Vyšší odborná škola a Střední průmyslová škola elektrotechnická
Božetěchova 3, Olomouc
Laboratoře elektrotechnických měření

DOKUMENTACE K PROJEKTU

Název úlohy

GONIOMETICKÉ FUNKCE

Poř. č.	Příjmení a jméno	Třída	Skupina	Školní rok
13	Markvart Patrik	4.B	2.	2022/23

		Počet listů				

Dokumentace obsahuje:

Zadání
Teoretický úvod
Závěr
Použité knihovky
Popis kódu

Zadání:

- Vytvořte v Pythonu aplikaci s grafickým uživatelským prostředím pro vykreslení grafu funkce např. y=a·sin(b·uhel+c)+d
- Umožněte zadávání parametrů funkce (a, b, c, d) a rozsah úhlu
- Umožněte zadávání základních parametrů grafu (mřížka, název, popisky os, barva a tloušťka čáry)
- Vytvořte dokumentaci

Použité knihovny:

1. základní python
2. matplotlib
3. tkinter
4. numpy
5. python3-pil

Teoretický úvod:

1. Tkinter = knihovna pro jazyk python, umožňující vytváření oken s grafickým rozhraním.
2. Matplotlib = knihovna pro jazyk python, umožňující vykreslování grafů v grafické podobě
3. Numpy= knihovna pro jazyk python, umožňující komplikovanější práci s čísly

Popis kódu:

Slovní:

Kód se skládá ze tří částí: 1. importování knihoven, 2. definování okna a elementů (nyní jen widgetů) v něm, 3. definování funkcí vykonávající různé úkony.

Podrobný s obrázky:

1. Import potřebných knihoven

```
from os.path import basename, splitext
import tkinter as tk
import tkinter.ttk as ttk
import numpy as np
import matplotlib
from matplotlib import pyplot as plt
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
```

2. Vytvoření Class okna a funkce __init__

```
class Application(tk.Tk):
    name = basename(splitext(basename(__file__.capitalize()))[0])
    name = "Goniometrické funkce"

    def __init__(self):
        super().__init__(className=self.name)
```

3. Definování základních parametů okna a klávesových zkratek

```
## Basic "all-n-all" setup
self.title(self.name)
self.bind("<Escape>", self.quit)
self.bind('<Return>', self.draw)
self.geometry("1300x700")
self.resizable(width=tk.FALSE, height=tk.FALSE)
```

4. Rozložení okna

```
## Label Frame setup + Label + Separator
self.headerFrame = tk.LabelFrame(self, borderwidth = 0, highlightthickness = 0)
self.headerFrame.pack(side=tk.TOP)
self.headerLabel = tk.Label(self.headerFrame, text="Goniometrické funkce", font=("Arial", 24))
self.headerLabel.pack()
ttk.Separator(self, orient='horizontal').pack(fill=tk.X, padx=2)

## Content Frame setup + Options Frame setup + Graph Frame setup
self.contentFrame = tk.Frame(self, width=1300, height=700)
self.contentFrame.pack(side=tk.TOP, pady=5)
self.optionsFrame = tk.LabelFrame(self.contentFrame, text="Výběr", width=700, height=700)
self.optionsFrame.pack(side=tk.LEFT, padx=5, pady=5)
self.optionsFrame.pack_propagate(False)
self.graphFrame = tk.LabelFrame(self.contentFrame, text="Graf", width=600, height=700)
self.graphFrame.pack(side=tk.RIGHT, padx=5, pady=5)
self.graphFrame.pack_propagate(False)
```

5. Kód definující widgety, jejich vlastnosti, pozicování a proměnné

```
# var kde vykreslit graf
self.displayOptVar = tk.StringVar(value="program")
self.displayOpt1 = tk.Radiobutton(self.displayFrame, text="V Programu", font=(None, 12), variable=self.displayOptVar, value="program")
self.displayOpt1.pack(side=tk.LEFT, padx=20)
self.displayOpt2 = tk.Radiobutton(self.displayFrame, text="Samostatně", font=(None, 12), variable=self.displayOptVar, value="standalone")
self.displayOpt2.pack(side=tk.LEFT, padx=20)

## Options Frame - graph selections
self.selectionFrame = tk.Frame(self.optionsFrame)
self.selectionFrame.pack(pady=50)
self.selectionFrame.place(relx=.45, rely=.45, anchor=tk.CENTER)


# Graph selections left side
self.labelA = tk.Label(self.selectionFrame, text="Hodnota a:")
self.labelA.grid(row=0, column=0, padx=10, pady=15)
self.entryA = tk.Entry(self.selectionFrame)
self.entryA.insert(tk.END, 1)
self.entryA.grid(row=0, column=1, padx=10, pady=15)
self.labelB = tk.Label(self.selectionFrame, text="Hodnota b:")
self.labelB.grid(row=1, column=0, padx=10, pady=15)
self.entryB = tk.Entry(self.selectionFrame)
self.entryB.insert(tk.END, 1)
self.entryB.grid(row=1, column=1, padx=10, pady=15)
self.labelC = tk.Label(self.selectionFrame, text="Hodnota c [°]:")
self.labelC.grid(row=2, column=0, padx=10, pady=15)
self.entryC = tk.Entry(self.selectionFrame)
self.entryC.insert(tk.END, 0)
self.entryC.grid(row=2, column=1, padx=10, pady=15)
self.labelD = tk.Label(self.selectionFrame, text="Hodnota d:")
self.labelD.grid(row=3, column=0, padx=10, pady=15)
self.entryD = tk.Entry(self.selectionFrame)
self.entryD.insert(tk.END, 0)
self.entryD.grid(row=3, column=1, padx=10, pady=15)
self.labelTypeCBox = tk.Label(self.selectionFrame, text="Typ grafu:")
self.labelTypeCBox.grid(row=4, column=0, padx=10, pady=15)
self.typeCBox = ttk.Combobox(self.selectionFrame, values=("sin", "cos", "tg", "cotg"))
self.typeCBox.current(0)
self.typeCBox.grid(row=4, column=1, padx=10, pady=15)
```

```
# Graph selections right side
self.labelXAxis = tk.Label(self.selectionFrame, text="Osa X:")
self.labelXAxis.grid(row=0, column=2, padx=10, pady=15)
self.entryXAxis = tk.Entry(self.selectionFrame)
self.entryXAxis.grid(row=0, column=3, padx=10, pady=15)
self.labelYAxis = tk.Label(self.selectionFrame, text="Osa Y:")
self.labelYAxis.grid(row=1, column=2, padx=10, pady=15)
self.entryYAxis = tk.Entry(self.selectionFrame)
self.entryYAxis.grid(row=1, column=3, padx=10, pady=15)
self.labelName = tk.Label(self.selectionFrame, text="Název:")
self.labelName.grid(row=2, column=2, padx=10, pady=15)
self.entryName = tk.Entry(self.selectionFrame)
self.entryName.grid(row=2, column=3, padx=10, pady=15)
self.labelLineWidth = tk.Label(self.selectionFrame, text="Tloušťka čar:")
self.labelLineWidth.grid(row=3, column=2, padx=10, pady=15)
self.entryLineWidth = tk.Entry(self.selectionFrame)
self.entryLineWidth.insert(tk.END, 1)
self.entryLineWidth.grid(row=3, column=3, padx=10, pady=15)
self.gridOptVar = tk.IntVar(value=0)
self.labelGrid = tk.Label(self.selectionFrame, text="Mřížka:")
self.labelGrid.grid(row=4, column=2, padx=10, pady=15)
self.gridChBox = tk.Checkbutton(self.selectionFrame, variable=self.gridOptVar)
self.gridChBox.grid(row=4, column=3, padx=10, pady=15)

## Options Frame - action group
self.actionFrame = tk.Frame(self.optionsFrame)
self.actionFrame.pack(side=tk.BOTTOM, fill=tk.X, pady=20)
self.drawBtn = tk.Button(self.actionFrame, text="Vykreslit", font=(None, 13), command=self.draw)
self.drawBtn.grid(row=0, column=1, pady=10, sticky=tk.NSEW)
self.equasionLabel = tk.Label(self.actionFrame, text="Funkce:")
self.equasionLabel.grid(row=1, column=0, padx=10)
self.equasionDisplay = tk.Text(self.actionFrame, state='disabled', width=30, height=1)
self.equasionDisplay.grid(row=1, column=1)

## Graph Frame - graph
self.graphFigure = Figure(figsize=(5, 5), dpi=100)
self.graphSubplot = self.graphFigure.add_subplot(111)
self.graphCanvas = FigureCanvasTkAgg(self.graphFigure, master=self.graphFrame)
self.graphToolbar = NavigationToolbar2Tk(self.graphCanvas, self.graphFrame)
```

6. Funkce na vypnutí okna a validaci vstupů

```
def quit(self, event=None):
    super().quit()

def entryValidate(self, check):
    for item in check:
        try:
            float(item)
        except:
            return False
    return True
```

7. Funkce na zpracování vstupů a jejich vykreslení

```
def draw(self, *arg):
    check = []
    valA = self.entryA.get()
    valB = self.entryB.get()
    valC = self.entryC.get()
    valD = self.entryD.get()
    check.append(valA)
    check.append(valB)
    check.append(valC)
    check.append(valD)
    valAction = self.typeCBox.get()
    display = self.equasionDisplay

    nameX = self.entryXAxis.get()
    nameY = self.entryYAxis.get()
    name = self.entryName.get()
    lineWidth = self.entryLineWidth.get()
    check.append(lineWidth)
    grid = self.gridOptVar.get()
    where = self.displayOptVar.get()

    if valAction == "sin":
        func = "y = {0} * sin({1} + {2}") + {3}".format(valA, valB, valC, valD)
        query = 1
    elif valAction == "cos":
        func = "y = {0} * cos({1} + {2}") + {3}".format(valA, valB, valC, valD)
        query = 1
    elif valAction == "tg":
        func = "y = {0} * tg({1} + {2}") + {3}".format(valA, valB, valC, valD)
        query = 1
    elif valAction == "cotg":
        func = "y = {0} * cotg({1} + {2}") + {3}".format(valA, valB, valC, valD)
        query = 1
    else:
        func = "Chybí ti hodnoty!"
        query = 0

    display.config(state="normal")
    display.delete('1.0', tk.END)
    display.insert(tk.INSERT, func)
    display.config(state="disabled")

    if self.entryValidate(check) == True:
        pass
    else:
        display.config(state="normal")
        display.delete('1.0', tk.END)
        display.insert(tk.INSERT, "Zapsali jste špatné hodnoty!")
        display.config(state="disabled")
        return False


    if query != 0:

        self.graphSubplot.clear()

        valA = float(valA)
        valB = float(valB)
        valC = float(valC)
        valD = float(valD)
        if where == "standalone":
            if valAction == "sin":
                x = np.arange(0, 4 * np.pi, 0.1)   # vždy nechat 0, počet period
                y = valA * np.sin(valB * x + valC)
                plt.plot(x, y + valD, linewidth=lineWidth)
            elif valAction == "cos":
                x = np.arange(0, 4 * np.pi, 0.1)
                y = valA * np.cos(valB * x + valC)
                plt.plot(x, y + valD, linewidth=lineWidth)
            elif valAction == "tg":
                x = np.linspace(-2 * np.pi, 0 * np.pi, 1000) # vždy nechat -2, p
                y = valA * np.tan(valB * x + valC)
                plt.plot(x, y + valD, linewidth=lineWidth)
                plt.ylim(-10, 10)
            elif valAction == "cotg":
                x = np.linspace(-2 * np.pi, 0 * np.pi, 1000)
                y = 1 / (valA * np.tan(valB * x + valC))
                plt.plot(x, y + valD, linewidth=lineWidth)
                plt.ylim(-10, 10)

            plt.title(name)
            plt.xlabel(nameX)
            plt.ylabel(nameY)
            if grid == 1:
                plt.grid()
            plt.show()
```

```
else:
    if valAction == "sin":
        x = np.arange(0, 4 * np.pi, 0.1)   # vždy nechat 0, počet period (2 = 1T),
        y = valA * np.sin(valB * x + valC)
        self.graphSubplot.plot(x, y + valD, linewidth=lineWidth)
        self.graphSubplot.set_title(name)
        self.graphSubplot.set_xlabel(nameX)
        self.graphSubplot.set_ylabel(nameY)
        if grid == 1:
            self.graphSubplot.grid()
        self.graphCanvas.draw()
        self.graphCanvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)
        self.graphToolbar.update()
        self.graphCanvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)
    elif valAction == "cos":
        x = np.arange(0, 4 * np.pi, 0.1)
        y = valA * np.cos(valB * x + valC)
        self.graphSubplot.plot(x, y + valD, linewidth=lineWidth)
        self.graphSubplot.set_title(name)
        self.graphSubplot.set_xlabel(nameX)
        self.graphSubplot.set_ylabel(nameY)
        if grid == 1:
            self.graphSubplot.grid()
        self.graphCanvas.draw()
        self.graphCanvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)
        self.graphToolbar.update()
        self.graphCanvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)
    elif valAction == "tg":
        x = np.linspace(-2 * np.pi, 0 * np.pi, 1000) # vždy nechat -2, počet period
        y = valA * np.tan(valB * x + valC)
        self.graphSubplot.plot(x, y + valD, linewidth=lineWidth)
        if valA > 10:
            self.graphSubplot.set_ylim([-valA, valA])
        else:
            self.graphSubplot.set_ylim([-10, 10])
        self.graphSubplot.set_title(name)
        self.graphSubplot.set_xlabel(nameX)
        self.graphSubplot.set_ylabel(nameY)
        if grid == 1:
            self.graphSubplot.grid()
        self.graphCanvas.draw()
        self.graphCanvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)
        self.graphToolbar.update()
        self.graphCanvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)
    elif valAction == "cotg":
        x = np.linspace(-2 * np.pi, 0 * np.pi, 1000)
        y = 1 / (valA * np.tan(valB * x + valC))
        self.graphSubplot.plot(x, y + valD, linewidth=lineWidth)
        if valA > 10:
            self.graphSubplot.set_ylim([-valA, valA])
        else:
            self.graphSubplot.set_ylim([-10, 10])
        self.graphSubplot.set_title(name)
        self.graphSubplot.set_xlabel(nameX)
        self.graphSubplot.set_ylabel(nameY)
        if grid == 1:
            self.graphSubplot.grid()
        self.graphCanvas.draw()
        self.graphCanvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)
        self.graphToolbar.update()
        self.graphCanvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)
```

Závěr:

Vytváření projektu proběhlo úspěšně bez větších komplikací. Největší použité knihovny jsou Tkinter a matplotlib.

Funkce „draw()“ je nejdůležitější funkcí celého kódu. Tato funkce zpracovává proměnné a nastavuje hodnoty pro následné vykreslení grafu (také ve funkci „draw()“).

Pro vykreslování grafů v okně programu byl použit modul TkAgg knihovny python3-pil. Pro vykreslení grafu vytváříme tzv. „subplot“ což je podřadný element okna vytvořeným knihovny matplotlib (nyní jen mainplot). Tyto „subploty“ následně můžeme mazat bez nutnosti vypnutí „instance“ mainplotu, čímž zabráníme špatnému zobrazování grafů.