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 програмування СУ

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

ХАІ.301 . <спец.>. 322. 1 ЛР

Виконав студент гр. \_\_\_\_\_\_322\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Кулагін Олексій\_\_\_

(підпис, дата) (П.І.Б.)

Перевірив

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н., доц. О. В. Гавриленко

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ас.  В. О. Білозерський

(підпис, дата) (П.І.Б.)

2023

# 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 Х),

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 тангажa 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. Вирішення задачі Func26  
Вхідні дані:

x1 - координата першої точки (ліворуч нижній кут) прямокутника, тип: float, обмеження: числові значення.

y1 - координата першої точки (ліворуч нижній кут) прямокутника, тип: float, обмеження: числові значення.

x2 - координата другої точки (правий верхній кут) прямокутника, тип: float, обмеження: числові значення.

y2 - координата другої точки (правий верхній кут) прямокутника, тип: float, обмеження: числові значення.

Вихідні дані:

Perimeter - периметр прямокутника, тип: float, одиниця вимірювання: одиниці довжини.

Area - площа прямокутника, тип: float, одиниця вимірювання: квадратні одиниці довжини.  
  
Завдання 2. Вирішення задачі math17  
Вхідні дані:

U\_0 - початкове значення U[0]U[0]U[0], тип: float, обмеження: U0>0U\_0 > 0U0​>0, U0=0.1U\_0 = 0.1U0​=0.1.

T - значення TTT, тип: float, обмеження: T>0T > 0T>0, T=1.3T = 1.3T=1.3.

K - константа KKK, тип: float, обмеження: K>0K > 0K>0, K=4K = 4K=4.

xi - коефіцієнт ξ\xiξ, тип: float, обмеження: 0<ξ<10 < \xi < 10<ξ<1, ξ=0.7\xi = 0.7ξ=0.7.

T\_0 - параметр T0T\_0T0​, тип: float, обмеження: T0>0T\_0 > 0T0​>0, T0=2T\_0 = 2T0​=2.

y - початкові умови для y[0]y[0]y[0] і y[1]y[1]y[1], тип: list of floats, обмеження: y[0]=0y[0] = 0y[0]=0, y[1]=0y[1] = 0y[1]=0.  
  
  
  
  
  
Вихідні дані:

y[k] - значення y[k]y[k]y[k] на кожній ітерації, тип: list of floats, розмір: num\_iterations, результат рекурентного обчислення.

Графік - побудований графік значень y[k]y[k]y[k] по осі kkk, тип: графічне зображення (для виведення результатів на екран).

Лістинг коду вирішення задачі наведено в дод. А.  
 Екран роботи програми показаний на рис. Б.

# ВИСНОВКИ В ході роботи було вивчено основи роботи з бібліотекою tkinter для створення графічного інтерфейсу в Python, а також закріплено на практиці навички використання бібліотеки matplotlib для побудови графіків. Відпрацьовано об'єктно-орієнтований підхід до проектування програм, отримано навички розробки інженерних додатків з інтерактивним інтерфейсом для вирішення математичних задач. ДОДАТОК А

Лістинг коду програми до задач Func26

import tkinter as tk

from tkinter import messagebox

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

from matplotlib.figure import Figure

# Функція для обчислення периметра і площі прямокутника

def RectPS(x1, y1, x2, y2):

    # Обчислюємо довжину і ширину прямокутника

    width = abs(x2 - x1)

    height = abs(y2 - y1)

    perimeter = 2 \* (width + height)

    area = width \* height

    return perimeter, area

class MainWindow(tk.Frame):

    def \_\_init\_\_(self, parent):

        super().\_\_init\_\_(parent)

        self.parent = parent

        self.pack(fill=tk.BOTH, expand=1)

        # Розміщення елементів GUI

        self.grid\_rowconfigure(0, weight=1)

        self.grid\_columnconfigure(0, weight=1)

        self.lb1 = tk.Label(self, text="Enter x1, y1, x2, y2 coordinates:")

        self.lb1.grid(row=0, column=0, columnspan=2, sticky=tk.NSEW)

        self.x1\_entry = tk.Entry(self)

        self.y1\_entry = tk.Entry(self)

        self.x2\_entry = tk.Entry(self)

        self.y2\_entry = tk.Entry(self)

        self.x1\_entry.grid(row=1, column=0, sticky=tk.NSEW)

        self.y1\_entry.grid(row=1, column=1, sticky=tk.NSEW)

        self.x2\_entry.grid(row=1, column=2, sticky=tk.NSEW)

        self.y2\_entry.grid(row=1, column=3, sticky=tk.NSEW)

        self.calculate\_button = tk.Button(self, text="Calculate", command=self.calculate)

        self.calculate\_button.grid(row=2, column=0, columnspan=4, sticky=tk.NSEW)

        self.result\_label = tk.Label(self, text="Results will be shown here.")

        self.result\_label.grid(row=3, column=0, columnspan=4, sticky=tk.NSEW)

        self.plot\_button = tk.Button(self, text="Show Plot", command=self.show\_plot)

        self.plot\_button.grid(row=4, column=0, columnspan=4, sticky=tk.NSEW)

        self.data = []  # Для зберігання результатів периметрів і площ

    def calculate(self):

        try:

            # Читання координат з полів введення

            x1 = float(self.x1\_entry.get())

            y1 = float(self.y1\_entry.get())

            x2 = float(self.x2\_entry.get())

            y2 = float(self.y2\_entry.get())

            # Обчислення периметра і площі

            perimeter, area = RectPS(x1, y1, x2, y2)

            # Виведення результату на екран

            result\_text = f"Perimeter: {perimeter}\nArea: {area}"

            self.result\_label.config(text=result\_text)

            # Додавання результату до списку даних

            self.data.append((x1, y1, x2, y2, perimeter, area))

        except ValueError:

            messagebox.showerror("Input Error", "Please enter valid numeric values.")

    def show\_plot(self):

        if not self.data:

            messagebox.showerror("No Data", "Please calculate first.")

            return

        # Підготовка даних для графіка

        x1\_values = [d[0] for d in self.data]

        y1\_values = [d[1] for d in self.data]

        x2\_values = [d[2] for d in self.data]

        y2\_values = [d[3] for d in self.data]

        perimeter\_values = [d[4] for d in self.data]

        area\_values = [d[5] for d in self.data]

        # Створення графіку

        fig = Figure(figsize=(6, 4))

        ax = fig.add\_subplot(111)

        # Побудова графіка периметрів і площ

        ax.plot(range(1, len(self.data) + 1), perimeter\_values, label="Perimeter", color="blue", marker="o")

        ax.plot(range(1, len(self.data) + 1), area\_values, label="Area", color="red", marker="x")

        ax.set\_title("Perimeter and Area of Rectangles")

        ax.set\_xlabel("Rectangle Number")

        ax.set\_ylabel("Value")

        ax.legend()

        ax.grid(True)

        # Відображення графіка в Tkinter

        canvas = FigureCanvasTkAgg(fig, master=self)

        canvas.get\_tk\_widget().grid(row=5, column=0, columnspan=4, sticky=tk.NSEW)

        canvas.draw()

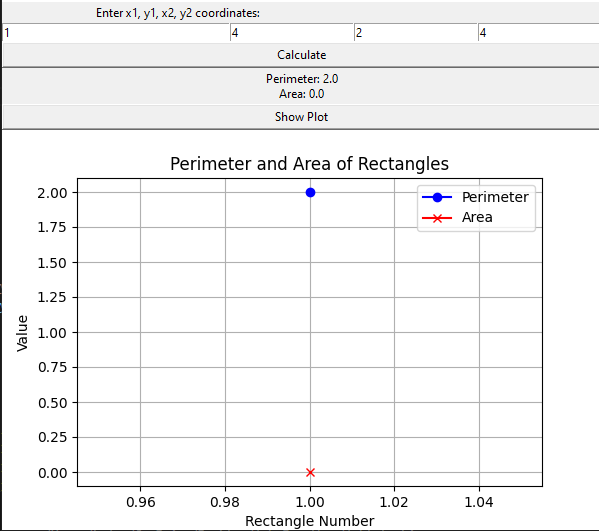
if \_\_name\_\_ == "\_\_main\_\_":

    app = tk.Tk()

    app.title("Rectangle Perimeter and Area Calculator")

    window = MainWindow(app)

    app.mainloop()

ДОДАТОК Б  
 Скрін-шоти вікна виконання програми Func26  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
 ДОДАТОК А.2

Лістинг коду програми до задач math17  
import numpy as np

import matplotlib.pyplot as plt

# Заданные параметры

U\_0 = 0.1  # Начальное значение U[0]

T = 1.3    # Значение T

K = 4      # Константа K

xi = 0.7   # Коэффициент xi

T\_0 = 2    # Параметр T0 (предположительно)

# Начальные условия

y = [0, 0]  # Начальные значения y[0] и y[1]

# Количество итераций

num\_iterations = 100

# Вычисление значений y по рекуррентной формуле

for k in range(0, num\_iterations - 2):  # Начинаем с 0, так как y[k+2] должно быть вычислено

    y\_next = (2 - 2 \* xi \* T\_0 / T) \* y[k+1] + (2 \* xi \* T\_0 / T - (T\_0\*\*2) / (T\*\*2)) \* y[k] + (K \* T\_0\*\*2) / (T\*\*2) \* U\_0

    y.append(y\_next)

# Построение графика

x = np.arange(num\_iterations)  # Индексы для оси x

plt.plot(x, y, label="y[k]", color="b")

plt.title("График y[k] по рекуррентной формуле")

plt.xlabel("Итерация k")

plt.ylabel("y[k]")

plt.grid(True)

plt.legend()

plt.show()

ДОДАТОК Б.2

Скрін-шоти вікна виконання програми Point17

