1. Для каждого задания создайте отдельную вкладку.

2. Разработать графический интерфейс для ввода интервала значений Х для

построения графиков функций (см. лабораторная работа 5, п.1). Для выбора функции

использовать Combobox.

3. Разработать форму для изменения данных в XML-файле, созданного в лабораторной

работе №3, п.1. Поиск абитуриента осуществлять по фамилии или шифру

абитуриента (использовать Radiobutton). Для выбора вступительного(ых)

испытания(й), по которому(ым) нужно внести изменения, используйте Checkbutton.

Для подтверждения корректности внесенных пользователем данных используйте

MessageBox.

4. Разработать форму для добавления новых данных в XML-файл (лабораторная

работа №3, п.1)

code:

from tkinter import \*

from tkinter import ttk

from tkinter.ttk import Combobox

from tkinter.ttk import Radiobutton

from tkinter.ttk import Checkbutton

import xml.etree.ElementTree as ET

import matplotlib.pyplot as plt

import numpy as np

def Debug():

    print('works!')

window = Tk()

#tabs

tabControl = ttk.Notebook(window)

tab1 = ttk.Frame(tabControl)

tab2 = ttk.Frame(tabControl)

tab3 = ttk.Frame(tabControl)

tabControl.add(tab1, text ='Tab 1')

tabControl.add(tab2, text ='Tab 2')

tabControl.add(tab3, text ='Tab 3')

tabControl.pack(expand = 1, fill ="both")

#2222222222222222222222222222222222222

def BuildGraph(xmin, xmax, funct):

    x = np.arange(xmin, xmax, round((xmax-xmin)/100, 2))

    match funct:

        case "x": y = x

        case 'x^2': y = x\*\*2

        case "x\*\*2 - 8\*x + 15": y = x\*\*2 - 8\*x + 15

        case '(x\*\*2 - 6\*x + 10)\*(x>=1) + (x + 2)\*(x<1)': y = (x\*\*2 - 6\*x + 10)\*(x>=1) + (x + 2)\*(x<1)

    plt.plot(x,y)

    plt.show()

    print (xmin, xmax, funct)

def clicked():

    try: xmin = int(xmin\_.get())

    except: xmin = 0

    try: xmax = int(xmax\_.get())

    except: xmax = 0

    funct = funct\_.get()

    BuildGraph(xmin, xmax, funct)

xmin\_ = Entry(tab1, width = 10)

xmin\_.grid(column=1, row=0)

xmax\_ = Entry(tab1, width = 10)

xmax\_.grid(column=2, row=0)

funct\_ = Combobox(tab1, values=('x', 'x^2', 'x\*\*2 - 8\*x + 15', '(x\*\*2 - 6\*x + 10)\*(x>=1) + (x + 2)\*(x<1)'))

funct\_.current(0)

funct\_.grid(column = 3, row=0)

xmin\_btn = Button(tab1, command=clicked)

xmin\_btn.grid(column=4, row=0)

#3333333333333333333333333333333333333

tree = ET.parse('лаб 6/1.xml')

root = tree.getroot()

def EditStudent():

    for i in entryesOnEdit:

        try:

            if int(i.get()) < 0 or int(i.get()) > 100:

                lblOnEdit.configure(text='invalid score')

                return

        except: pass

    for enrolle in root.findall(f"enrolle[name='{students\_names[selected.get()]}']"):

        for i in range(len(exam\_names)):

            try:

                enrolle.find(exam\_names[i]).text

            except:

                new\_exam = ET.Element(exam\_names[i])

                root.find(f"enrolle[name='{students\_names[selected.get()]}']").append(new\_exam)

                tree.write('лаб 6/1.xml', encoding="UTF-8")

            if checkbuttons\_stateOnEdit[i].get(): enrolle.find(exam\_names[i]).text = entryesOnEdit[i].get()

    tree.write('лаб 6/1.xml', encoding="UTF-8")

    lblOnEdit.configure(text='done')

def get\_names():

    global students\_names

    global exam\_names

    students\_names = list({elem.findtext('name') for elem in root.iter()})

    students\_names.remove(None)

    students\_names.sort()

    exam\_names = list({elem.tag for elem in root.findall('enrolle/\*')})

    exam\_names.remove('name')

    exam\_names.sort()

get\_names()

radiobuttonsOnEdit = []

selected = IntVar()

for i in range(len(students\_names)):

    radiobuttonsOnEdit.append(Radiobutton(tab2, text = students\_names[i], value = i, variable=selected))

    radiobuttonsOnEdit[i].grid(column = 0, row = i)

checkbuttons\_stateOnEdit = []

checkbuttonsOnEdit = []

entryesOnEdit = []

for i in range(len(exam\_names)):

    checkbuttons\_stateOnEdit.append(BooleanVar())

    checkbuttonsOnEdit.append(Checkbutton(tab2, text=exam\_names[i], var = checkbuttons\_stateOnEdit[i]))

    checkbuttonsOnEdit[i].grid(column = 1, row = i)

    entryesOnEdit.append(Entry(tab2, width = 10))

    entryesOnEdit[i].grid(column = 2, row = i)

submit\_btn = Button(tab2, command=EditStudent, width=10)

submit\_btn.grid(column=0, row=len(students\_names))

lblOnEdit = Label(tab2,text='waiting input')

lblOnEdit.grid(column=0, row=len(students\_names)+1)

#4444444444444444444444444444444444444

def AddStudent():

    if student\_name.get() in students\_names:

        lblOnAdd.configure(text='student already exists')

        return

    for i in entryes:

        try:

            if int(i.get()) < 0 or int(i.get()) > 100:

                lblOnAdd.configure(text='invalid score')

                return

        except: pass

    new\_student = ET.Element('enrolle')

    new\_student.set('id', str(len(students\_names) + 1))

    ET.SubElement(new\_student, 'name').text = student\_name.get()

    for i in range(len(exam\_names)):

        if checkbuttons\_state[i].get():

            print("works")

            ET.SubElement(new\_student, exam\_names[i]).text = entryes[i].get()

    root.append(new\_student)

    tree.write('лаб 6/1.xml', encoding="UTF-8")

    get\_names()

    lblOnAdd.configure(text='done')

get\_names()

student\_name = Entry(tab3, width = 10)

student\_name.grid(column=1, row = 0)

student\_name\_lbl = Label(tab3, text = 'student\_name').grid(column=0, row=0)

checkbuttons\_state = []

checkbuttons = []

entryes = []

for i in range(len(exam\_names)):

    checkbuttons\_state.append(BooleanVar())

    checkbuttons.append(Checkbutton(tab3, text=exam\_names[i], var = checkbuttons\_state[i]))

    checkbuttons[i].grid(column = 0, row = i + 2)

    entryes.append(Entry(tab3, width = 10))

    entryes[i].grid(column = 1, row = i + 2)

submit\_btn = Button(tab3, command=AddStudent, width=10)

submit\_btn.grid(column=0, row=len(exam\_names)+2)

lblOnAdd = Label (tab3, text='waiting input')

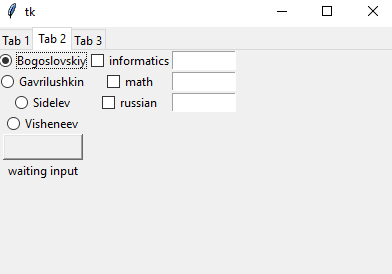
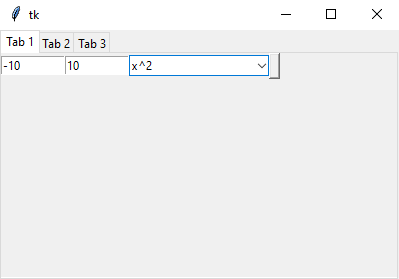
lblOnAdd.grid(column=0, row=len(exam\_names)+3)

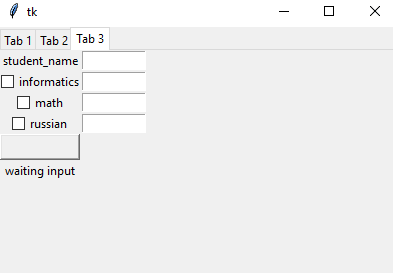
#draw window

window.geometry('400x250')

window.mainloop()

result:





xml:

<enrolles>

    <enrolle id="1">

        <name>Gavrilushkin</name>

        <math>74</math>

        <russian>78</russian>

        <informatics>123</informatics>

    </enrolle>

    <enrolle id="2">

        <name>Sidelev</name>

        <math />

        <russian />

        <informatics>1</informatics>

    </enrolle>

    <enrolle id="3">

        <name>Visheneev</name>

        <math>50</math>

        <russian>50</russian>

        <informatics>50</informatics>

    </enrolle>

    <enrolle id="4">

        <name>Bogoslovskiy</name>

        <math />

        <russian />

        <informatics>1</informatics>

    </enrolle>

</enrolles>