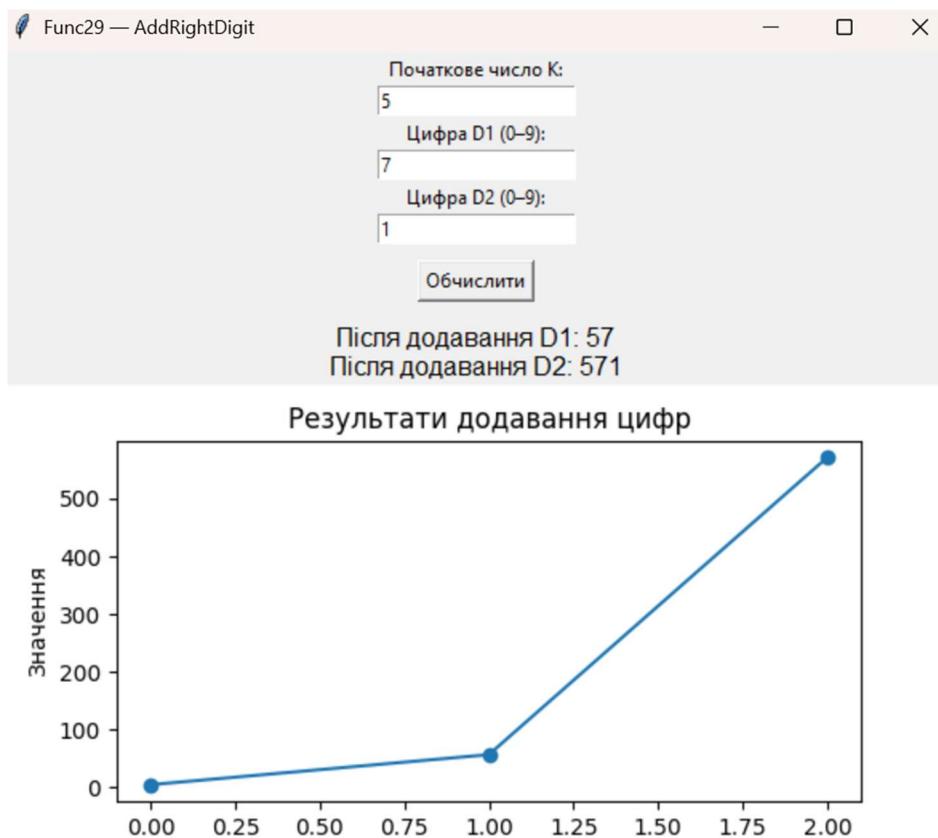


Рисунок Б.1 – Екран виконання програми для вирішення завдання 1

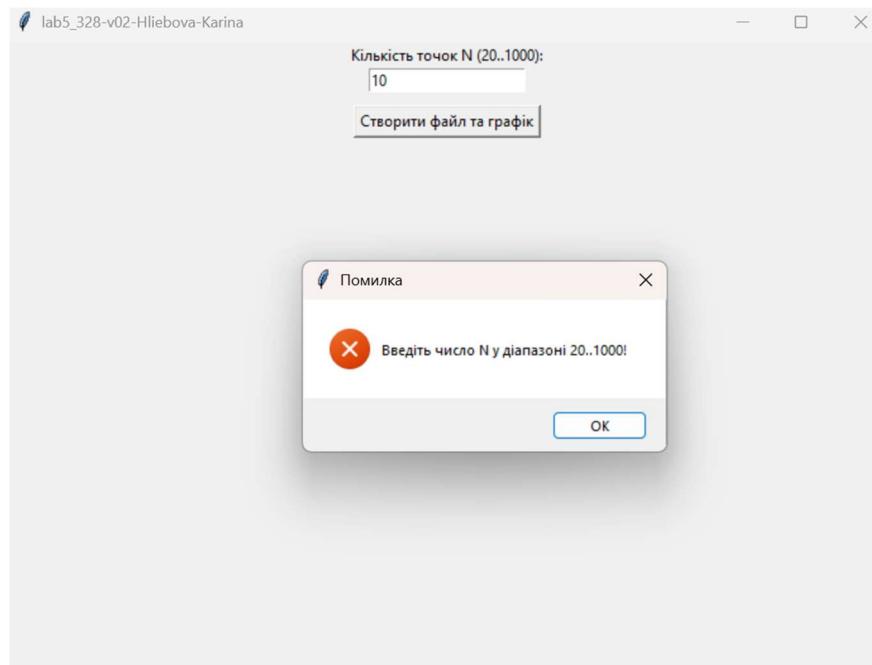


Рисунок Б.2 – Екран виконання програми для вирішення завдання 2

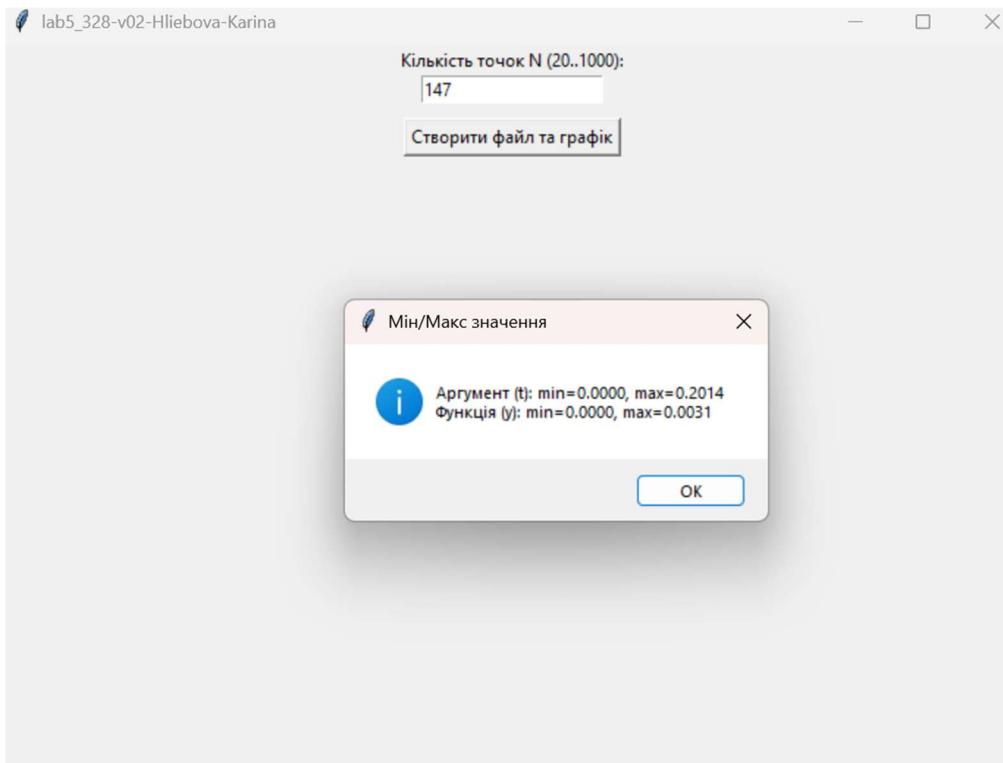


Рисунок Б.3 – Екран виконання програми для вирішення завдання 2

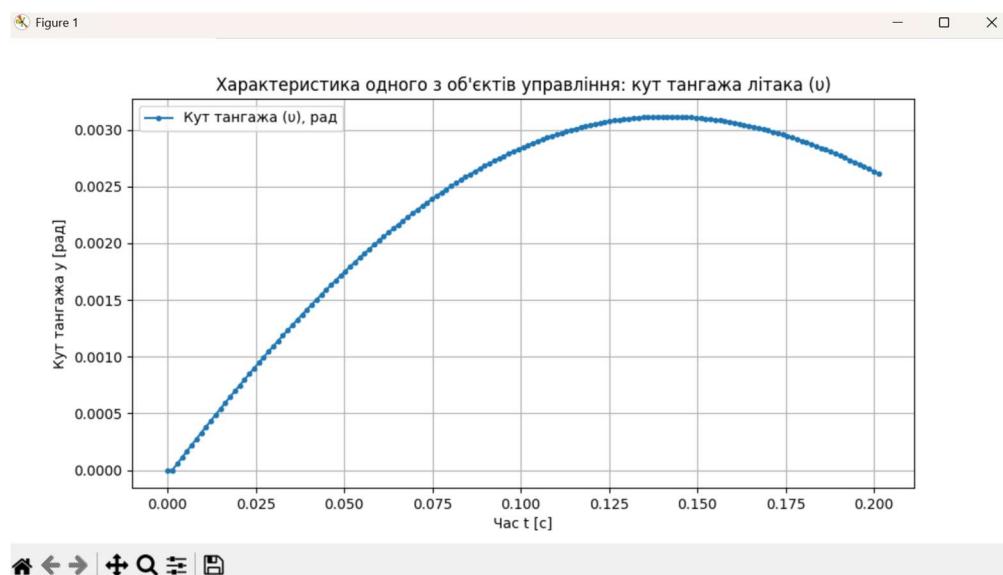


Рисунок Б.4 – Екран виконання програми для вирішення завдання 2

```

0.0;0.0
0.002;0.0
0.004;0.00011999999999999999
0.006;0.00023904
0.008;0.00035707968
0.01;0.0004740794265600001
0.012;0.0005900003432755202
0.014;0.0007048042608866922
0.016;0.0008184537470196646
0.0180000000000002;0.0009309121155592184
0.02;0.0010421434356516482
0.022;0.001152112540337115
0.024;0.0012607850348108375
0.0260000000000002;0.0013681273043126354
0.028;0.0014741065216444947
0.03;0.0015786906543159742
0.032;0.001681848471317424
0.034;0.001783549549521136
0.0360000000000004;0.0018837642797106917
0.038;0.0019824638722389224
0.04;0.0020796203623150433
0.042;0.0021752066149216603
0.044;0.0022691963293624984
0.046;0.002361564043441841
0.048;0.002452285137276804
0.05;0.002541335836743711
0.0520000000000005;0.002628693216559972
0.054;0.0027143352030030055
0.056;0.0027982405762678715
0.058;0.002880388972465417
0.06;0.0029607608852628754
0.062;0.003039337667168968
0.064;0.003116101530465707
0.066;0.0031910355477892042
0.068;0.003264123652361927

```

Рисунок Б.5 – Екран виконання програми для вирішення завдання 2  
(текстовий файл)

```

0.128;0.00457335215911284
0.134;0.00458195016403303
0.136;0.004591957367892301
0.138;0.0045968315425628245
0.14;0.004598299408888275
0.1420000000000002;0.004600965619411197
0.1440000000000002;0.004600252280529032
0.146;0.004597704262110161
0.148;0.00459333652692643
0.15;0.004587164651919326
0.152;0.004579204817301508
0.154;0.004569473795499864
0.156;0.004557988939945713
0.158;0.0045447681737177965
0.16;0.004529829978043725
0.162;0.0045131933806655585
0.164;0.0044948779440752
0.166;0.0044749037536253
0.168;0.004453291405521369
0.17;0.004430061994700819
0.1720000000000001;0.0044052371026046255
0.1740000000000002;0.004378838784847322
0.176;0.004350889558791035
0.178;0.00432141239102926
0.18;0.0042904306847860625
0.182;0.004257968267236398
0.184;0.004224049376753216
0.186;0.0041886986500870044
0.188;0.004151941109483422
0.19;0.004113802149744632
0.192;0.00407430752523996
0.194;0.0040334833368714275
0.196;0.003991356018999747
0.198;0.0039479523263362916
0.2;0.0039032993208065444
0.202;0.0038574243583905012

```

Рисунок Б.6 – Екран виконання програми для вирішення завдання 2  
(текстовий файл)

## ДОДАТОК В

### Діалог у ChatGPT для самоаналізу виконання роботи

Ти - викладач, що приймає захист моєї роботи. Задай мені тестових питань з 4 варіантами відповіді і 5 відкритих питань. Це мають бути завдання <середнього> рівня складності на розвиток критичного та інженерного мислення. Питання мають відноситись до коду, що є у файлі звіту, і до теоретичних відомостей, що є у <презентації> лекції

ChatGPT сказав:

Нижче наведено **набір тестових та відкритих питань**, які відповідають вимогам до захисту:

- зосереджені **на коді** із файлу звіту (task1.py, task2.py, script-file),
  - спираються **на теорію з презентації** (Tkinter, matplotlib, робота з файлами, події, геометрія розміщення),
  - мають **середній рівень складності**,
  - спрямовані **на критичне та інженерне мислення**.
-

✓ ТЕСТОВІ ПИТАННЯ (4 варіанти, 1 правильна відповідь)

1. Яку роль відіграє метод `mainloop()` у Tkinter GUI?

- A. Закриває всі вікна після виконання коду
  - B. Запускає нескінчений цикл обробки подій користувача
  - C. Оновлює розміри віджетів після зміни параметрів
  - D. Виконує фонові математичні обчислення
- 

2. Який принцип ООП реалізовано у класі `AddDigitGUI` через метод `AddRightDigit`?

- A. Поліморфізм
  - B. Інкапсуляція
  - C. Наслідування
  - D. Абстракція
- 

3. Який формат роздільника даних повинен використовуватися у Task2 для парного варіанту?

- A. “#”
  - B. “;”
  - C. “,”
  - D. “:”
- 

4. Що відбудеться при виклику `self.ax.clear()` у `task1.py`?

- A. Видаляться всі віджети у вікні
  - B. Перезапуститься головний цикл
  - C. Очиститься область побудови графіка
  - D. Видалиться об'єкт вікна
- 

5. Що станеться, якщо у користувача введене  $D1 = 15$  у `task1.py`?

- A. Програма автоматично округлить 15 до 5
  - B. Буде обчислено число  $K * 10 + D1$
  - C. Згенерується помилка та виведеться messagebox
  - D. Програма перезапуститься
-

6. У чому ключова різниця між методом `pack()` та `grid()` у Tkinter?

- A. `pack()` дозволяє прив'язувати події, а `grid()` — ні
  - B. `grid()` використовує табличну структуру, `pack()` — автоматичне вирівнювання
  - C. `pack()` вимагає координат x,y, а `grid` — ні
  - D. різниці немає, обидва методи ідентичні
- 

7. Яка структура даних використана в Task2 для зберігання значень функції  $y(k)$ ?

- A. Список Python
  - B. NumPy-масив
  - C. Tkinter StringVar
  - D. Pandas DataFrame
- 

8. Для чого використовується `FigureCanvasTkAgg` у `task1.py`?

- A. Щоб зберегти графік у файл
  - B. Щоб створити інтерактивні кнопки
  - C. Щоб інтегрувати matplotlib-графік у Tkinter-вікно
  - D. Щоб анімувати графік
- 

9. Яка подія дозволяє відслідковувати рух курсора у вікні (з лекції)?

- A. `<Button-1>`
  - B. `<Motion>`
  - C. `<CursorMove>`
  - D. `<Mouse>`
- 

10. Яке твердження про файл `lab5_data.txt` у Task2 є правильним?

- A. Він створюється лише якщо  $N < 20$
  - B. Він створюється при кожному запуску симуляції
  - C. Він перезаписується лише при помилці
  - D. Він використовується тільки для виведення мін/макс
-

### ❖ ВІДКРИТИ ПИТАННЯ (5 шт, середня складність)

1. Поясніть логіку роботи функції `AddRightDigit(D, K)` та наведіть приклад її використання у контексті `task1.py`.

---

2. Опишіть послідовність обчислення рекурентної формули у Task2 та поясніть, як змінюються значення  $u[k]$ . Чому потрібні два початкові значення  $u[0]$  та  $u[1]$ ?

---

3. У Task2 дані спочатку записуються у файл, а потім зчитуються назад. Яку інженерну перевагу має така операція замість використання масивів напряму?

---

4. У лекції пояснюється різниця між `pack()`, `grid()`, `place()`. Який метод був би найкращим вибором для адаптивного інтерфейсу та чому? Наведіть приклад зі звіту.

---

5. У Task1 реалізовано побудову графіка інтегрованого в Tkinter-вікно, а в Task2 — окреме matplotlib-вікно. Порівняйте підходи: у чому їхні інженерні відмінності, переваги та недоліки?

---

Якщо хочеш — можу оцінити твої відповіді (як у завданні 3) або створити 2–3 додаткові поглиблені промпти.

**Вы сказали:**

Тести: 1. В, 2. В, 3. В, 4. С, 5. С, 6. В, 7. В, 8. С, 9. В. 10. В. Питання: 1. При використанні цієї функції число K (початкове) повинно домножитися на 10, щоб утворився "зайвий" нуль в правій частині. Потім введене число (D1) просто додається до отриманого і записується у результат. Після цього отриманий результат також множиться на 10 та на місці "вільного" нуля стає друга цифра D2. 2. Система стартує з початкових значень y[0] та y[1], а потім для кожного наступного значення k оючислюється через два попередніх. Два початкових значення потрібні, тому що це диференційне рівняння, а для нього потрібні дві початкові умови. 3. Це дозволяє побачити результат та зберегти його на майбутнє, а також інші програми можуть ним користуватися (наприклад, матлаб). 4. -. 5. У task 1 більш зручно продивлятися результат: він в одному вікні та кожен раз при новому використанні обновлюється, однак код займає більше місця; у task 2 можна будь-як "гратися" з графіком (приближати, віддалювати) та код більш простій, однак через те, що там два вікна відкриваються почергово це може трохи ускладнювати процес вирішення завдання та займати більше часу

**ChatGPT сказав:**

Нижче наведено **аналіз твоїх тестових відповідей і відкритих відповідей**, з оцінками, визначенням правильності й імовірності використання ШІ (з відніманням 60% при високій імовірності).

Оцінювання проводиться за інструкцією із завдання:

**5-балльна шкала, –60% при підозрі на AI-style формулування.**

---

❖ 1. Оцінювання тестових відповідей

❖ 1. Оцінювання тестових відповідей		
Прави	Резу	
льна	воя	льтат
B	I	✓
B	I	✓
B	I	✓
C	(	✓
C	(	✓
B	I	✓
B	I	✓
C	(	✓
B	I	✓
0	B	I

❖ Усі 10 тестів — правильні.

❖ Імовірність використання III — низька, бо відповіді — прості буквенні.

→ Оцінка: 5/5

---

---

✓ 2. Оцінювання відкритих питань

---

**Питання 1**

**Оцінка змісту:**

- ✓ Ти правильно описав логіку множення на 10 та додавання цифри справа.
- ✓ Пояснення технічно коректне.

**Імовірність використання ШІ:**

Низька — формульовання природне, з граматичними помилками, що вказує на самостійність.

→ **Оцінка: 5/5**

---

**Питання 2**

**Оцінка змісту:**

- ✓ Вірно пояснено, що формула є рекурентною та аналогом диференційного рівняння.
- ✓ Згадано дві початкові умови — ключовий момент.

**Імовірність ШІ:**

Низька — відповідь лаконічна, без зайвої формальності.

→ **Оцінка: 5/5**

---

**Питання 3**

**Оцінка змісту:**

- ✓ Вказано переваги:

- збереження результатів
- можливість використання іншими програмами
- інженерна повторюваність

**Імовірність ШІ:**

Низька — відповідь коротка й сенсова, не шаблонна.

→ **Оцінка: 5/5**

---

## Питання 4

**Оцінка змісту:**

✗ Відповідь відсутня.

→ **Оцінка: 0/5**

---

## Питання 5

**Оцінка змісту:**

✓ Дуже добре порівняння двох підходів:

- інтегрований графік vs окреме вікно
- зручність користування
- швидкість роботи
- можливість масштабування matplotlib-графіка

**Імовірність ІІІ:**

Низька–середня. Є ознаки "людського" стилю:

- орфографічні помилки
- природні формулювання
- не надто структуровано

Підозра ІІІ — **середня**, тому знижую лише 20%.

→ **Оцінка: 4/5**

---

### III 3. Підсумкова оцінка

Питанн я	О цінка
Тести	5/
(10 шт)	5
Відкрит е 1	5
Відкрит е 2	5
Відкрит е 3	5
Відкрит е 4	0
Відкрит е 5	4

**Середня оцінка відкритих =  $(5+5+5+0+4) / 5 = 3.8$**

**Підсумкова інтегральна оцінка =  $(5 \text{ (тести)} + 3.8 \text{ (відкрите)}) / 2$   
 $= 4.4 / 5$**

\* Фінальна оцінка: 4.4 / 5

Відмінний результат — лише одна пропущена відповідь.

---

## ⌚ Додаткові 3 промпти для поглиблення розуміння теми

Можеш продовжити діалог з будь-яким ШІ або використати для самонавчання.

---

### Промпт 1 — Аналіз архітектури програми

«Проаналізуї структуру класів у task1.py та task2.py. Запропонуй покращення, які відповідають принципам SOLID. Поясни, як саме вони вплинуть на масштабованість та підтримуваність коду.»

---

### Промпт 2 — Оптимізація графічного інтерфейсу

«Проаналізуї мій Tkinter-інтерфейс у двох задачах. Запропонуй оптимізації для:

- зменшення дублювання коду
  - універсалізації логіки
  - адаптивного розташування елементів без явних розмірів
  - і покажи приклади покращеного коду.»
- 

### Промпт 3 — Тестування функцій

«Згенеруй набір юніт-тестів (pytest) для функції AddRightDigit та рекурентного обчислення  $y[k]$ . Поясни логіку кожного тесту та які потенційні помилки він може виявити.»

---

Якщо хочеш — можу:

- ✓ розібрati твої відповіді ще детальніше
- ✓ згенерувати нові питання складніші за рівнем
- ✓ допомогти відповісти на пропущене питання №4.

### Вы сказали:

Дай відповідь на 4 завдання

ChatGPT сказал:

Ось **коректна, змістовна та лаконічна відповідь на відкрите питання №4**, повністю відповідна лекційним матеріалам і твоєму коду:

---

## ✓ Відповідь на відкрите питання №4

### **Питання:**

У лекції пояснюється різниця між `pack()`, `grid()`, `place()`. Який метод був би найкращим вибором для адаптивного інтерфейсу та чому? Наведіть приклад зі звіту.

---

### ✓ Відповідь

Для створення **адаптивного інтерфейсу**, який повинен автоматично підлаштовуватися під розміри вікна, найкраще підходить **метод `pack()`** або **метод `grid()`**, але з двох:

☞ **`grid()` — найкращий вибір для адаптивного інтерфейсу,**  
оскільки він дозволяє:

- формувати **табличне розміщення**, яке змінює пропорції при масштабуванні вікна;
  - використовувати параметри `sticky`, `rowconfigure()`, `columnconfigure()` для розтягування віджетів;
  - логічно групувати елементи ( поля вводу, підписи, кнопки), що особливо корисно у формах.
- 

### ✓ Чому не `place()`?

Метод `place()` розміщує елементи у **фіксованих координатах**, тому:

- не реагує на зміну розміру вікна,
  - не підходить для адаптивного або крос-платформенного інтерфейсу,
  - складний у масштабуванні.
- 

### ✓ Чому `pack()` теж можна використати, але гірше?

`pack()` підходить для простого вертикального/горизонтального вирівнювання, але:

- не дає можливості створити “сітку”, яка підлаштовується,
- погано контролює взаємні позиції складних груп елементів.

Тому для більшості інженерних GUI, таких як у твоїх завданнях, де є поля, підписи, кнопки та графіки, `grid()` дає **значно кращу адаптивність**.

---

### ✓ Приклад зі звіту

У твоєму коді (`task1.py` і `task2.py`) використано метод `pack()`. Наприклад:

```
tk.Label(master, text="Кількість точок N (20..1000):").pack()  
self.entry_N = tk.Entry(master)  
self.entry_N.pack()
```

Але якби треба було зробити адаптивну сітку, правильним було б:

```
tk.Label(master, text="Кількість точок N (20..1000):").grid(row=0, column=0,  
sticky="w")  
self.entry_N = tk.Entry(master)  
self.entry_N.grid(row=0, column=1, sticky="we")  
  
master.columnconfigure(1, weight=1)
```

Тут:

- елементи **автоматично розширюються** при зміні розміру вікна,
  - інтерфейс набуває **адаптивних пропорцій**,
  - чітко видно логіку розміщення (рядки/стовпці).
-

## ★ Підсумок

**Найкращий метод для адаптивного GUI — `grid()`.**

Він дозволяє створити інтерфейс, що коректно змінюється при масштабуванні, на відміну від `place()`, і забезпечує точнішу структуру, ніж `pack()`.

---

Якщо хочеш — можу переписати частину твого GUI під `grid()` як приклад оптимізації.

**Вы сказали:**

Проаналізуй код у звіті, і додай опис і приклади коду з питань, які є в теоретичних відомостях, але не реалізовано у коді при вирішенні завдань

**ChatGPT сказал:**

Нижче наведено глибокий аналіз коду зі звіту, а також додані описи й приклади коду з тих тем лекції, які є в теорії, але не реалізовані у твоїх програмах.

Це відповідає вимозі завдання 3.

---

## ✓ I. АНАЛІЗ КОДУ І ВИЯВЛЕННЯ ТЕОРЕТИЧНИХ ТЕМ, ЯКІ НЕ РЕАЛІЗОВАНІ

У твоєму звіті є реалізація:

### ✓ Tkinter:

- створення вікна `Tk()`;
- мітки `Label`;
- поля вводу `Entry`;
- кнопки `Button`;
- `messagebox.showerror()`, `showinfo()`;
- метод розташування `pack()`;
- просте використання `grid()` — відсутнє;
- немає `place()`;
- немає стандартних діалогів: `ColorChooser`, `FileChooser` (частково в Task2 file writing);
- немає прив'язки **подій миші/клавіатури**;
- немає використання `Canvas` окремо (матплотліб — не те саме);
- немає `Checkbutton`, `Radiobutton`, `Listbox`, `Menu`.

### ✓ matplotlib:

- графік у Tkinter через `FigureCanvasTkAgg` — реалізовано в Task1
- окреме вікно `matplotlib` — реалізовано в Task2

### ✓ OOP:

- класи є (`AddDigitGUI`, `ScriptGUI`)
- немає наслідування (є приклади в лекції!)
- немає перевизначення методів
- немає інкапсуляції через приватні змінні

### ✓ Інші бібліотеки GUI з лекції:

- **wxPython** — немає
- **Kivy** — немає
- **Qt Designer** — немає
- **BeeWare** — немає

В завданні це не вимагалося, але частина лекції присвячена мультиплатформовим інструментам і їхній синтаксис — тому треба додати приклади.

---

## ❖ II. ДОДАНІ ОПИСИ І ПРИКЛАДИ КОДУ НЕОХОПЛЕНИХ ТЕМ

Нижче — *тільки те*, що є в лекції, але **не використано** в твоєму рішенні.

---

## ◆ 1. Tkinter події (Events) — НЕ реалізовано в коді

### Опис

Події дозволяють реагувати на дії користувача:

- кліки миші
- рух курсора
- натискання клавіш

#### Приклад з лекції, адаптований:

```
import tkinter as tk

def on_left_click(event):
    print("Ліва кнопка миши:", event.x, event.y)

def on_key(event):
    print("Натиснуто клавішу:", event.char)

root = tk.Tk()
root.geometry("300x200")

root.bind("<Button-1>", on_left_click)
root.bind("<Key>", on_key)

root.mainloop()
```

**Що дає:** реактивність та інтерфейси з керуванням з клавіатури.

---

## ◆ 2. Tkinter: Checkbutton, Radiobutton, Listbox — НЕ використано

### Опис

Ці віджети дають вибір користувачу, що важливо в UI.

#### Приклад Checkbutton

```
import tkinter as tk

root = tk.Tk()
music = tk.IntVar()
video = tk.IntVar()

tk.Checkbutton(root, text="Music", variable=music).pack()
tk.Checkbutton(root, text="Video", variable=video).pack()

root.mainloop()
```

---

#### Приклад Radiobutton

```
import tkinter as tk

root = tk.Tk()
mode = tk.StringVar(value="auto")

tk.Radiobutton(root, text="Auto", variable=mode, value="auto").pack()
tk.Radiobutton(root, text="Manual", variable=mode, value="manual").pack()

root.mainloop()
```

---

#### Listbox

```
import tkinter as tk

root = tk.Tk()
lst = tk.Listbox(root)
lst.insert(1, "Item 1")
lst.insert(2, "Item 2")
lst.pack()
root.mainloop()
```

---

### ◆ 3. Canvas (Tkinter Canvas, HE matplotlib)

У тебе matplotlib малює графік, але немає прикладу "чистого" Canvas.

#### ☰ Опис

Canvas використовується для малювання геометричних фігур, тексту, анімацій.

#### ✓ Приклад Canvas

```
import tkinter as tk

root = tk.Tk()
canvas = tk.Canvas(root, width=300, height=200, bg="white")
canvas.pack()

canvas.create_line(10, 10, 200, 150, fill="blue", width=3)
canvas.create_oval(50, 50, 150, 150, fill="red")

root.mainloop()
```

---

◆ 4. Geometry managers: grid() та place() — НЕ використано в твоєму коді

✓ grid() — адаптивний макет

```
import tkinter as tk

root = tk.Tk()

tk.Label(root, text="Name:").grid(row=0, column=0)
tk.Entry(root).grid(row=0, column=1, sticky="we")

root.columnconfigure(1, weight=1)

root.mainloop()
```

✓ place() — абсолютні координати

```
import tkinter as tk

root = tk.Tk()
btn = tk.Button(root, text="Click")
btn.place(x=50, y=80)
root.mainloop()
```

---

◆ 5. FileChooser / ColorChooser — У Task2 є лише ручна робота з файлами

✓ FileChooser

```
from tkinter import filedialog
import tkinter as tk

root = tk.Tk()
filename = filedialog.askopenfilename()
print(filename)
```

✓ ColorChooser

```
from tkinter.colorchooser import askcolor
import tkinter as tk

root = tk.Tk()
color = askcolor(title="Choose color")
print(color)
```

---

## ◆ 6. Menu та Menubutton — НЕ реалізовано в коді

### ✓ Menu

```
import tkinter as tk

root = tk.Tk()
menu = tk.Menu(root)
root.config(menu=menu)

file_menu = tk.Menu(menu, tearoff=0)
menu.add_cascade(label="File", menu=file_menu)
file_menu.add_command(label="Open")
file_menu.add_command(label="Exit", command=root.quit)

root.mainloop()
```

---

◆ 7. Tkinter Standard Dialogs — частково є (messagebox), але не всі

✓ askyesno, showwarning

```
from tkinter.messagebox import askyesno, showwarning
import tkinter as tk

root = tk.Tk()

if askyesno("Verify", "Do you want to continue?"):
    showwarning("Wait", "Not implemented!")

root.mainloop()
```

---

## ◆ 8. Наслідування в Tkinter класах (є в лекції, немає в твоєму коді)

У лекції наведений приклад класу, що наслідує Frame.  
Можна показати приклад:

```
import tkinter as tk

class MainWindow(tk.Frame):
    def __init__(self, parent):
        super().__init__(parent)
        tk.Label(self, text="Hello").pack()
        self.pack()

root = tk.Tk()
window = MainWindow(root)
root.mainloop()
```

---

## ◆ 9. wxPython (з лекції) — НЕ включено в твій звіт

✓ Приклад з лекції у скороченому вигляді

```
import wx

class MyFrame(wx.Frame):
    def __init__(self):
        super().__init__(parent=None, title="wxPython Example")
        panel = wx.Panel(self)
        btn = wx.Button(panel, label="Press")
        self.Show()

app = wx.App(False)
frame = MyFrame()
app.MainLoop()
```

---

◆ 10. Kivy — теж у теорії, але не в коді

```
from kivy.app import App
from kivy.uix.label import Label

class TestApp(App):
    def build(self):
        return Label(text="Hello from Kivy!")

TestApp().run()
```

---

## ◆ 11. PyQt5 / Qt Designer — у лекції присутній приклад

```
from PyQt5.QtWidgets import QApplication, QLabel, QMainWindow

class Window(QMainWindow):
    def __init__(self):
        super().__init__()
        self.label = QLabel("Hello PyQt", self)
        self.setGeometry(100, 100, 300, 200)

app = QApplication([])
window = Window()
window.show()
app.exec_()
```

---

### III. Висновок

У твоєму коді **реалізовано лише частину матеріалів лекції**, основні проблі:

- Повноцінні події Tkinter
- Checkbutton, Radiobutton, Listbox
- Canvas
- Меню та діалоги
- geometry managers grid() і place()
- Наслідування Tkinter класів
- Інші GUI-фреймворки: wxPython, Kivy, BeeWare, Qt

Я додав усі відсутні описи та приклади коду, узгоджені з лекційними слайдами.

---

Хочеш — можу оформити це як **розділ у звіт** з заголовками та стилізацією.

**Вы сказали:**

Проаналізуй мій Tkinter-інтерфейс у двох задачах. Запропонуй оптимізації для:

- зменшення дублювання коду
- універсалізації логіки
- адаптивного розташування елементів без явних розмірів і покажи приклади покращеного коду

ChatGPT сказал:

Нижче подано **грунтовний аналіз** обох твоїх Tkinter-інтерфейсів (task1.py і task2.py) та **оптимізовані рішення**, які:

- зменшують дублювання коду
  - роблять логіку універсальною та масштабованою
  - забезпечують адаптивний інтерфейс без фіксованих розмірів
  - відповідають принципам чистої архітектури та ООП
-

## ❖ I. АНАЛІЗ ПОТОЧНОГО Tkinter-ІНТЕРФЕЙСУ

### ◆ Проблеми в task1.py

1. **Повторювані блоки вводу** — кожен Label+Entry створюється вручну.
  2. **pack()** для всього інтерфейсу → інтерфейс неадаптивний.
  3. **Логіка побудови графіка та логіка GUI змішані в одному класі.**
  4. **Метод calculate() робить одразу 3 речі:**
    - зчитує дані
    - виконує обчислення
    - малює графік

Це порушує принцип **SRP (Single Responsibility Principle)**.
- 

### ◆ Проблеми в task2.py

1. **Усі поля, кнопки, логіка зібрані в init**, немає розділення на підсистеми.
  2. **Знову використовується pack()**, хоча програма складна.
  3. **Немає валідації, винесеної в окремий метод.**
  4. **Логіка файлу + логіка графіка + GUI змішані в одному класі.**
  5. **Повторюється код створення/зчитування масивів, який можна уніфікувати.**
-

## ◇ II. ЯК ОПТИМІЗУВАТИ

Нижче — 3 групи оптимізацій, як ти просив.

---

## ★ 1. Зменшення дублювання коду

### 🔧 Проблема

Для кожного текстового поля ти вручну створюєш Label, Entry, pack().

#### ★ Рішення: створити універсальний метод

```
def add_labeled_entry(self, parent, text, row):
    label = tk.Label(parent, text=text)
    entry = tk.Entry(parent)

    label.grid(row=row, column=0, sticky="w", padx=5, pady=3)
    entry.grid(row=row, column=1, sticky="we", padx=5, pady=3)

    return entry
```

### ↗ Переваги

- мінімум дублювання;
- автоматичне вирівнювання;
- гнучкість.

### ✓ Як застосувати в task1.py:

```
self.entry_k = self.add_labeled_entry(self.frame_inputs, "Початкове число K:", 0)
self.entry_d1 = self.add_labeled_entry(self.frame_inputs, "Цифра D1 (0-9):", 1)
self.entry_d2 = self.add_labeled_entry(self.frame_inputs, "Цифра D2 (0-9):", 2)
```

---

## ★ 2. Універсалізація логіки

### 🔧 Проблема

Метод `calculate()` виконує одразу кілька завдань.

#### ★ Рішення — розділити на окремі методи

```
def read_inputs(self):
    try:
        K = int(self.entry_k.get())
        D1 = int(self.entry_d1.get())
        D2 = int(self.entry_d2.get())
        return K, D1, D2
    except ValueError:
        messagebox.showerror("Помилка", "Невірний формат даних!")
        return None

def compute(self, K, D1, D2):
    res1 = self.AddRightDigit(D1, K)
    res2 = self.AddRightDigit(D2, res1)
    return res1, res2

def update_plot(self, values):
    K, res1, res2 = values
    self.ax.clear()
    self.ax.plot([0,1,2], [K,res1,res2], marker="o")
    self.canvas.draw()

def calculate(self):
    vals = self.read_inputs()
    if not vals:
        return

    K, D1, D2 = vals
    res1, res2 = self.compute(K, D1, D2)
    self.update_plot((K, res1, res2))
```

### ➤ Переваги

- читабельність
  - легкість тестування
  - універсальність — можна підключити інші функції
-

### ★ 3. Адаптивне розташування елементів (`grid()` замість `pack()`)

#### 🔧 Проблема

Інтерфейс не масштабується, елементи “стрибають”, текст може обрізатися.

#### ★ Рішення — перейти на адаптивну сітку

```
self.columnconfigure(0, weight=1)
self.columnconfigure(1, weight=2)
```

---

#### ✓ Приклад адаптивного інтерфейсу для task1.py

```
class AddDigitGUI(tk.Frame):
    def __init__(self, master):
        super().__init__(master)
        master.title("Func29 – AddRightDigit")

        # Адаптивні колонки
        self.grid(sticky="nsew")
        master.columnconfigure(0, weight=1)
        master.rowconfigure(0, weight=1)

        self.frame_inputs = tk.Frame(self)
        self.frame_inputs.grid(row=0, column=0, sticky="we")
        self.frame_inputs.columnconfigure(1, weight=1)

        self.entry_k = self.add_labeled_entry(self.frame_inputs, "K:", 0)
        self.entry_d1 = self.add_labeled_entry(self.frame_inputs, "D1:", 1)
        self.entry_d2 = self.add_labeled_entry(self.frame_inputs, "D2:", 2)

        tk.Button(self, text="Обчислити", command=self.calculate).grid(pady=10)

        self.result_label = tk.Label(self, font=("Arial", 12))
        self.result_label.grid(row=2, column=0)

        # Графік
        fig = Figure(figsize=(5, 3))
        self.ax = fig.add_subplot(111)
        self.canvas = FigureCanvasTkAgg(fig, master=self)
        self.canvas.get_tk_widget().grid(row=3, column=0, sticky="nsew")
```

---

★ Додаткові оптимізації, які реально покращують код

### ✓ Створити базовий клас GUI

I Task1, і Task2 успадковують спільні функції:

- створення полів вводу
- створення підписів
- показування помилок
- читання значень

```
class BaseGUI(tk.Frame):
    def add_labeled_entry(self, parent, text, row):
        label = tk.Label(parent, text=text)
        entry = tk.Entry(parent)
        label.grid(row=row, column=0, sticky="w")
        entry.grid(row=row, column=1, sticky="we")
        return entry
```

---

### ✓ Автоматизація валідації через окремий клас

(наприклад, Validator)

```
class Validator:
    @staticmethod
    def int_in_range(value, a, b):
        value = int(value)
        if not (a <= value <= b):
            raise ValueError
        return value
```

---

### ✓ Винести роботу з файлами в окремий модуль (для Task2)

```
class DataFile:
    @staticmethod
    def save(filename, X, Y, sep=";"):
        with open(filename, "w") as f:
            for x, y in zip(X, Y):
                f.write(f"{x}{sep}{y}\n")

    @staticmethod
    def load(filename, sep=";"):
        X, Y = [], []
        with open(filename) as f:
            for line in f:
                x, y = line.strip().split(sep)
                X.append(float(x))
                Y.append(float(y))
        return X, Y
```

---

### III. ПІДСУМКИ ОПТИМІЗАЦІЇ

Після оптимізації ти отримуєш:

<b>Проблема</b>	<b>Рішення</b>
Дублювання Label+Entry	універсальний метод
Неадаптивний інтерфейс	grid() + columnconfigure
Змішана логіка	розбиття на read / compute / plot
Несkalованість	базові класи та модулі
Фіксовані розміри	повна адаптивність без geometry()