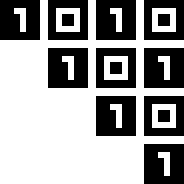
Министерство науки и высшего образования РФ

ФГБОУ ВО «Тамбовский государственный технический университет»

Кафедра

«САПР»



Отчет по лабораторной работе №1

по дисциплине «Офисные технологии»

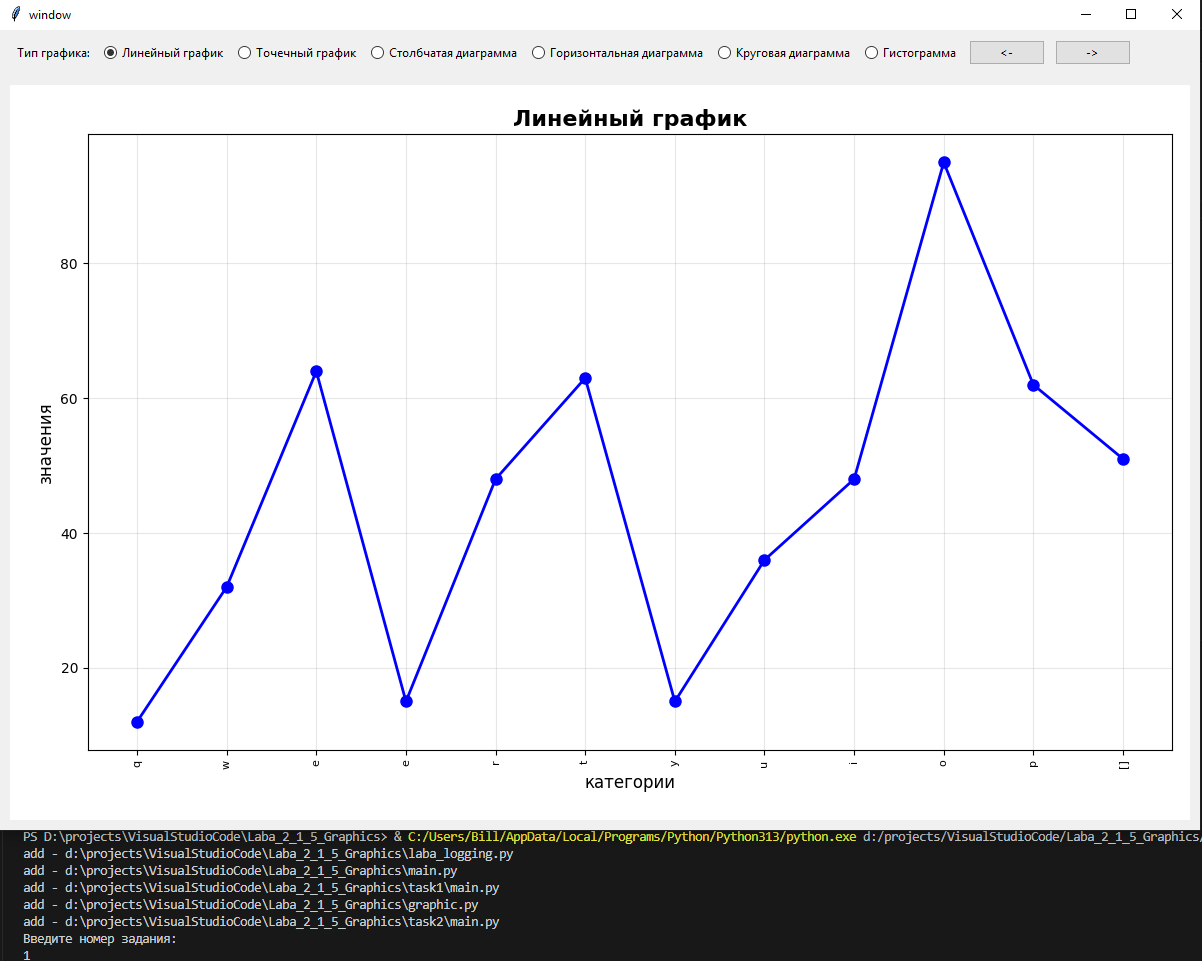
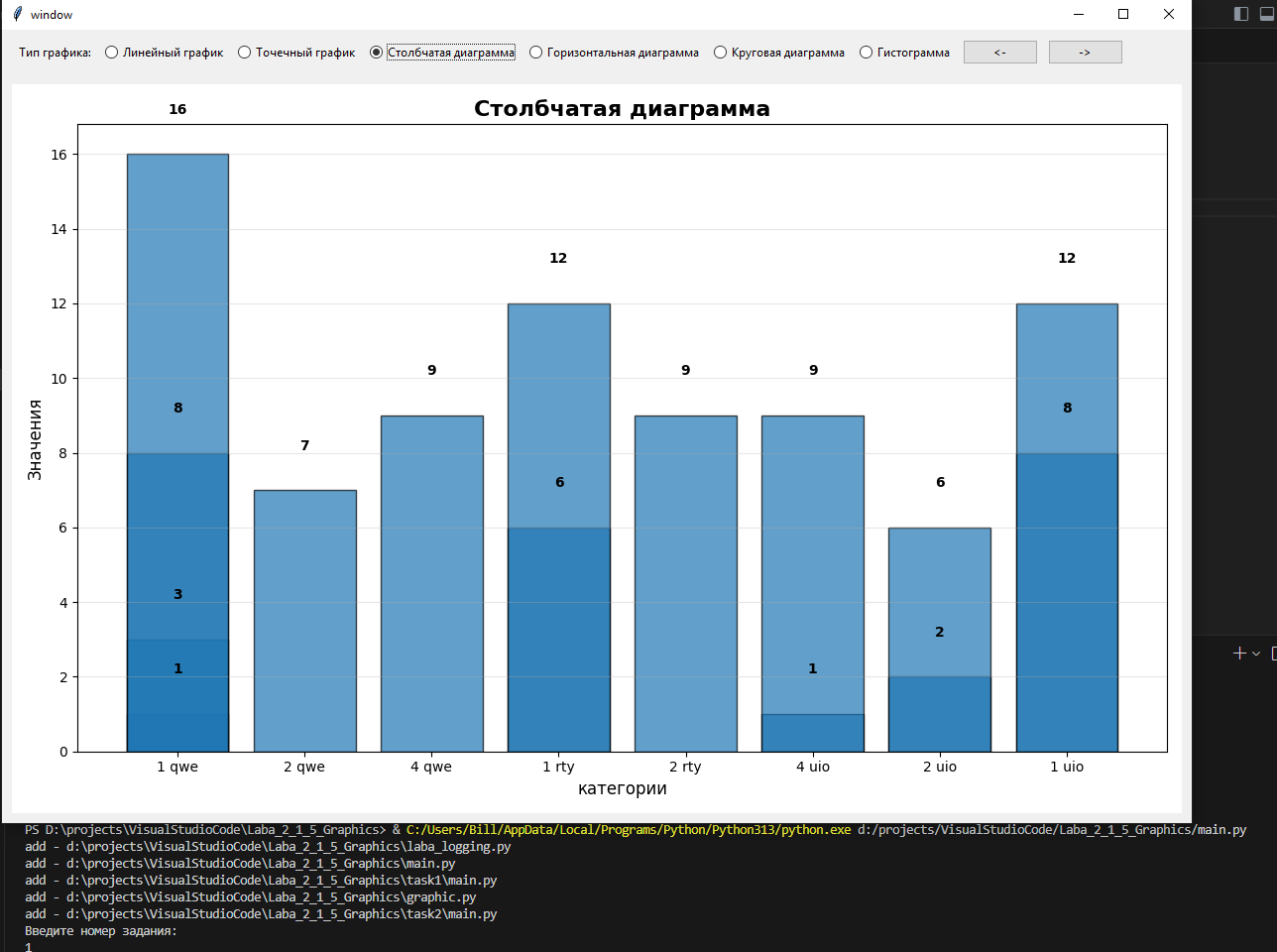
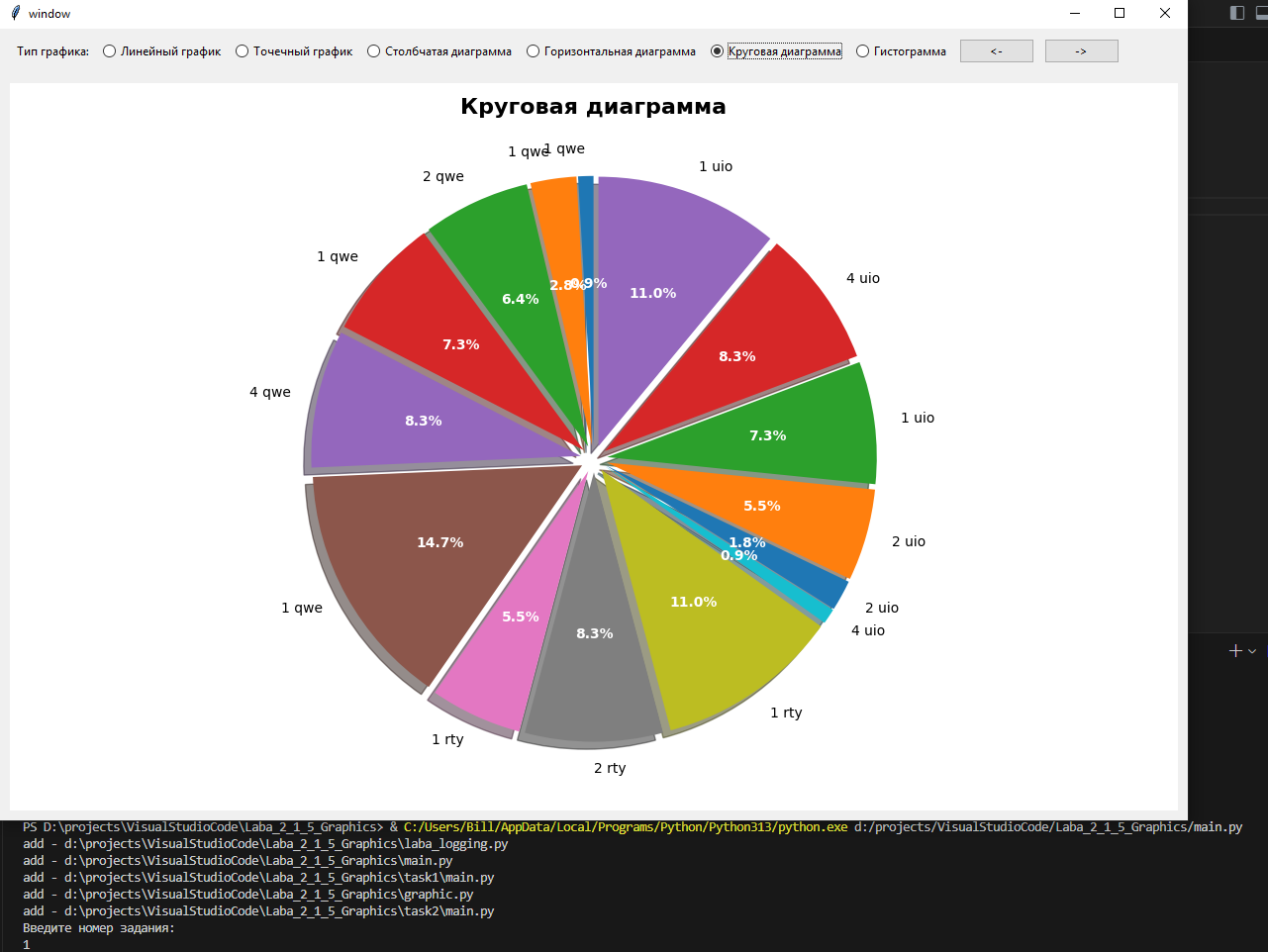
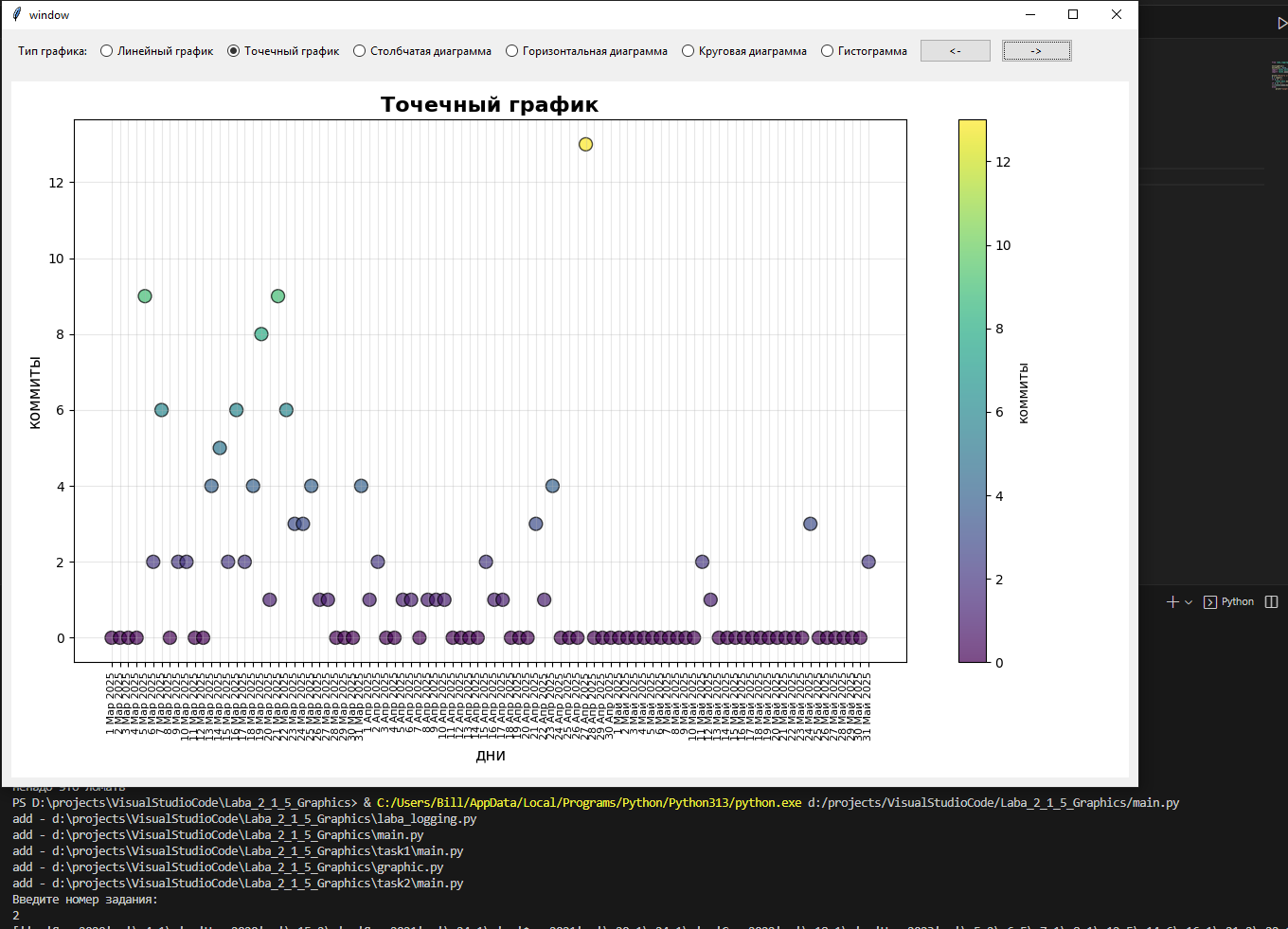
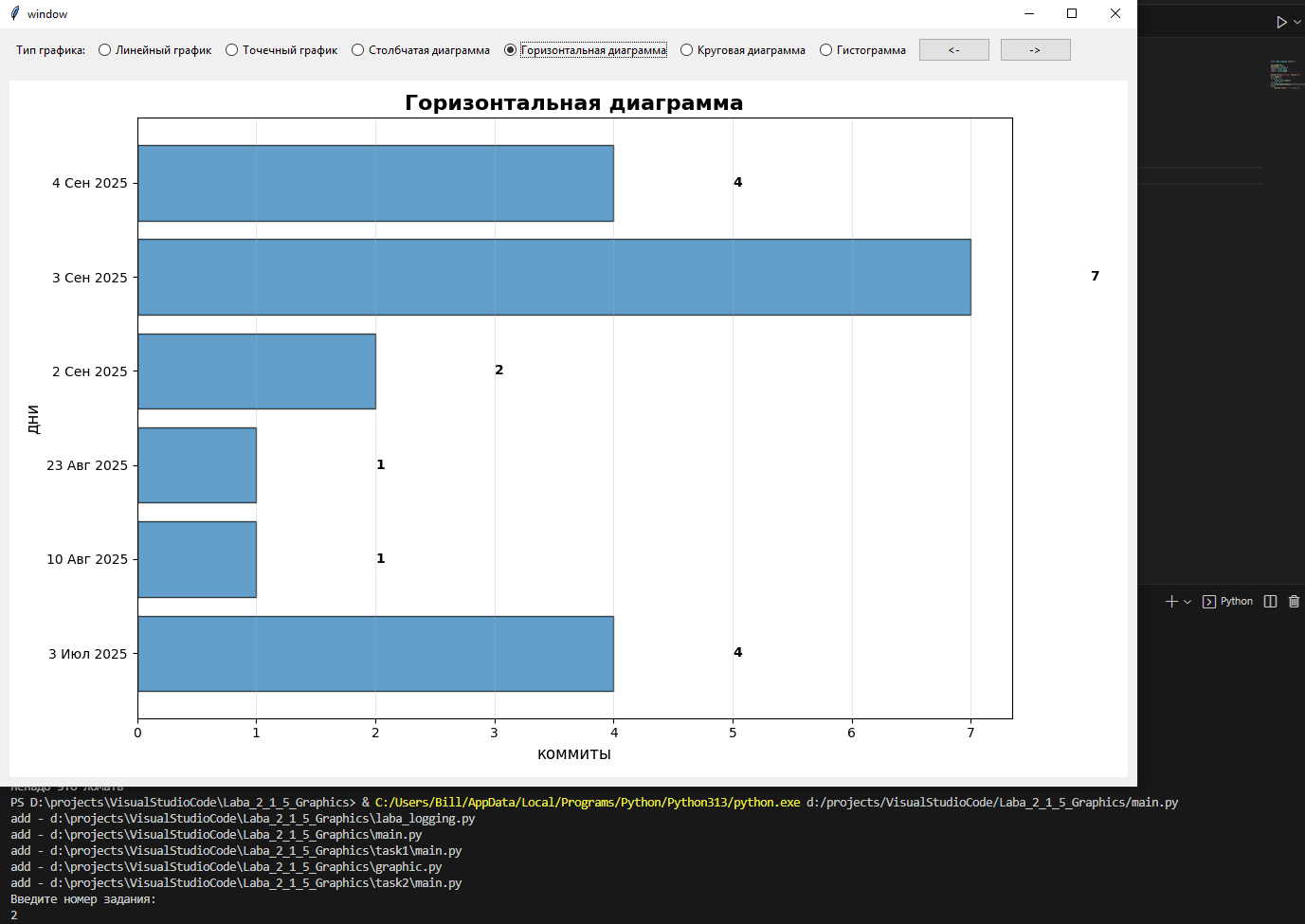
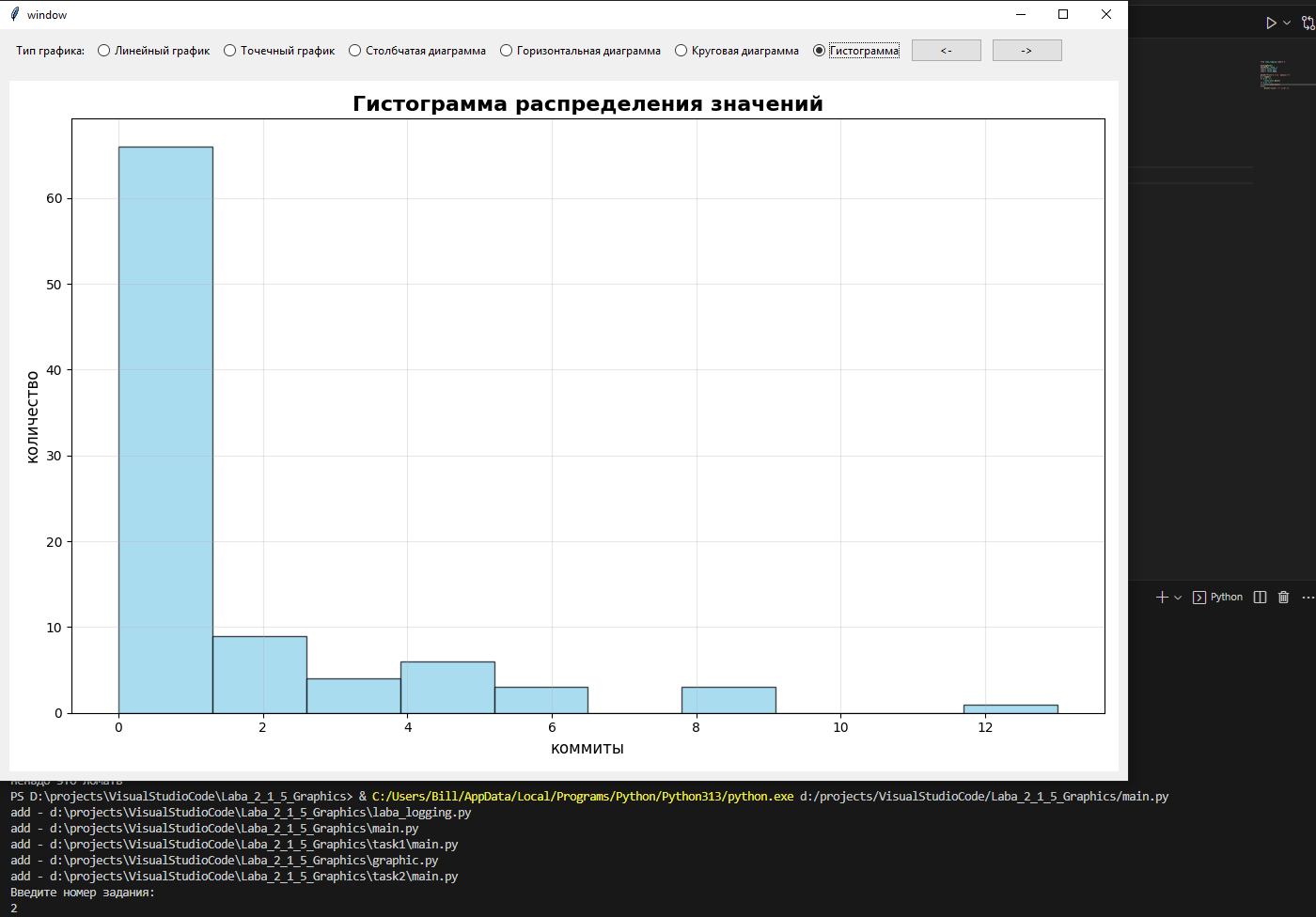
Выполнил: студент группы БВТ241

Белов А.К.

Проверил: преподаватель

Патутин К. И.

Тамбов 2025

**Цель**  
Изучить построение графиков с помощью библиотеки matplotlib, произвести анализ полеученых данных.  
  
 **Задание 1**  
Разработать программу, в которой с помощью библиотеки matplotlib будут построены графики различных типов  
 **Задание 2**  
Проанализировать один из предложенных вариантов данных, либо собственные (Счета коммунальных услуг, данные с датчиков умного дома, данные датчиков экг, данные о расходе топлива автомобиля и т.д.) взятые из реальных источников и на их основе построить сравнительные графики, подробно описать, что это за данные, как они сравнивались и т.п.  
  
 **Решение**  
Используя matpolib и несколько сторонних библиотек создал класс окна имеющий некоторые настройки которые позволяют его использовать для отслечивания частоты коммитов (они записаны в .txt вручную).   
 **Листинг:  
  
Из файла ...\_Graphics\laba\_logging.py.cpp**import os  
import time  
  
LOG\_FILE = 'loggs.txt'   
inited = []  
def InitLogFile():  
 with open(LOG\_FILE, "w", encoding="utf-8") as f:  
 # fcntl.flock(f, fcntl.LOCK\_EX) # Блокировка файла  
 f.write("=== Start logging ===\n")  
 InitFile(\_\_file\_\_)  
 # fcntl.flock(f, fcntl.LOCK\_UN) # Разблокировка  
  
  
def InitFile(filename):  
 with open(LOG\_FILE, "a+", encoding="utf-8") as f:  
 print("add - "+filename)  
 if filename not in inited:  
 f.write("-init|" + str(filename) + "\n")   
 inited.append(filename)  
 time.sleep(0.1)  
  **Из файла ...ba\_2\_1\_5\_Graphics\main.py.cpp**  
from laba\_logging import \*  
  
InitLogFile()  
InitFile(\_\_file\_\_)  
import task1.main  
import task2.main  
  
print("Введите номер задания:")  
a = input()  
if a == '1':  
 task1.main.main()  
if a == '2':  
 task2.main.main()  
else:  
 print("ненадо это ломать")  
  
 **Из файла ...\_5\_Graphics\task1\main.py.cpp**import sys  
import os  
  
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_))))  
from laba\_logging import \*  
InitFile(\_\_file\_\_)  
  
  
import matplotlib.pyplot as plt  
import numpy as np  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
import tkinter as tk  
from tkinter import ttk  
from graphic import MultiPlotApp  
  
  
def main():  
 root = tk.Tk()  
 app = MultiPlotApp(root, "window", "D:\\projects\\VisualStudioCode\\Laba\_2\_1\_5\_Graphics\\task1\\task1\_data.txt")  
 root.mainloop()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main() **Из файла ...2\_1\_5\_Graphics\graphic.py.cpp**  
import matplotlib.pyplot as plt  
import numpy as np  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
import tkinter as tk  
from tkinter import ttk  
  
import sys  
import os  
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_))))  
from laba\_logging import \*  
InitFile(\_\_file\_\_)  
  
class MultiPlotApp:  
 def \_\_init\_\_(self, root, name:str, data):  
 self.root = root  
 self.root.title(name)  
 self.root.geometry("1200x800")  
  
  
 self.categories\_name = 'категории'  
 self.values\_name = 'значения'  
  
 # self.colors = ['#FF6B6B', '#4ECDC4', '#45B7D1', '#F9A826', '#6C5CE7']  
 self.categories = '0'  
 self.values = 1  
  
 self.current\_index = 1  
 self.current\_index\_s = 1  
 self.categories\_list = [['q', 'w', 'e', 'e'], ['r', 't', 'y', 'u'], ['i', 'o', 'p', '[]'], ['a', 's', 'd', 'f'], ['g', 'h', 'j', 'k']]  
 self.values\_list = [[12, 32, 64, 15], [48, 63, 15, 36], [48, 95, 62, 51], [42, 51, 54, 36], [75, 35, 42, 63]]  
 self.categories\_list\_without\_null = self.categories\_list  
 self.values\_list\_without\_null = self.values\_list  
 # self.switch\_data()   
 self.create\_widgets()  
 self.create\_plots()  
 # Пример данных  
 #   
 #   
  
 #data изъятие  
 with open(data, "r", encoding="utf-8") as f:  
 read\_data = f.read().split('-')  
 print(read\_data)  
 self.categories\_list.clear()  
 self.values\_list.clear()  
 for i in range(1, len(read\_data), 2):  
 part\_data\_categories = []  
 part\_data\_values = []  
 print(read\_data[i] + " - " + read\_data[i+1])  
  
 for items in (read\_data[i+1].split('\n')):  
 item = items.split(' ')  
 if(items == ''):  
 continue  
  
 print(item)  
 print(f'|{item[0]}|{item[1]}|')  
  
 part\_data\_categories.append(item[0] + " " + read\_data[i])  
 part\_data\_values.append(int(item[1]))  
 self.categories\_list.append(part\_data\_categories)  
 self.values\_list.append(part\_data\_values)  
 print("all ------------")  
 print(part\_data\_categories)  
 print(part\_data\_values)  
   
  
 def set\_categories(self, data):  
 self.categories = data  
  
 def set\_values(self, data):  
 self.values = data  
  
 def create\_widgets(self):  
 control\_frame = ttk.Frame(self.root, padding="10")  
 control\_frame.grid(row=0, column=0, sticky="ew")  
   
 plot\_frame = ttk.Frame(self.root, padding="10")  
 plot\_frame.grid(row=1, column=0, sticky="nsew")  
   
 self.root.grid\_rowconfigure(1, weight=1)  
 self.root.grid\_columnconfigure(0, weight=1)  
   
 ttk.Label(control\_frame, text="Тип графика:").grid(row=0, column=0, padx=5)  
   
 self.plot\_type = tk.StringVar(value="linear")  
 plot\_types = [  
 ("Линейный график", "linear"),  
 ("Точечный график", "scatter"),  
 ("Столбчатая диаграмма", "bar"),  
 ("Горизонтальная диаграмма", "barh"),  
 ("Круговая диаграмма", "pie"),  
 ("Гистограмма", "hist")  
 ]  
   
 for i, (text, value) in enumerate(plot\_types):  
 ttk.Radiobutton(control\_frame, text=text, variable=self.plot\_type,   
 value=value, command=self.update\_plot).grid(row=0, column=i+1, padx=5)  
   
 ttk.Button(control\_frame, text="<-", command=self.switch\_data\_l).grid(row=0, column=len(plot\_types)+1, padx=5)   
 ttk.Button(control\_frame, text="->", command=self.switch\_data\_r).grid(row=0, column=len(plot\_types)+2, padx=5)  
  
 self.figure = plt.Figure(figsize=(10, 6), dpi=100)  
 self.canvas = FigureCanvasTkAgg(self.figure, plot\_frame)  
 self.canvas.get\_tk\_widget().pack(fill=tk.BOTH, expand=True)  
   
 def create\_plots(self):  
 self.update\_plot()  
   
 def switch\_data\_l(self):  
 self.current\_index = max(self.current\_index - 1, 1)  
 self.current\_index\_s = max(self.current\_index\_s - 1, 1)  
 self.update\_plot()  
  
 def switch\_data\_r(self):  
 self.current\_index = min(self.current\_index + 1, len(self.categories\_list)-2)  
 self.current\_index\_s = min(self.current\_index + 1, len(self.categories\_list\_without\_null)-2)  
 self.update\_plot()  
  
  
 def switch\_data(self):  
 self.set\_categories(self.categories\_list[self.current\_index-1]+self.categories\_list[self.current\_index]+self.categories\_list[self.current\_index+1])  
 self.set\_values(self.values\_list[self.current\_index-1]+self.values\_list[self.current\_index]+self.values\_list[self.current\_index+1])  
  
  
 def update\_plot(self):  
  
 self.switch\_data()  
 plot\_type = self.plot\_type.get()  
 self.figure.clear()  
   
 ax = self.figure.add\_subplot(111)  
   
 if plot\_type == "linear":  
 self.create\_linear\_plot(ax)  
 elif plot\_type == "scatter":  
 self.create\_scatter\_plot(ax)  
 elif plot\_type == "bar":  
 self.set\_categories(self.categories\_list\_without\_null[self.current\_index\_s-1]+self.categories\_list\_without\_null[self.current\_index\_s]+self.categories\_list\_without\_null[self.current\_index\_s+1])  
 self.set\_values(self.values\_list\_without\_null[self.current\_index\_s-1]+self.values\_list\_without\_null[self.current\_index\_s]+self.values\_list\_without\_null[self.current\_index\_s+1])  
 self.create\_bar\_plot(ax)  
 elif plot\_type == "barh":  
 self.set\_categories(self.categories\_list\_without\_null[self.current\_index\_s-1]+self.categories\_list\_without\_null[self.current\_index\_s]+self.categories\_list\_without\_null[self.current\_index\_s+1])  
 self.set\_values(self.values\_list\_without\_null[self.current\_index\_s-1]+self.values\_list\_without\_null[self.current\_index\_s]+self.values\_list\_without\_null[self.current\_index\_s+1])  
 self.create\_barh\_plot(ax)  
 elif plot\_type == "pie":  
 self.set\_categories(self.categories\_list\_without\_null[self.current\_index\_s-1]+self.categories\_list\_without\_null[self.current\_index\_s]+self.categories\_list\_without\_null[self.current\_index\_s+1])  
 self.set\_values(self.values\_list\_without\_null[self.current\_index\_s-1]+self.values\_list\_without\_null[self.current\_index\_s]+self.values\_list\_without\_null[self.current\_index\_s+1])  
 self.create\_pie\_plot(ax)  
 elif plot\_type == "hist":  
 self.create\_hist\_plot(ax)  
   
 self.figure.tight\_layout()  
 self.canvas.draw()  
   
 def create\_linear\_plot(self, ax):  
 x = np.arange(len(self.values))  
 ax.plot(x, self.values, marker='o', linestyle='-', linewidth=2, markersize=8, color='blue')  
 ax.set\_title('Линейный график', fontsize=16, fontweight='bold')  
 ax.set\_xlabel(self.categories\_name, fontsize=12)  
 ax.set\_ylabel(self.values\_name, fontsize=12)  
 ax.grid(True, alpha=0.3)  
 ax.set\_xticks(x)  
 # ax.set\_xticklabels(self.categories)  
 ax.set\_xticklabels(self.categories, rotation=90, ha='center', fontsize=8)  
  
 def create\_scatter\_plot(self, ax):  
 x = np.arange(len(self.values))  
 scatter = ax.scatter(x, self.values, s=100, c=self.values, cmap='viridis', alpha=0.7, edgecolors='black')  
 ax.set\_title('Точечный график', fontsize=16, fontweight='bold')  
 ax.set\_xlabel(self.categories\_name, fontsize=12)  
 ax.set\_ylabel(self.values\_name, fontsize=12)  
 ax.grid(True, alpha=0.3)  
 ax.set\_xticks(x)  
 # ax.set\_xticklabels(self.categories)  
 ax.set\_xticklabels(self.categories, rotation=90, ha='center', fontsize=8)  
  
 plt.colorbar(scatter, ax=ax, label=self.values\_name)  
   
 def create\_bar\_plot(self, ax):  
 bars = ax.bar(self.categories, self.values, alpha=0.7, edgecolor='black')  
 ax.set\_title('Столбчатая диаграмма', fontsize=16, fontweight='bold')  
 ax.set\_xlabel(self.categories\_name, fontsize=12)  
 ax.set\_ylabel('Значения', fontsize=12)  
 ax.grid(True, alpha=0.3, axis='y')  
   
 for bar, value in zip(bars, self.values):  
 height = bar.get\_height()  
 ax.text(bar.get\_x() + bar.get\_width()/2., height + 1,  
 f'{value}', ha='center', va='bottom', fontweight='bold')  
   
 def create\_barh\_plot(self, ax):  
 bars = ax.barh(self.categories, self.values, alpha=0.7, edgecolor='black')  
 ax.set\_title('Горизонтальная диаграмма', fontsize=16, fontweight='bold')  
 ax.set\_xlabel(self.values\_name, fontsize=12)  
 ax.set\_ylabel(self.categories\_name, fontsize=12)  
 ax.grid(True, alpha=0.3, axis='x')  
   
 for bar, value in zip(bars, self.values):  
 width = bar.get\_width()  
 ax.text(width + 1, bar.get\_y() + bar.get\_height()/2.,  
 f'{value}', ha='left', va='center', fontweight='bold')  
   
 def create\_pie\_plot(self, ax):  
 wedges, texts, autotexts = ax.pie(  
 self.values,   
 labels=self.categories,   
 autopct='%1.1f%%',  
 startangle=90,  
 shadow=True,  
 explode=[0.05] \* len(self.values)  
 )  
   
 ax.set\_title('Круговая диаграмма', fontsize=16, fontweight='bold')  
   
 for autotext in autotexts:  
 autotext.set\_color('white')  
 autotext.set\_fontweight('bold')  
   
 def create\_hist\_plot(self, ax):  
 ax.hist(self.values, bins=10, color='skyblue', edgecolor='black', alpha=0.7)  
 ax.set\_title('Гистограмма распределения значений', fontsize=16, fontweight='bold')  
 ax.set\_xlabel(self.values\_name, fontsize=12)  
 ax.set\_ylabel("количество", fontsize=12)  
 ax.grid(True, alpha=0.3)  
  
def main():  
 root = tk.Tk()  
 app = MultiPlotApp(root, "window", "D:\\projects\\VisualStudioCode\\Laba\_2\_1\_5\_Graphics\\task1\\task1\_data.txt")  
 root.mainloop()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main() **Из файла ...\_5\_Graphics\task2\main.py.cpp**import sys  
import os  
import copy  
  
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_))))  
from laba\_logging import \*  
InitFile(\_\_file\_\_)  
  
  
import tkinter as tk  
from graphic import MultiPlotApp  
  
months = {"Янв": 31,   
 "Фев": 28 ,   
 "Мар": 31 ,  
 "Апр": 30 ,  
 "Май": 31 ,  
 "Июн": 30 ,  
 "Июл": 31 ,  
 "Авг": 31 ,  
 "Сен": 30 ,  
 "Окт": 31 ,  
 "Ноя": 30 ,  
 "Дек": 31 }  
  
months\_num = {"Янв": 1,   
 "Фев": 2 ,   
 "Мар": 3 ,  
 "Апр": 4 ,  
 "Май": 5 ,  
 "Июн": 6 ,  
 "Июл": 7 ,  
 "Авг": 8 ,  
 "Сен": 9 ,  
 "Окт": 10 ,  
 "Ноя": 11 ,  
 "Дек": 12 }  
  
  
  
def main():  
 root = tk.Tk()  
 app = MultiPlotApp(root, "window", "D:\\projects\\VisualStudioCode\\Laba\_2\_1\_5\_Graphics\\task2\\task2\_data.txt")  
 app.categories\_name = "дни"  
 app.values\_name = "коммиты"  
 days\_name = app.categories\_list  
 days\_data = app.values\_list  
  
 app.categories\_list\_without\_null = copy.deepcopy(app.categories\_list)  
 app.values\_list\_without\_null = copy.deepcopy(app.values\_list)  
  
 for year in range(0, 6):  
 for month in months:  
 for day in range(1, months[month]+1):  
 # print(f"year {year}, month {month}, day {day}")  
 title = f'{month} {year+2020}'  
 num\_cell = months\_num[month] + year\*12 - 1  
  
 if (num\_cell >= len(days\_name)) or (title not in days\_name[num\_cell][0]):  
 days\_name.insert(num\_cell, [])  
 days\_data.insert(num\_cell, [])  
 # print(f" поиск {day} {month} {year+2020} в {days\_name[num\_cell]}")  
 # print((len(days\_name[num\_cell]) == 0) , (len(days\_name[num\_cell]) < day))  
 if (len(days\_name[num\_cell]) == 0) or (len(days\_name[num\_cell]) < day) or (f'{day} {month} {year+2020}' != days\_name[num\_cell][day-1]):  
 days\_name[num\_cell].insert(day-1, f'{day} {title}')  
 days\_data[num\_cell].insert(day-1, 0)  
  
 print(app.categories\_list)  
 # days\_name = list(map(lambda x: list(x), app.categories\_list))  
 app.categories\_list = days\_name  
 app.values\_list = days\_data  
 root.mainloop()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()  
Снимок1.PNG:  
Снимок2.PNG:  
Снимок3.PNG:  
Снимок4.PNG:  
Снимок5.PNG:  
Снимок6.PNG:  
  
  
 Ссылка https://github.com/albel-t/Laba\_2\_1\_5\_Graphics.git на главный репозиторий  
 **Отчет сгенерирован report.py:**   
from docx import Document  
from docx.shared import Pt, RGBColor  
from docx.enum.text import WD\_UNDERLINE  
from docx.shared import Inches, Cm, Pt  
import glob  
import os  
  
def addCode(name, paragraph):  
 p = paragraph.add\_run(name)  
 p.font.name = 'Consolas'  
 p.font.size = Pt(12)  
 return p  
  
def addTask(task, paragraph):  
 p = paragraph.add\_run(task)  
 p.font.name = 'Times\_New\_Roman'  
 p.font.size = Pt(14)  
 return p  
   
def addCodeTitle(code, paragraph):  
 p = addTask(code, paragraph)  
 p.font.bold = True  
 return p  
  
def addTaskTitle(text, paragraph):  
 p = addTask(text, paragraph)  
 p.font.bold = True  
 return p  
  
def addImage(path, image, paragraph):  
 p = addTask(f'  
{image}:' , paragraph)  
 p.add\_picture(path + '\' + image, width=Inches(4.0))  
 return p  
  
#visual studio  
# path\_to\_project = input()  
   
LOG\_FILE = 'loggs.txt'   
  
path\_to\_this\_document = os.path.dirname(os.path.abspath(\_\_file\_\_))  
  
path\_to\_template\_document = "D:\projects\Templates\шаблон отчета Кафедра САПР.docx"  
path\_to\_README = "README.md"  
path\_to\_imgs = path\_to\_this\_document + '\img'  
name\_laba = "laba\_1"  
  
doc = Document(path\_to\_template\_document)  
  
style = doc.styles['Normal']  
style.font.name = 'Times\_New\_Roman'  
style.font.size = Pt(14)  
p = doc.add\_paragraph(" ")  
  
file\_log = open(LOG\_FILE, "r", encoding="utf8").readlines()  
files = []  
for string in file\_log:  
 string = string.split('|')  
 if "-init" in string[0]:  
 files.append(string[1].replace('  
', ''))  
 print("добавлени путь:" + string[1])  
  
  
file\_README = open(path\_to\_README, "r", encoding="utf8").read()  
for paragraph\_README in file\_README.split("##"):  
 times\_README = paragraph\_README.split(":")  
 print(f"  
{times\_README}  
")  
 if len(times\_README) > 1:  
 addTaskTitle(f"  
{times\_README[0]}  
", p)  
 print(f"  
{times\_README[1]}  
")  
 addTask(times\_README[1] + "  
", p)  
  
  
addTaskTitle("  
Листинг:  
", p)  
  
for file\_name in files:  
 print("записан путь:" + file\_name)  
 addCodeTitle("  
Из файла ..." + file\_name[-25:] + '.cpp  
', p)  
 addCode(open(file\_name, "r", encoding="utf8").read(), p)  
  
for item in os.listdir(path\_to\_imgs):  
 print("добавлени путь:" + item)  
 addImage(path\_to\_imgs, item, p)  
  
doc.save(path\_to\_this\_document + '/Oтчет\_'+name\_laba+'.docx')