ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ

**Руководство разработчика**

**для самостоятельной работы по курсу - «Проектный семинар “Python в науке о данных”»**

Работу выполнили студенты бригады №6 группы БИВ222:

* Игошина Дарья Дмитриевна, ddigoshina@ehu.hse.ru , +79069826495.
* Джибилова Дана Аслановна, dulusoi@yandex.ru, +79287107987.

Тема работы:

Приложение для работы с базами данных. Предметная область: столовая МИЭМ

Руководитель:

Поляков Константин Львович

Москва 2023

Оглавление

[Требование к характеристикам компьютера и ОС 3](#_Toc137936482)

[Версии языка и библиотек 4](#_Toc137936483)

[Описание архитектуры приложения 5](#_Toc137936484)

[Листинг программы 17](#_Toc137936485)

# Требование к характеристикам компьютера и ОС

МИНИМАЛЬНЫЕ СИСТЕМНЫЕ ТРЕБОВАНИЯ К КОМПЬЮТЕРУ:

ОС: Windows 10

Процессор: CPU AMD FX-4350 или аналогичный

Оперативная память: 2гб

Видеокарта: любая видеокарта с объемом видеопамяти не менее 500 мб. Например: GeForce 7300 GT, или Intel HD Graphics 620

РЕКОМЕНДОВАННЫЕ СИСТЕМНЫЕ ТРЕБОВАНИЯ К КОМПЬЮТЕРУ:

ОС: Windows 10/11

Процессор: CPU AMD Ryzen 5 или аналогичный

Оперативная память: 4гб

Видеокарта: любая видеокарта с объемом видеопамяти не менее 1гб. Например: AMD Radeon R5 Graphics, NVIDIA GeForce 930MX

# Версии языка и библиотек

Используемая версия Python 3.11 в составе дистрибутива Anaconda.

Используемые библиотеки и их версии:

tkinter==8.6

pandas==2.0.2

matplotlib==3.7.1

os== 3.9

sys==3.8

numpy==1.24.3

configparser == 5.3.0

# Описание архитектуры приложения

Листинг приложения разбит на шесть модулей. Главным является main.py, в котором создается главное окно приложения, а также осуществляется навигация между остальными окнами.

Модуль data\_manager.py отвечает за вывод списков в отдельном окне по нажатию кнопки, а также в нем прописаны функции редактирования справочников, добавления и удаления элементов.

Модуль graf\_information.py содержит в себе все виды графических отчетов: их параметры, а также их вывод на экран.

Модуль get\_fron\_ini.py получает данные из конфигурационного файла settings.ini, а также читает базы данных из файлов формата .csv с помощью библиотека pandas.

Модуль common\_functions хранит неперсонализированные функции, которые можно использовать в других проектах

Модуль text\_information.py содержит в себе все функции, необходимые для составления всех требуемых текстовых отчетов.

Файл settings.ini отвечает за хранения в себе данных настройки интерфейса и путей используемых файлов.

Интерфейс приложения.

Главное окно

рис. 1. Главное окно

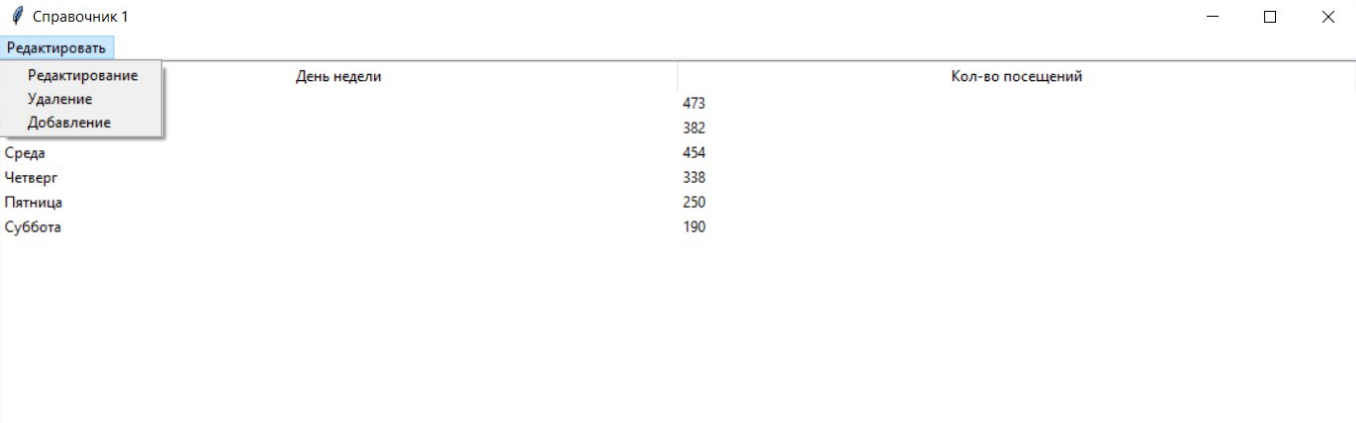
Изображение выглядит как снимок экрана, текст, программное обеспечение

Автоматически созданное описание

Для создания окна используется виджет Tk.

Кнопки сделаны с помощью виджеты Button.

рис. 2. Окно «Справочник 1»



Для создания окна используется виджет Tk.

Кнопки сделаны с помощью виджеты Button.

Меню сделано с помощью виджета Menu.

При нажатии на «Редактировать» открывается окно с пунктами меню.

При нажатии на «Редактирование» подпункта меню открывается окно для ввода данных, необходимых для редактирования.

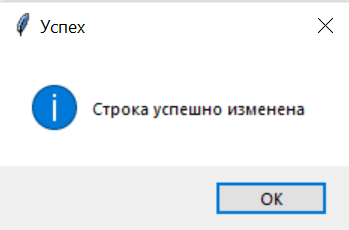
рис. 3. Редактирование

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

В случае успешного изменения данных появляется уведомление.

рис. 4. Всплывающее окно messagebox



При нажатии на «Удаление» открывается данное окно.

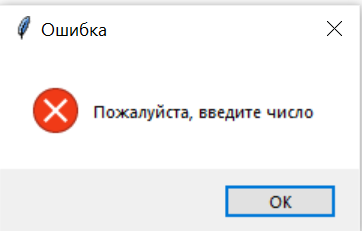
рис. 5. Окно удаления

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

В случае ввода символа приложение высвечивает уведомление.

рис. 6. Всплывающее окно messagebox



В случае ввода номера строки больше, чем есть в нашем файле, открывается окно.

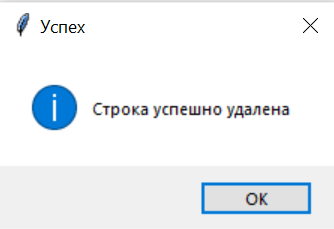
рис. 7. Всплывающее окно messagebox

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

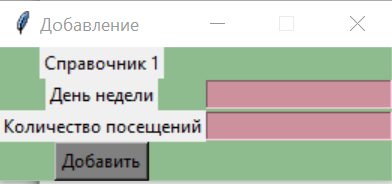
В случае же успешного удаления мы получаем следующий message box.

рис. 8. Всплывающее окно messagebox



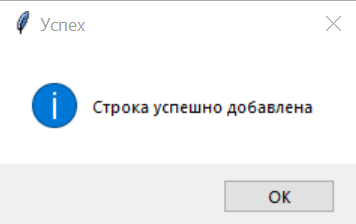
При нажатии на «Добавление» открывается окно для ввода данных

рис. 9. Окно добавления



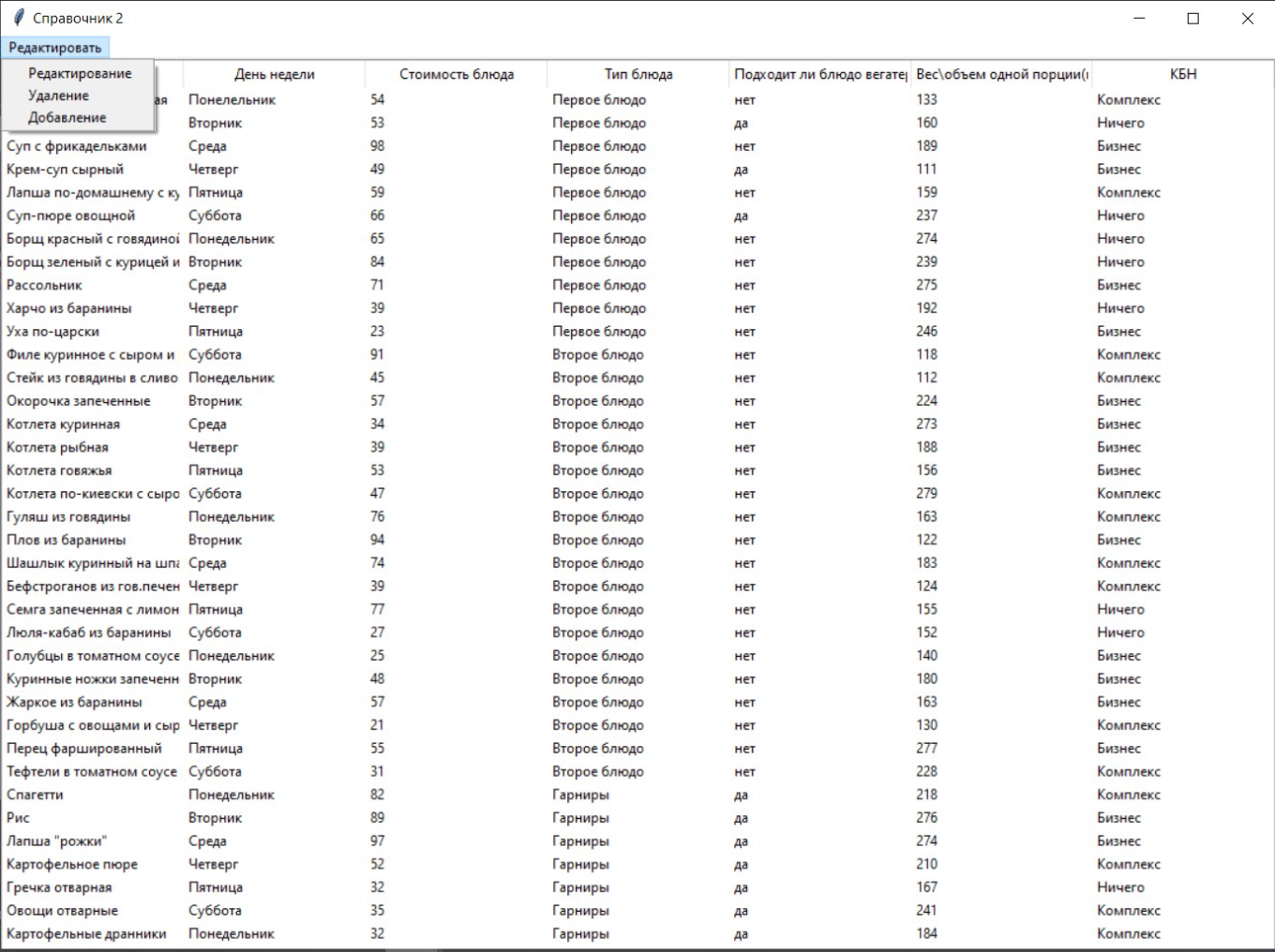
В случае успеха получаем:

рис. 10. Всплывающее окно messagebox



Окно «Справочник 2»

рис. 11. Окно «Справочник 2»



Для создания окна используется виджет Tk.

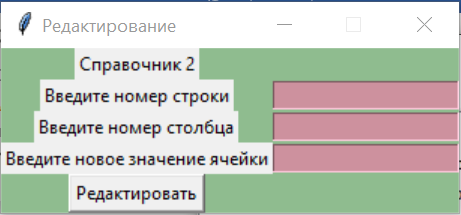
Кнопки сделаны с помощью виджеты Button.

Меню сделано с помощью виджета Menu.

При нажатии на «Редактировать» открывается окно с пунктами меню.

При нажатии на «Редактирование» подпункта меню открывается окно для ввода данных, необходимых для редактирования.

рис. 12. Окно редактирования

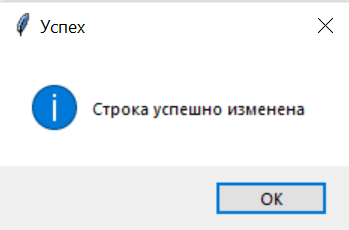


Поле ввода сделано с помощью виджета Entry.

Кнопка сделана с помощью виджета Button.

В случае успешного изменения данных появляется уведомление.

рис. 13. Всплывающее окно messagebox



Используется виджет messagebox.

При нажатии на «Удаление» открывается данное окно.

рис. 14. Окно удаления

Изображение выглядит как текст, Шрифт, линия, снимок экрана

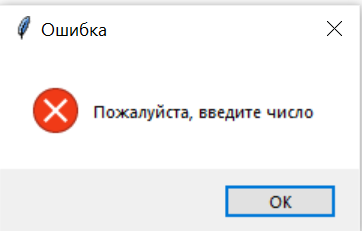
Автоматически созданное описание

Поле ввода сделано с помощью виджета Entry.

Кнопка сделана с помощью виджета Button.

В случае ввода символа приложение высвечивает уведомление.

рис. 15. Всплывающее окно messagebox



Используется виджет messagebox.

В случае ввода номера строки больше, чем есть в нашем файле, открывается окно.

рис. 16. Всплывающее окно messagebox

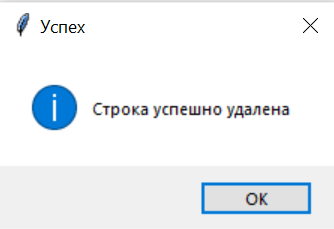
Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Используется виджет messagebox.

В случае же успешного удаления мы получаем следующий message box.

рис. 17. Всплывающее окно messagebox



Используется виджет messagebox.

При нажатии на «Добавление» открывается окно для ввода данных

рис. 18. Окно добавления

Изображение выглядит как текст, снимок экрана, Шрифт, линия

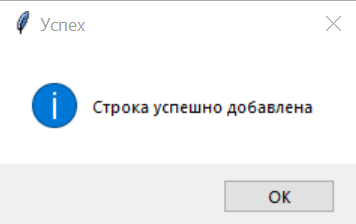
Автоматически созданное описание

Поле ввода сделано с помощью виджета Entry.

Кнопка сделана с помощью виджета Button.

В случае успеха получаем

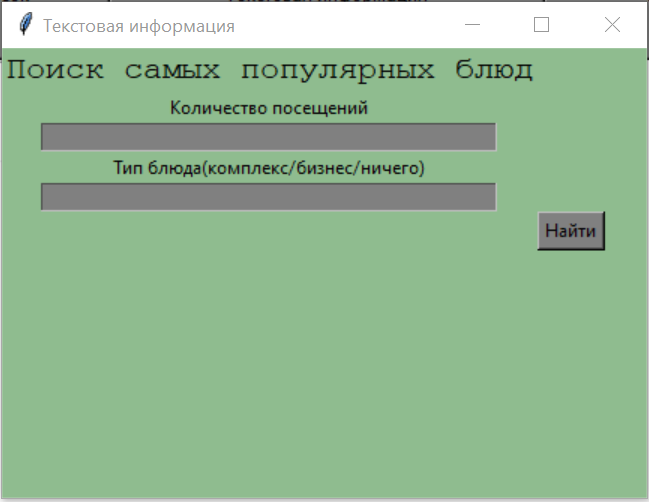
рис. 19. Всплывающее окно messagebox



Используется виджет messagebox.

Поиск самых популярных блюд

рис. 20. Окно получения данных для текстового отчета



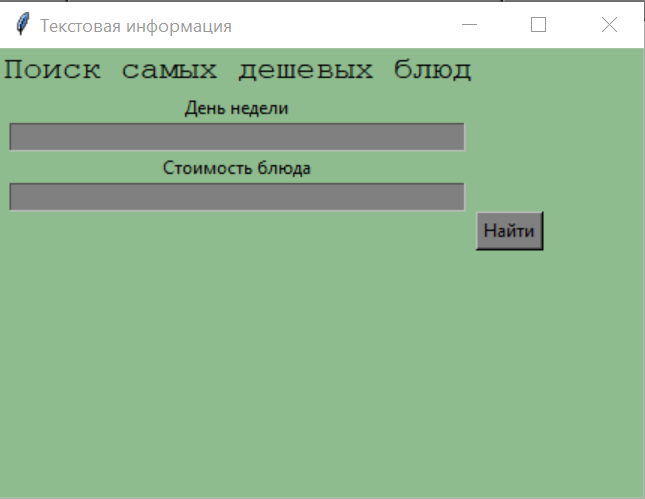
Для создания окна используется виджет Tk.

Кнопки сделаны с помощью виджеты Button.

Поле ввода сделано с помощью виджета Entry.

Поиск самых дешевых блюд

рис. 21. Окно получения данных для текстового отчета



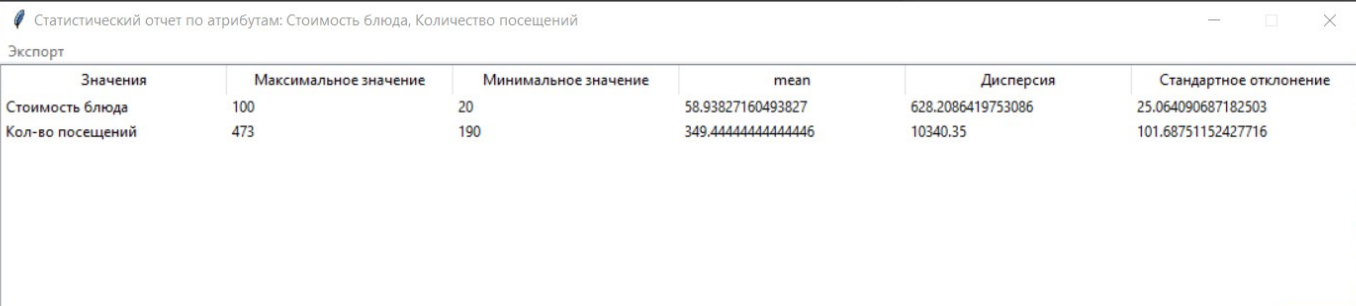
Для создания окна используется виджет Tk.

Кнопки сделаны с помощью виджеты Button.

Поле ввода сделано с помощью виджета Entry.

Статистический отчет

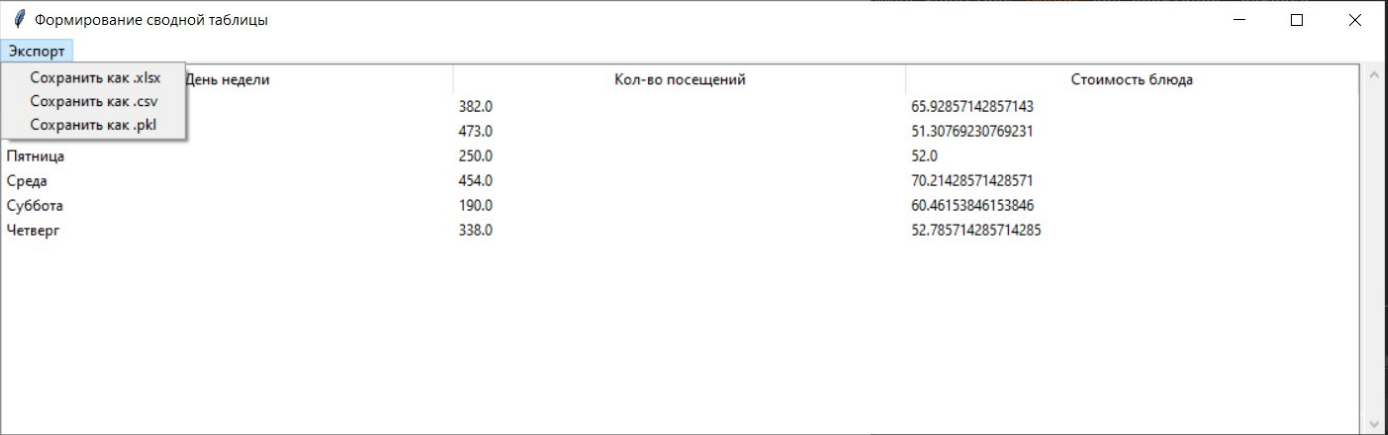
рис. 22. Вывод статистического отчета



Меню сделано с помощью виджета Menu.

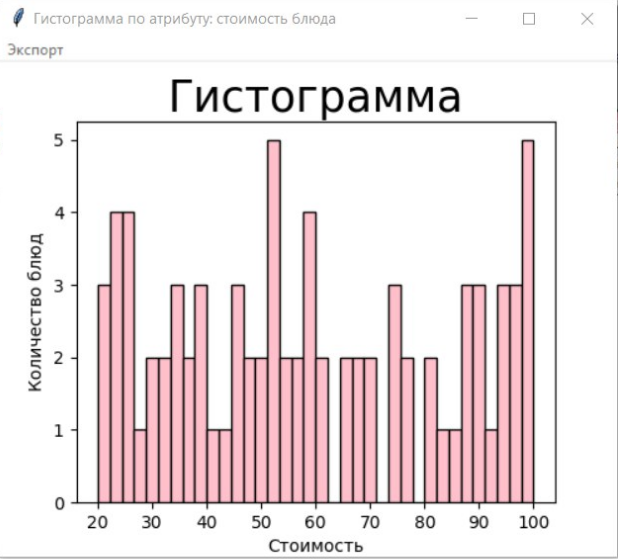
Сводная таблица

рис. 23. Вывод сводной таблицы



Меню сделано с помощью виджета Menu.

График 1. Гистограмма



Меню сделано с помощью виджета Menu.

График 2. Столбчатая диаграмма

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Меню сделано с помощью виджета Menu.

График 3. Диаграмма Бокса-Вискера



Меню сделано с помощью виджета Menu.

График 4. Диаграмма рассеивания

Изображение выглядит как текст, снимок экрана, число, График

Автоматически созданное описание

Меню сделано с помощью виджета Menu.

# Листинг программы

Листинг модуля main.py:

"""

Файл: main.py

Авторы: Игошина Дарья Дмиртриевна, Джибилова Дана Аслановна

Этот модуль реализует графический пользовательский интерфейс

главного окна приложения для работы с базами данных.

"""

import tkinter as tk

from tkinter import \*

import os

import sys

from data\_manager import DF1, DF2

from graf\_information import graf\_report\_1, graf\_report\_2, graf\_report\_3, graf\_report\_4

from text\_information import stat\_report, simp\_report\_1, simp\_report\_2, summary\_table

from get\_from\_ini import root\_size, light\_col, dark\_col, sun, moon

# Добавляем директорию с проектом в путь

os.chdir("C:\\work")

sys.path.append("C:\\work")

# Создаем главное окно

root = Tk()

root.title("Итоговый проект")

root.geometry(root\_size)

root["bg"] = light\_col

root.resizable(width=False, height=False)

# Инициализация изображений для переключения тем

light = PhotoImage(file=sun)

dark = PhotoImage(file=moon)

switch\_value = BooleanVar(value=True)

def switch\_theme():

"""

Данная функция реализует изменение графического интерфейса

(светлая и темная тема)

"""

if switch\_value.get():

switch.config(image=dark, bg=dark\_col,

activebackground=dark\_col)

# Изменение на темную тему

root.config(bg=dark\_col)

switch\_value.set(False)

else:

switch.config(image=light, bg=light\_col,

activebackground=light\_col)

# Изменение на светлую тему

root.config(bg=light\_col)

switch\_value.set(True)

# Создание кнопки для переключения между светлой и темной темой

switch = Button(root, image=light,

bd=0, bg=light\_col,

activebackground=light\_col,

command=switch\_theme)

switch.place(relx=0.9, rely=0.85)

def go\_box\_1(\_):

"""

Функция реализует работу listbox под кнопкой 'Список'

"""

lb\_course\_selection = Lb1.curselection()

print(lb\_course\_selection)

for lst in lb\_course\_selection:

if lst == 0:

DF1()

elif lst == 1:

DF2()

# Создание листбокса 'Список'

Lb1 = Listbox(selectbackground='pink3', height=4, width=20)

Lb1.insert(0, 'Cписок 1')

Lb1.insert(1, 'Список 2')

Lb1.bind('<Double-Button-1>', go\_box\_1)

Lb1.grid()

def lb1():

"""

Данная функция вызывает listbox по кнопке 'Список'

"""

Lb1.grid(column=0, row=1)

Lb1.grid()

def go\_box\_2(\_):

"""

Функция реализует работу listbox под кнопкой 'Текстовая информация'

"""

lb\_course\_selection = Lb2.curselection()

print(lb\_course\_selection)

for lst in lb\_course\_selection:

if lst == 0:

simp\_report\_1()

elif lst == 1:

simp\_report\_2()

elif lst == 2:

stat\_report()

elif lst == 3:

summary\_table()

# Создание листбокса 'Текстовая информация'

Lb2 = Listbox(selectbackground='pink3', height=4, width=45)

Lb2.insert(0, 'Поиск самых популярных блюд')

Lb2.insert(1, 'Поиск самых дешевых блюд')

Lb2.insert(2, 'Статистический отчет')

Lb2.insert(3, 'Сводная таблица')

Lb2.bind('<Double-Button-1>', go\_box\_2)

def lb2():

"""

Данная функция вызывает listbox по кнопке 'Текстовая информация'

"""

Lb2.grid(column=1, row=1)

Lb2.grid()

def go\_box\_3(\_):

"""

Функция реализует работу listbox под кнопкой 'Графическая информация'

"""

lb\_course\_selection = Lb3.curselection()

for lst in lb\_course\_selection:

if lst == 0:

graf\_report\_1()

elif lst == 1:

graf\_report\_2()

elif lst == 2:

graf\_report\_3()

elif lst == 3:

graf\_report\_4()

# Создание листбокса 'Графическая информация'

Lb3 = Listbox(selectbackground='pink3', height=4, width=45)

Lb3.insert(0, 'График 1. Гистограмма')

Lb3.insert(1, 'График 2. Столбчатая диаграмма')

Lb3.insert(2, 'График 3. Диаграмма Бокса-Вискера')

Lb3.insert(3, 'График 4. Диаграмма рассеивания')

def lb3():

"""

Данная функция вызывает listbox по кнопке 'Графическая информация'

"""

Lb3.grid(column=2, row=1)

Lb3.bind('<Double-Button-1>', go\_box\_3)

Lb3.grid()

# Создание кнопки 'Список'

b1 = tk.Button(root, text='Список', bg='grey', height=5,

width=20, command=lb1, font=("Arial Bold", 9))

def on\_enter\_3(\_):

"""

Данная функция изменяет цвет кпопки 'Список' при наведении курсора

"""

b1['background'] = 'pink3'

def on\_leave\_3(\_):

"""

Данная функция изменяет цвет кпопки 'Список' при выводе курсора

из поля кнопки

"""

b1['background'] = 'grey'

b1.bind("<Enter>", on\_enter\_3)

b1.bind("<Leave>", on\_leave\_3)

# Создание кнопки 'Текстовая информация'

b2 = tk.Button(root, text='Текстовая информация', bg='grey', height=5,

width=40, command=lb2, font=("Arial Bold", 9))

def on\_enter\_1(\_):

"""

Данная функция изменяет цвет кпопки 'Текстовая информация' при наведении курсора

"""

b2['background'] = 'pink3'

def on\_leave\_1(\_):

"""

Данная функция изменяет цвет кпопки 'Текстовая информация' при выводе курсора

из поля кнопки

"""

b2['background'] = 'grey'

b2.bind("<Enter>", on\_enter\_1)

b2.bind("<Leave>", on\_leave\_1)

# Создание кнопки 'Графическая информация'

b3 = tk.Button(root, text='Графическая информация', bg='grey',

height=5, width=40, command=lb3, font=("Arial Bold", 9))

def on\_enter\_2(\_):

"""

Данная функция изменяет цвет кпопки 'Графическая информация' при наведении курсора

"""

b3['background'] = 'pink3'

def on\_leave\_2(\_):

"""

Данная функция изменяет цвет кпопки 'Графическая информация' при выводе курсора

из поля кнопки

"""

b3['background'] = 'grey'

b3.bind("<Enter>", on\_enter\_2)

b3.bind("<Leave>", on\_leave\_2)

# Вызов и расположение кнопок

b1.grid(column=0, row=0)

b2.grid(column=1, row=0)

b3.grid(column=2, row=0)

root.mainloop()

**Листинг модуля get\_from\_ini.py:**

"""

Файл: get\_from\_ini.py

Авторы: Игошина Дарья Дмиртриевна, Джибилова Дана Аслановна

Этот модуль получает данные из конфигурационного файла settings.ini,

считывает базы данных из файлов формата .csv c помощью библиотеки pandas.

"""

import configparser

import pandas as pd

# создание объекта парсера

config = configparser.ConfigParser()

config.read("C:/work/scripts/settings.ini")

# чтение баз данных из файлов .csv

data\_frame\_1 = pd.read\_csv(config["datas"]["data\_1"], delimiter=",", encoding="utf-8",

dtype='O')

data\_frame\_1 = data\_frame\_1.astype({"Стоимость блюда": int})

data\_frame\_2 = pd.read\_csv(config["datas"]["data\_2"], delimiter=",", encoding="utf-8",

dtype='O')

data\_frame\_2 = data\_frame\_2.astype({"Кол-во посещений": int})

# деномарлизация базы даннных

denorm = pd.merge(data\_frame\_1, data\_frame\_2, on='День недели')

# получение настроек интерфейса

root\_size = config["settings"]["size"]

light\_col = config["settings"]["light\_col"]

dark\_col = config["settings"]["dark\_col"]

# получение пути к изображениям для настройки интерфейся

sun = config["images"]["sun"]

moon = config["images"]["moon"]

# получение путей файлов для экспорта

export\_simp\_1\_xlsx = config["export"]["export\_simp\_1\_xlsx"]

export\_simp\_1\_csv = config["export"]["export\_simp\_1\_csv"]

export\_simp\_1\_pkl = config["export"]["export\_simp\_1\_pkl"]

export\_simp\_2\_xlsx = config["export"]["export\_simp\_2\_xlsx"]

export\_simp\_2\_csv = config["export"]["export\_simp\_2\_csv"]

export\_simp\_2\_pkl = config["export"]["export\_simp\_2\_pkl"]

export\_stat\_xlsx = config["export"]["export\_stat\_xlsx"]

export\_stat\_csv = config["export"]["export\_stat\_csv"]

export\_stat\_pkl = config["export"]["export\_stat\_pkl"]

export\_sum\_xlsx = config["export"]["export\_sum\_xlsx"]

export\_sum\_csv = config["export"]["export\_sum\_csv"]

export\_sum\_pkl = config["export"]["export\_sum\_pkl"]

graf\_1\_png = config["graf"]["graf\_1\_png"]

graf\_2\_png = config["graf"]["graf\_2\_png"]

graf\_3\_png = config["graf"]["graf\_3\_png"]

graf\_4\_png = config["graf"]["graf\_4\_png"]

**Листинг модуля common\_functions.py:**

"""

Файл: common\_functions.py

Авторы: Игошина Дарья Дмиртриевна, Джибилова Дана Аслановна

Этот модуль содержит функции, которые могут быть использованы в других проектах

и не адаптированы под данное приложение.

"""

from tkinter import ttk

def cnv\_configure(cnv):

"""

Функция создает фигуру для вставки таблицы.

"""

cnv.configure(scrollregion=cnv.bbox("all"))

def refresh(window, file):

"""

Данная функция осуществляет вывод справочника/таблицы на экран,

обновление данных таблиц.

"""

l\_1 = list(file)

r\_set = file.to\_numpy().tolist()

tree\_v = ttk.Treeview(window, selectmode='browse', height=6,

show='headings', columns=l\_1)

tree\_v.place(x=0, y=0, width=1100, height=800)

for i in l\_1:

tree\_v.column(i, width=150)

tree\_v.heading(i, text=str(i))

for dif\_t in r\_set:

value = list(dif\_t)

tree\_v.insert("", 'end', iid=value[0], values=value)

**Листинг модуля text\_information.py:**

"""

Файл: text\_information.py

Авторы: Игошина Дарья Дмиртриевна, Джибилова Дана Аслановна

Данный файл содержит функции для для получения следующих текстовых отчетов:

поиск самых популярных блюд (простой текстовый отчет 1)

поиск самых дешевых блюд (простой текстовый отчет 2), статистический отчет,

сводная таблица

"""

import tkinter as tk

from tkinter import \*

from tkinter import ttk

from tkinter import messagebox

from common\_functions import cnv\_configure

from common\_functions import refresh

from get\_from\_ini import \*

from get\_from\_ini import denorm

def refresh1(window, d\_f):

"""

Данная функция выводит на экран реализованную сводную таблицу.

"""

l\_1\_dict = dict(zip(d\_f.index, d\_f.values))

l\_1\_list = list(l\_1\_dict.keys())

l\_1 = list(d\_f)

l\_1.insert(0, 'День недели')

r\_set = d\_f.to\_numpy().tolist() # create list of list using rows

need = 0

for i in r\_set:

i.insert(0, l\_1\_list[need])

need += 1

trv = ttk.Treeview(window, selectmode='browse', height=6,

show='headings', columns=l\_1)

trv.place(x=0, y=0, width=1100, height=800)

for i in l\_1:

trv.column(i, width=150)

trv.heading(i, text=str(i))

for dif\_t in r\_set:

value = list(dif\_t)

trv.insert("", 'end', iid=value[0], values=value)

def simp\_report\_1():

"""

Данная функция реализует простой текстовый отчет:

поиск самых популярных блюд.

"""

def export(out, k):

"""

Функция осуществяет экспорт отчета в выбранный файл (.xlsx/.csv/.pkl)

"""

if k == 0:

out.to\_excel(export\_simp\_1\_xlsx, index=True)

elif k == 1:

out.to\_csv(export\_simp\_1\_csv, index=True)

elif k == 2:

out.to\_pickle(export\_simp\_1\_pkl, index=True)

def win():

try:

visits = int(txt1.get())

except ValueError:

messagebox.showerror("Ошибка ввода", "Пожалуйста,"

" введите корректное"

" количество посещений (число).")

return

type\_dish = txt2.get()

if type\_dish not in ['Комплекс', 'Бизнес', 'Ничего']:

messagebox.showerror("Ошибка ввода", "Пожалуйста,"

" введите корректный тип"

" блюда (комплекс, бизнес, ничего).")

return

visits = int(visits)

type\_dish = txt2.get()

menu = tk.Menu(window)

window.config(menu=menu)

# Создание меню для экспорта

exp = tk.Menu(menu, tearoff=0)

exp.add\_command(label="Cохранить как .xlsx",

command=lambda: export(simp\_1, 0))

exp.add\_command(label="Cохранить как .csv",

command=lambda: export(simp\_1, 1))

exp.add\_command(label="Cохранить как .pkl",

command=lambda: export(simp\_1, 2))

menu.add\_cascade(label="Экспорт", menu=exp)

k = (denorm["Кол-во посещений"] >= visits) & (denorm["КБН"] == type\_dish)

simp\_1 = denorm.loc[k, ["Название блюда", "День недели"]]

dish\_name = simp\_1.get("Название блюда")

dish\_day = simp\_1.get("День недели")

frame = Frame(window)

frame.grid()

table = ttk.Treeview(frame)

table['columns'] = ('name', 'day')

table.column("#0", width=0, stretch=NO)

table.column("name", anchor=CENTER, width=100)

table.column("day", anchor=CENTER, width=80)

table.heading("#0", text="", anchor=CENTER)

table.heading("name", text="Название блюда", anchor=CENTER)

table.heading("day", text="День недели", anchor=CENTER)

table.pack()

a\_1 = 0

for row in dish\_name:

print(row)

a\_2 = 0

for i in dish\_day:

if a\_2 == a\_1:

print(i)

table.insert(parent='', index='end', text='', values=(row, i))

a\_2 += 1

a\_1 += 1

# создание нового окна с помощью tkinter

window = Tk()

window.title("Текстовая информация")

window.geometry("430x300")

window["bg"] = "dark sea green"

lb1 = Label(window, text="Поиск самых популярных блюд", bg='dark sea green')

lb1.configure(font=("Courier", 16))

lb1.grid(column=0, row=0)

lb2 = Label(window, text="Количество посещений", bg='dark sea green')

lb2.grid(column=0, row=1)

lb3 = Label(window, text="Тип блюда(Комплекс/Бизнес/Ничего)", bg='dark sea green')

lb3.grid(column=0, row=3)

txt1 = Entry(window, width=50, bg='grey')

txt1.grid(column=0, row=2)

txt2 = Entry(window, width=50, bg='grey')

txt2.grid(column=0, row=4)

bt1 = Button(window, text="Найти", bg='grey', command=win)

bt1.grid(column=1, row=5)

window.mainloop()

def simp\_report\_2():

"""

Данная функция реализует простой текстовый отчет:

поиск самых популярных блюд.

"""

def export(out, k):

"""

Функция осуществяет экспорт отчета в выбранный файл (.xlsx/.csv/.pkl)

"""

if k == 0:

out.to\_excel(export\_simp\_2\_xlsx, index=True)

elif k == 1:

out.to\_csv(export\_simp\_2\_csv, index=True)

elif k == 2:

out.to\_pickle(export\_simp\_2\_pkl, index=True)

def most\_cheap\_dish():

day = txt1.get()

if day not in ['Понедельник', 'Вторник', 'Среда',

'Четверг', 'Пятница', 'Суббота', 'Воскресенье']:

messagebox.showerror("Ошибка ввода", "Пожалуйста, введите корректный день недели.")

return

try:

price = int(txt2.get())

except ValueError:

messagebox.showerror("Ошибка ввода", "Пожалуйста,"

" введите корректную стоимость блюда (число).")

return

# Создание меню для экспорта

menu = tk.Menu(window)

window.config(menu=menu)

exp = tk.Menu(menu, tearoff=0)

exp.add\_command(label="Cохранить как .xlsx",

command=lambda: export(simp\_2, 0))

exp.add\_command(label="Cохранить как .csv",

command=lambda: export(simp\_2, 1))

exp.add\_command(label="Cохранить как .pkl",

command=lambda: export(simp\_2, 2))

menu.add\_cascade(label="Экспорт", menu=exp)

k = (denorm["День недели"] == day) & (denorm["Стоимость блюда"] <= price)

simp\_2 = denorm.loc[k, ["Название блюда", "Тип блюда"]]

dish\_name = simp\_2.get("Название блюда")

type\_dish = simp\_2.get("Тип блюда")

dish\_type = simp\_2.get("Тип блюда")

print(dish\_type)

print(dish\_name)

frame = Frame(window)

frame.grid()

table = ttk.Treeview(frame)

table['columns'] = ('name', 'type')

table.column("#0", width=0, stretch=NO)

table.column("name", anchor=CENTER, width=100)

table.column("type", anchor=CENTER, width=80)

table.heading("#0", text="", anchor=CENTER)

table.heading("name", text="Название блюда", anchor=CENTER)

table.heading("type", text="Тип блюда", anchor=CENTER)

table.pack()

a\_1 = 0

for row in dish\_name:

print(row)

a\_2 = 0

for i in type\_dish:

if a\_2 == a\_1:

print(i)

table.insert(parent='', index='end', text='', values=(row, i))

a\_2 += 1

a\_1 += 1

# создание нового окна с помощью tkinter

window = Tk()

window.title("Текстовая информация")

window.geometry("430x300")

window["bg"] = "dark sea green"

lb1 = Label(window, text="Поиск самых дешевых блюд", bg='dark sea green')

lb1.configure(font=("Courier", 16))

lb1.grid(column=0, row=0)

lb2 = Label(window, text="День недели", bg='dark sea green')

lb2.grid(column=0, row=1)

lb3 = Label(window, text="Стоимость блюда", bg='dark sea green')

lb3.grid(column=0, row=3)

txt1 = Entry(window, width=50, bg='grey')

txt1.grid(column=0, row=2)

txt2 = Entry(window, width=50, bg='grey')

txt2.grid(column=0, row=4)

bt1 = Button(window, text="Найти", bg='grey', command=most\_cheap\_dish)

bt1.grid(column=1, row=5)

window.mainloop()

def stat\_report():

"""

Данная функция реализует статистический текстовый отчет по атрибутам:

'Стоимость блюда', 'Количество посещений'.

"""

def export(out, k):

"""

Функция осуществяет экспорт отчета в выбранный файл (.xlsx/.csv/.pkl)

"""

if k == 0:

out.to\_excel(export\_stat\_xlsx, index=True)

elif k == 1:

out.to\_csv(export\_stat\_csv, index=True)

elif k == 2:

out.to\_pickle(export\_stat\_pkl, index=True)

# создание нового окна с помощью tkinter

window = tk.Toplevel()

window.geometry("1100x200")

# Создание меню для экспорта

menu = tk.Menu(window)

window.config(menu=menu)

exp = tk.Menu(menu, tearoff=0)

exp.add\_command(label="Cохранить как .xlsx",

command=lambda: export(table\_stat, 0))

exp.add\_command(label="Cохранить как .csv",

command=lambda: export(table\_stat, 1))

exp.add\_command(label="Cохранить как .pkl",

command=lambda: export(table\_stat, 2))

menu.add\_cascade(label="Экспорт", menu=exp)

window.title("Формирование сводной таблицы")

window.resizable(False, False)

window.title("Статистический отчет по атрибутам: Стоимость блюда, Количество посещений")

canvas = tk.Canvas(window, borderwidth=0)

frame = tk.Frame(canvas)

scroll\_ver = tk.Scrollbar(window, orient="vertical", command=canvas.yview)

scroll\_hor = tk.Scrollbar(window, orient="horizontal", command=canvas.xview)

canvas.configure(yscrollcommand=scroll\_ver.set)

scroll\_ver.grid()

canvas.configure(xscrollcommand=scroll\_hor.set)

canvas.grid()

canvas.create\_window((1, 1), window=frame, anchor="nw")

frame.bind("<Configure>", lambda event, cnv=canvas: cnv\_configure(canvas))

table\_stat = pd.DataFrame()

table\_stat["Значения"] = ["Стоимость блюда", "Кол-во посещений"]

table\_stat["Максимальное значение"] = [denorm["Стоимость блюда"].max(

), denorm["Кол-во посещений"].max()]

table\_stat["Минимальное значение"] = [denorm["Стоимость блюда"].min(

), denorm["Кол-во посещений"].min()]

table\_stat["mean"] = [denorm["Стоимость блюда"].mean(

), denorm["Кол-во посещений"].mean()]

table\_stat["Дисперсия"] = [

denorm["Стоимость блюда"].var(), denorm["Кол-во посещений"].var()]

table\_stat["Стандартное отклонение"] = [

denorm["Стоимость блюда"].std(), denorm["Кол-во посещений"].std()]

refresh(window, table\_stat)

def summary\_table():

"""

Данная функция реализует сводную таблицу по атрибутам:

'Стоимость блюда', 'Количество посещений',

по индексу 'День недели'.

"""

# Получение сводной таблицы

sum\_table = pd.pivot\_table(denorm, index="День недели",

values=["Стоимость блюда", "Кол-во посещений"])

def export(out, k):

"""

Функция осуществяет экспорт отчета в выбранный файл (.xlsx/.csv/.pkl)

"""

if k == 0:

out.to\_excel(export\_sum\_xlsx, index=True)

elif k == 1:

out.to\_csv(export\_sum\_csv, index=True)

elif k == 2:

out.to\_pickle(export\_sum\_pkl, index=True)

# создание нового окна с помощью tkinter

window = tk.Toplevel()

window.geometry("1120x300")

menu = tk.Menu(window)

window.config(menu=menu)

# Создание меню для экспорта

exp = tk.Menu(menu, tearoff=0)

exp.add\_command(label="Cохранить как .xlsx",

command=lambda: export(sum\_table, 0))

exp.add\_command(label="Cохранить как .csv",

command=lambda: export(sum\_table, 1))

exp.add\_command(label="Cохранить как .pkl",

command=lambda: export(sum\_table, 2))

menu.add\_cascade(label="Экспорт", menu=exp)

window.title("Формирование сводной таблицы")

canvas = tk.Canvas(window, borderwidth=0)

frame = tk.Frame(canvas)

scroll\_ver = tk.Scrollbar(window, orient="vertical", command=canvas.yview)

scroll\_hor = tk.Scrollbar(window, orient="horizontal", command=canvas.xview)

canvas.configure(yscrollcommand=scroll\_ver.set)

scroll\_ver.pack(side="right", fill='y')

canvas.configure(xscrollcommand=scroll\_hor.set)

scroll\_hor.pack(side="bottom", fill='x')

canvas.pack(side="left", fill="both", expand=True)

canvas.create\_window((1, 1), window=frame, anchor="nw")

frame.bind("<Configure>", lambda event, cnv=canvas: cnv\_configure(canvas))

refresh1(window, sum\_table)

**Листинг модуля graf\_information.py:**

"""

Файл: graf\_information.py

Авторы: Игошина Дарья Дмиртриевна, Джибилова Дана Аслановна

Этот модуль реализует cоздание графических отчетов.

"""

import tkinter as tk

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

from matplotlib.figure import Figure

from get\_from\_ini import denorm, graf\_1\_png, graf\_2\_png, graf\_3\_png, graf\_4\_png

def graf\_report\_1():

"""

Создание гистограммы по атрибуту 'Стоимость блюда'.

"""

def export():

"""

Экспорт гистограммы в формат .png.

"""

fig.savefig(graf\_1\_png)

# создание нового окна с помощью tkinter

window = tk.Tk()

window.title('Гистограмма по атрибуту: стоимость блюда')

# создание меню для экспорта

menu = tk.Menu(window)

window.config(menu=menu)

menu.add\_cascade(label="Экспорт", command=export)

# создание фигуры для графика

fig = Figure(figsize=(5, 4), dpi=100)

# добавление подграфика в фигуру

plot = fig.add\_subplot(1, 1, 1)

# отрисовка столбчатого графика

plot.hist(denorm['Стоимость блюда'], color='pink', edgecolor='black',

bins=int(180 / 5))

plot.set\_title("Гистограмма", fontsize=25, color="black")

plot.set\_ylabel('Количество блюд')

plot.set\_xlabel('Стоимость')

canvas = FigureCanvasTkAgg(fig, master=window)

canvas.draw()

canvas.get\_tk\_widget().pack()

def graf\_report\_2():

"""

Создание столбчатой диаграммы по атрибутам 'Стоимость блюда', 'День недели'.

"""

def export():

"""

Экспорт гистограммы в формат .png.

"""

fig.savefig(graf\_2\_png)

# создание нового окна с помощью tkinter

window = tk.Tk()

window.title('Столбчатая диаграмма по атрибутам: стоимость блюда, день недели')

menu = tk.Menu(window)

window.config(menu=menu)

menu.add\_cascade(label="Экспорт", command=export)

# создание фигуры для графика

fig = Figure(figsize=(5, 4), dpi=100)

# добавление подграфика в фигуру

plot = fig.add\_subplot(1, 1, 1)

# отрисовка столбчатого графика

plot.bar(denorm['День недели'], denorm['Стоимость блюда'], color='pink')

plot.set\_xlabel("День недели")

plot.set\_ylabel("Стоимость блюда")

plot.set\_title("Столбчатая диаграмма")

canvas = FigureCanvasTkAgg(fig, master=window)

canvas.draw()

canvas.get\_tk\_widget().pack()

def graf\_report\_3():

"""

Создание диаграммы Бокса-Вискера по атрибуту: 'Стоимость блюда'.

"""

def export():

"""

Экспорт гистограммы в формат .png.

"""

fig.savefig(graf\_3\_png)

# создание нового окна с помощью tkinter

window = tk.Tk()

window.title('Диаграмма Бокса-Вискера по атрибуту: cтоимость блюда')

menu = tk.Menu(window)

window.config(menu=menu)

menu.add\_cascade(label="Экспорт", command=export)

# создание фигуры для графика

fig = Figure(figsize=(5, 4), dpi=100)

# добавление подграфика в фигуру

plot = fig.add\_subplot(1, 1, 1)

# отрисовка столбчатого графика

data = denorm['Стоимость блюда']

plot.set\_title("Диаграмма Бокса-Вискера: cтоимость блюда")

plot.boxplot(data)

canvas = FigureCanvasTkAgg(fig, master=window)

canvas.draw()

canvas.get\_tk\_widget().pack()

def graf\_report\_4():

"""

Создание диаграммы рассеивания по атрибутам 'Стоимость блюда', 'Количество посещений'.

"""

def export():

"""

Экспорт гистограммы в формат .png.

"""

fig.savefig(graf\_4\_png) # создание нового окна с помощью tkinter

window = tk.Tk()

window.title('Диаграмма рассеивания по атрибутам: кол-во посещений, стоимость блюда')

menu = tk.Menu(window)

window.config(menu=menu)

menu.add\_cascade(label="Экспорт", command=export)

# создание фигуры для графика

fig = Figure(figsize=(5, 4), dpi=100)

# добавление подграфика в фигуру

plot = fig.add\_subplot(1, 1, 1)

# отрисовка столбчатого графика

plot.scatter(denorm['Кол-во посещений'], denorm['Стоимость блюда'], color='pink')

plot.set\_xlabel("Количество посещений")

plot.set\_ylabel("Стоимость блюда")

plot.set\_title("Диаграмма рассеивания")

canvas = FigureCanvasTkAgg(fig, master=window)

canvas.draw()

canvas.get\_tk\_widget().pack()

Листинг модуля data\_manager.py:

"""

**Файл: data\_manager.py**

Авторы: Игошина Дарья Дмиртриевна, Джибилова Дана Аслановна

Этот модуль реализует вывод справочников на экран, возможность редактирования,

добавления, удаления

"""

import tkinter as tk

from tkinter import \*

from tkinter import ttk

from tkinter import messagebox

import pandas as pd

from get\_from\_ini import data\_frame\_1, data\_frame\_2

from common\_functions import cnv\_configure

class DF1(tk.Tk):

"""

В данном классе реализован вывод справочника 1 в отдельное окно,

а также он содержит функции ручной модификации файла, ручного добавления и удаления

"""

def \_\_init\_\_(self):

"""Создание отдельного окна с выведенным справочником 1,

а также прописано меню этого окна"""

super().\_\_init\_\_()

self.resizable(width=False, height=False)

self.configure(bg="dark sea green")

self.geometry("1100x800")

mainmenu = Menu(self)

self.config(menu=mainmenu)

self.resizable(True, True)

self.title("Cправочник 1")

canvas = tk.Canvas(self, borderwidth=0)

frame = tk.Frame(canvas)

scroll\_ver = tk.Scrollbar(self, orient="vertical", command=canvas.yview)

scroll\_hor = tk.Scrollbar(self, orient="horizontal", command=canvas.xview)

canvas.configure(yscrollcommand=scroll\_ver.set)

scroll\_ver.grid()

canvas.configure(xscrollcommand=scroll\_hor.set)

scroll\_hor.grid()

canvas.grid()

canvas.create\_window((1, 1), window=frame, anchor="nw")

frame.bind("<Configure>", lambda event, cnv=canvas: cnv\_configure)

actionmenu = Menu(mainmenu, tearoff=0)

def add\_win():

"""Создание окна 'Добавление', содержащее два поля ввода данных, а также кнопку"""

window = Tk()

window.resizable(width=False, height=False)

window.title("Добавление")

window["bg"] = "dark sea green"

text = tk.Label(master=window, text="Справочник 1")

text.grid(column=0, row=0)

def add\_row():

"""Функция добавления полученных в полях ввода данных."""

day\_text = day.get()

quantity\_text = quantity.get()

data\_frame\_2 = pd.read\_csv("./data/Posecheniya.csv",

delimiter=",", encoding="utf-8", dtype='O')

data\_frame\_2.loc[len(data\_frame\_2) + 1] = [day\_text, quantity\_text]

data\_frame\_2.sort\_index()

data\_frame\_2.to\_csv("./data/Posecheniya.csv")

exclude\_column = ""

data\_frame\_2 = data\_frame\_2.loc[:, data\_frame\_2.columns != exclude\_column]

data\_frame\_2.to\_csv("./data/Posecheniya.csv", index=False, sep=',')

messagebox.showinfo("Успех", "Строка успешно добавлена")

refresh(self, data\_frame\_2)

tx1 = tk.Label(master=window, text="День недели")

tx1.grid(row=1)

tx2 = tk.Label(master=window, text="Количество посещений")

tx2.grid(row=2)

day = tk.Entry(master=window, bg='pink3')

day.grid(row=1, column=1, sticky=tk.W)

quantity = tk.Entry(master=window, bg='pink3')

quantity.grid(row=2, column=1, sticky=tk.W)

btn\_submit = tk.Button(master=window, text="Добавить", bg='grey', command=add\_row)

btn\_submit.grid(column=0, row=3)

def edit():

"""Создание окна 'Редактирование' с тремя полями ввода и одной кнопкой"""

window = Tk()

window.resizable(width=False, height=False)

window.title("Редактирование")

window["bg"] = "dark sea green"

text = tk.Label(master=window, text="Справочник 1")

text.grid(column=0, row=0)

text\_1 = tk.Label(master=window, text="Введите номер строки")

text\_1.grid(row=1)

text\_2 = tk.Label(master=window, text="Введите номер столбца")

text\_2.grid(row=2)

text\_3 = tk.Label(master=window, text="Введите новое значение ячейки")

text\_3.grid(row=3)

row\_num = tk.Entry(master=window, bg='pink3')

row\_num.grid(row=1, column=1, sticky=tk.W)

column\_num = tk.Entry(master=window, bg='pink3')

column\_num.grid(row=2, column=1, sticky=tk.W)

new\_value = tk.Entry(master=window, bg='pink3')

new\_value.grid(row=3, column=1, sticky=tk.W)

def cell\_change():

"""Функция добавления полученных в полях ввода данных в заданную ячейку"""

row\_num\_text = row\_num.get()

row\_num\_text = int(row\_num\_text)

column\_num\_text = column\_num.get()

column\_num\_text = int(column\_num\_text)

new\_value\_text = new\_value.get()

data\_frame\_2 = pd.read\_csv("./data/Posecheniya.csv", delimiter=",",

encoding="utf-8", dtype='O')

data\_frame\_2.iloc[row\_num\_text - 1, column\_num\_text - 1] = new\_value\_text

data\_frame\_2.to\_csv("./data/Posecheniya.csv", index=False, sep=',')

messagebox.showinfo("Успех", "Строка успешно изменена")

refresh(self, data\_frame\_2)

button = tk.Button(window, text="Редактировать", command=cell\_change)

button.grid(column=0, row=4)

def delete\_row(entry, window):

"""Функция проверяет введенное число на соответствие необходимым

параметрам, а также удаляет нужную строку.

На вход принимает переменные:

window - окно с полем введения номера удаляемой строки

enrty - поле ввода данных

"""

data\_frame\_2 = pd.read\_csv("./data/Posecheniya.csv", delimiter=",", encoding="utf-8",

dtype='O')

# получаем введенное значение из текстового поля

row\_num = entry.get()

# проверяем, что введено число

if row\_num.isdigit() == False:

messagebox.showerror("Ошибка", "Пожалуйста, введите число")

return

row\_num = int(row\_num)

# Проверяем, что введенное число соответствует диапазону индексов строк DataFrame

if row\_num < 0 or row\_num >= len(data\_frame\_2):

messagebox.showerror("Ошибка", "Недопустимый номер строки")

return

else:

# Удаляем строку из DataFrame

data\_frame\_2 = data\_frame\_2[data\_frame\_2.index != row\_num]

data\_frame\_2.to\_csv("./data/Posecheniya.csv", index=False)

messagebox.showinfo("Успех", "Строка успешно удалена")

window.destroy()

refresh(self, data\_frame\_2)

def delete\_window():

"""

Создание окна 'Удаление строки'.

Содержит одно поле ввода, а также одну кнопку

"""

# Создаем графический интерфейс

window = tk.Toplevel()

window.title("Удаление строки")

# Создаем метку и текстовое поле

label = tk.Label(window, text="Введите номер строки:")

label.grid(column=0, row=0)

entry = tk.Entry(window)

entry.grid(column=0, row=1)

def delete\_row\_without\_traceback():

"""Функция вызывает delete\_row без получения ошибки Tracebask"""

delete\_row(entry, window)

# Создаем кнопку для удаления строки

button = tk.Button(window, text="Удалить", command=delete\_row\_without\_traceback)

button.grid(column=0, row=2)

actionmenu.add\_command(label="Редактирование", command=edit)

actionmenu.add\_command(label="Удаление", command=delete\_window)

actionmenu.add\_command(label="Добавление", command=add\_win)

mainmenu.add\_cascade(label="Редактировать", menu=actionmenu)

def refresh(window, data):

"""

Функция обновляет данные в окне с выведенным справочнииком 1

На вход получает переменные:

window - окно с выведенным сравочником

data\_frame\_2 - DataFrame, содержащий считанный файл формата .csv"""

list1 = list(data)

r\_set = data.to\_numpy().tolist() # create list of list using rows

trv = ttk.Treeview(window, selectmode='browse', height=6,

show='headings', columns=list1)

trv.place(x=0, y=0, width=1100, height=800)

for i in list1:

trv.column(i, width=150)

trv.heading(i, text=str(i))

for dif\_t in r\_set:

value = list(dif\_t)

trv.insert("", 'end', iid=value[0], values=value)

refresh(self, data\_frame\_2)

class DF2(tk.Tk):

"""

Создание отдельного окна с выведенным справочником 2,

а также тут прописано меню этого окна

"""

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.resizable(width=False, height=False)

self.configure(bg="dark sea green")

self.geometry("1100x800")

mainmenu = Menu(self)

self.config(menu=mainmenu)

self.resizable(True, True)

self.title("Cправочник 2")

canvas = tk.Canvas(self, borderwidth=0)

frame = tk.Frame(canvas)

scroll\_ver = tk.Scrollbar(self, orient="vertical", command=canvas.yview)

scroll\_hor = tk.Scrollbar(self, orient="horizontal", command=canvas.xview)

canvas.configure(yscrollcommand=scroll\_ver.set)

scroll\_ver.grid()

canvas.configure(xscrollcommand=scroll\_hor.set)

scroll\_hor.grid()

canvas.grid()

canvas.create\_window((1, 1), window=frame, anchor="nw")

frame.bind("<Configure>", lambda event, cnv=canvas: cnv\_configure)

actionmenu = Menu(mainmenu, tearoff=0)

def add\_win():

"""Создание окна 'Добавление', содержащее семь полей ввода данных, а также кнопку"""

window = Tk()

window.resizable(width=False, height=False)

window.title("Добавление")

window["bg"] = "dark sea green"

label = tk.Label(master=window, text="Справочник 1")

label.grid(column=0, row=0)

def add\_row():

"""Функция добавления полученных в полях ввода данных в виде новой строки"""

name\_text = name.get()

day\_text = day.get()

cost\_text = cost.get()

dish\_type\_text = dish\_type.get()

vegan\_text = vegan.get()

heft\_text = heft.get()

kbn\_text = kbn.get()

data\_frame\_1 = pd.read\_csv("./data/StolovoyaMIEM.csv",

delimiter=",", encoding="utf-8", dtype='O')

data\_frame\_1.loc[len(data\_frame\_1)] = [name\_text, day\_text, cost\_text,

dish\_type\_text, vegan\_text, heft\_text, kbn\_text]

data\_frame\_1.sort\_index()

data\_frame\_1.to\_csv("./data/StolovoyaMIEM.csv")

exclude\_column = ""

data\_frame\_1 = data\_frame\_1.loc[:, data\_frame\_1.columns != exclude\_column]

data\_frame\_1.to\_csv("./data/StolovoyaMIEM.csv", index=False, sep=',')

messagebox.showinfo("Успех", "Строка успешно добавлена")

refresh(self, data\_frame\_1)

tx1 = tk.Label(master=window, text="Название блюда")

tx1.grid(row=1)

tx2 = tk.Label(master=window, text="День недели")

tx2.grid(row=2)

tx3 = tk.Label(master=window, text="Стоимость блюда")

tx3.grid(row=3)

tx4 = tk.Label(master=window, text="Тип блюда")

tx4.grid(row=4)

tx5 = tk.Label(master=window, text="Подходит ли блюдо вегатерианцам")

tx5.grid(row=5)

tx6 = tk.Label(master=window, text="Вес/объем одной порции(г/мл)")

tx6.grid(row=6)

tx7 = tk.Label(master=window, text="Комплекс/бизнес/ничего")

tx7.grid(row=7)

name = tk.Entry(master=window, bg='pink3')

name.grid(row=1, column=1, sticky=tk.W)

day = tk.Entry(master=window, bg='pink3')

day.grid(row=2, column=1, sticky=tk.W)

cost = tk.Entry(master=window, bg='pink3')

cost.grid(row=3, column=1, sticky=tk.W)

dish\_type = tk.Entry(master=window, bg='pink3')

dish\_type.grid(row=4, column=1, sticky=tk.W)

vegan = tk.Entry(master=window, bg='pink3')

vegan.grid(row=5, column=1, sticky=tk.W)

heft = tk.Entry(master=window, bg='pink3')

heft.grid(row=6, column=1, sticky=tk.W)

kbn = tk.Entry(master=window, bg='pink3')

kbn.grid(row=7, column=1, sticky=tk.W)

btn\_submit = tk.Button(master=window, text="Добавить", bg='grey', command=add\_row)

btn\_submit.grid(column=0, row=8)

def edit():

"""Создание окна 'Редактирование' с тремя полями ввода и одной кнопкой"""

window = Tk()

window.resizable(width=False, height=False)

window.title("Редактирование")

window["bg"] = "dark sea green"

tx = tk.Label(master=window, text="Справочник 2")

tx.grid(column=0, row=0)

tx1 = tk.Label(master=window, text="Введите номер строки")

tx1.grid(row=1)

tx2 = tk.Label(master=window, text="Введите номер столбца")

tx2.grid(row=2)

tx3 = tk.Label(master=window, text="Введите новое значение ячейки")

tx3.grid(row=3)

row\_num = tk.Entry(master=window, bg='pink3')

row\_num.grid(row=1, column=1, sticky=tk.W)

column\_num = tk.Entry(master=window, bg='pink3')

column\_num.grid(row=2, column=1, sticky=tk.W)

new\_value = tk.Entry(master=window, bg='pink3')

new\_value.grid(row=3, column=1, sticky=tk.W)

def cell\_change():

"""Функция добавления полученных в полях ввода данных в заданную ячейку"""

row\_num\_text = row\_num.get()

row\_num\_text = int(row\_num\_text)

column\_num\_text = column\_num.get()

column\_num\_text = int(column\_num\_text)

new\_value\_text = new\_value.get()

data\_frame\_2 = pd.read\_csv("./data/StolovoyaMIEM.csv", delimiter=",",

encoding="utf-8", dtype='O')

data\_frame\_2.iloc[row\_num\_text - 1, column\_num\_text - 1] = new\_value\_text

data\_frame\_2.to\_csv("./data/StolovoyaMIEM.csv", index=False, sep=',')

messagebox.showinfo("Успех", "Строка успешно изменена")

refresh(self, data\_frame\_2)

button = tk.Button(window, text="Редактировать", command=cell\_change)

button.grid(column=0, row=4)

def delete\_row(entry, window):

"""

Функция проверяет введенное число на соответствие

необходимым параметрам, а также удаляет нужную строку.

На вход принимает переменные:

window - окно с полем введения номера удаляемой строки

enrty - поле ввода данных

"""

data\_frame\_1 = pd.read\_csv("./data/StolovoyaMIEM.csv", delimiter=",", encoding="utf-8",

dtype='O')

# Получаем введенное значение из текстового поля

row\_num = entry.get()

# Проверяем, что введено число'

if not row\_num.isdigit():

messagebox.showerror("Ошибка", "Пожалуйста, введите число")

return

row\_num = int(row\_num)

# Проверяем, что введенное число соответствует диапазону индексов строк DataFrame

if row\_num < 0 or row\_num >= len(data\_frame\_1):

messagebox.showerror("Ошибка", "Недопустимый номер строки")

return

else:

# Удаляем строку из DataFrame

print(data\_frame\_1)

data\_frame\_1 = data\_frame\_1[data\_frame\_1.index != row\_num]

print(data\_frame\_1)

data\_frame\_1.to\_csv("./data/StolovoyaMIEM.csv", index=False)

messagebox.showinfo("Успех", "Строка успешно удалена")

window.destroy()

refresh(self, data\_frame\_1)

def delete\_window():

"""

Создание окна 'Удаление строки'.

Содержит одно поле ввода, а также одну кнопку

"""

# Создаем графический интерфейс

window = tk.Toplevel()

window.title("Удаление строки")

# Создаем метку и текстовое поле

label = tk.Label(window, text="Введите номер строки:")

label.grid(column=0, row=0)

entry = tk.Entry(window)

entry.grid(column=0, row=1)

def delete\_row\_without\_traceback():

"""

Функция вызывает delete\_row без

получения ошибки Tracebask

"""

delete\_row(entry, window)

# Создаем кнопку для удаления строки

button = tk.Button(window, text="Удалить", command=delete\_row\_without\_traceback)

button.grid(column=0, row=2)

actionmenu.add\_command(label="Редактирование", command=edit)

actionmenu.add\_command(label="Удаление", command=delete\_window)

actionmenu.add\_command(label="Добавление", command=add\_win)

mainmenu.add\_cascade(label="Редактировать", menu=actionmenu)

def refresh(window, data\_frame\_1): # Refresh the Treeview to reflect changes

"""Функция обновляет данные в окне с выведенным справочнииком 2

На вход получает переменные:

window - окно с выведенным сравочником

data\_frame\_1 - DataFrame, содержащий считанный файл формата .csv"""

list1 = list(data\_frame\_1)

r\_set = data\_frame\_1.to\_numpy().tolist() # create list of list using rows

trv = ttk.Treeview(window, selectmode='browse', height=6,

show='headings', columns=list1)

trv.place(x=0, y=0, width=1100, height=800)

for i in list1:

trv.column(i, width=150)

trv.heading(i, text=str(i))

for dif\_t in r\_set:

value = list(dif\_t)

trv.insert("", 'end', iid=value[0], values=value)

refresh(self, data\_frame\_1)