**Федеральное государственное автономное образовательное учреждение высшего образования    
«Национальный исследовательский университет   
"Высшая школа экономики"»**

**Московский институт электроники и математики им. А.Н. Тихонова**

**НИУ ВШЭ**

**Департамент компьютерной инженерии**

**Курс: Проектный семинар "Python в науке о данных"**

**ОТЧЁТ по самостоятельной работе:**

**Руководство разработчика**

**Работу выполнили студенты бригады №9 группы БИВ 222:**

**Мишенин Роман Максимович**

**(+79152014847)**

**Некеров Ярослав Александрович**

**МОСКВА 2023**

**Оглавление**

[Требования к характеристикам компьютера 4](#_Toc9689)

[Версии языка и библиотеки 5](#_Toc3779)

[Описание архитектуры приложения 6](#_Toc23976)

[Главное окно 6](#_Toc20227)

[Интерфейс основного справочника 7](#_Toc12291)

[Интерфейс справочника владельцев 8](#_Toc16443)

[Всплывающее окно удаления 9](#_Toc16020)

[Всплывающее окно экспорта 10](#_Toc7287)

[Интерфейс отчёта "Необходимость поставки" 11](#_Toc1589)

[Интерфейс отчёта "Загруженность стеллажа" 12](#_Toc21561)

[Интерфейс отчёта "Продукты от владельца" 13](#_Toc18472)

[Интерфейс отчёта "Товары на стеллаж" 14](#_Toc4951)

[Интерфейс отчёта "Вес на стеллажах" 15](#_Toc14851)

[Структура каталогов 16](#_Toc30153)

[Листинг 17](#_Toc6599)

[Модуль main.py 17](#_Toc13255)

[Модуль specialfunc.py 26](#_Toc28683)

[Модуль generalfunc.py 45](#_Toc8617)

# Требования к характеристикам компьютера

Для работы приложения должна быть создана виртуальная среда исключительно средствами менеджера пакетов conda из актуального дистрибутива Anaconda. Таким образом, приложение должно работать в собственной виртуальной среде на любом компьютере, соответствующем требованиям данного документа, без необходимости добавления библиотек к актуальному дистрибутиву Anaconda.

Приложение является локальным приложением с графическим интерфейсом, которое работает на изолированном компьютере. Это означает, что для работы приложения не нужны вычислительные ресурсы Интернета и не используется клиентсерверная архитектура. В то же время приложение может анализировать данные из Интернета, если перевести их в соотвествующий формат (CSV) и поместить в каталог data. Также приложение должно исполняться на компьютере средней мощности с 8Gb оперативной памяти и 2Gb видеопамяти не очень мощной видеокарты.

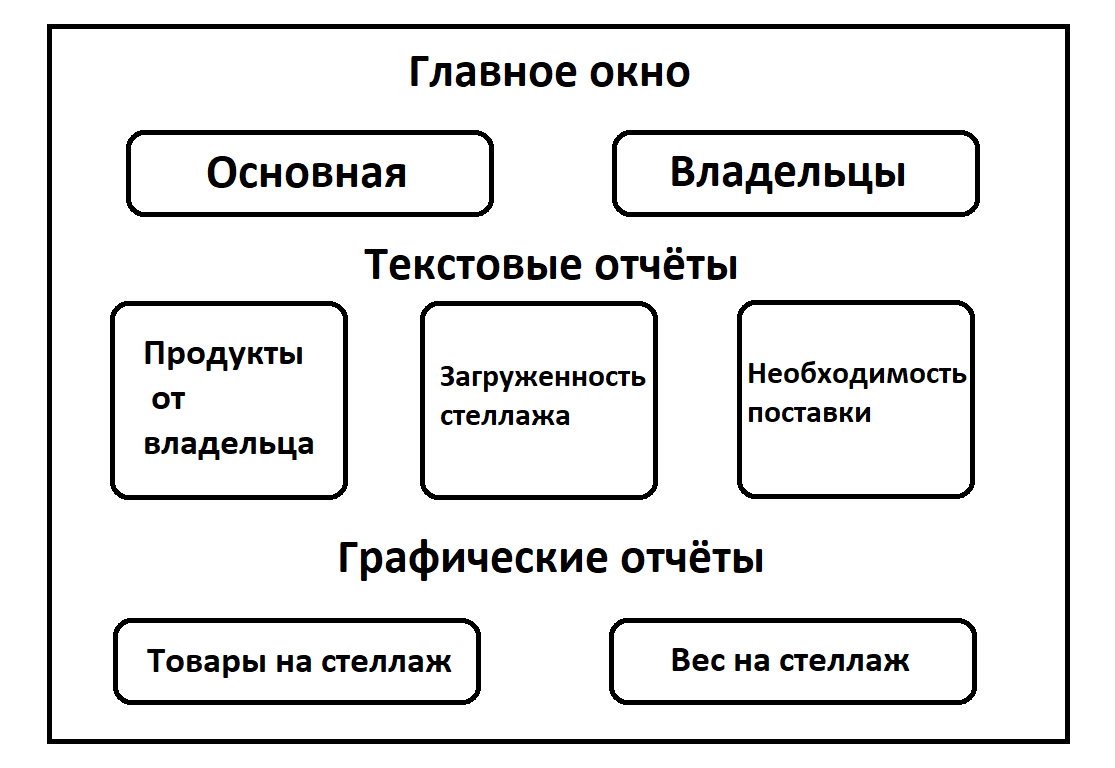
# Версии языка и библиотеки

Версия языка Python: 3.9

Актуальные версии библиотек Pandas, NumPy, Matplotlib и Tkinter содержатся в дистрибутиве Anaconda, настоятельно рекомендуемом к скачиванию для работы с данным приложением.

**Описание архитектуры приложения**

**Главное окно**

****

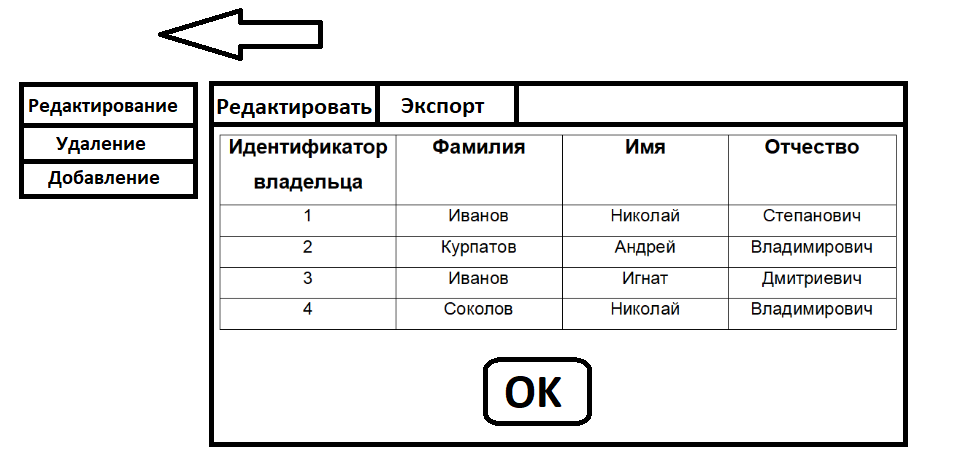
* Кнопки сделаны с помощью виджета Button
* Надписи сделаны с помощью виджета Label
* Окно сделано с помощью виджета Tk

**Интерфейс основного справочника**

****

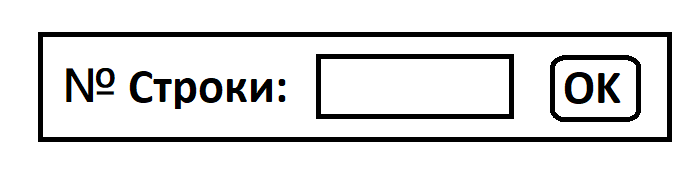
* Кнопка сделана с помощью виджета Button
* Окно сделано с помощью виджета Tk
* Меню сделано с помощью виджета Menu
* Таблица сделана при помощи виджетов Entry
* При нажатии на "удаление" и "экспорт" всплывают отдельные меню.
* При нажатии на "редактирование" открывается доступ к ячейкам таблицы. Чтобы закончить работу, нужно нажать OK.
* При нажатии на "добавление" появляется новая пустая строка. Чтобы закончить работу, нужно нажать OK.

**Интерфейс справочника владельцев**

****

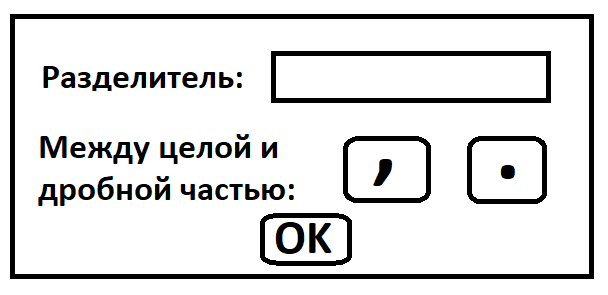
* Окно сделано с помощью виджета Tk
* Меню сделано с помощью виджета Menu
* Таблица сделана при помощи виджетов Entry
* При нажатии на "удаление" и "экспорт" всплывают отдельные меню.
* При нажатии на "редактирование" открывается доступ к ячейкам таблицы. Чтобы закончить работу, нужно нажать OK.
* При нажатии на "добавление" появляется новая пустая строка. Чтобы закончить работу, нужно нажать OK.

**Всплывающее окно удаления**

****

* Окно сделано с помощью виджета Tk
* Поле ввода сделано с помощью виджета Entry
* Кнопка сделана с помощью виджета Button
* Надписи сделаны с помощью виджета Label

**Всплывающее окно экспорта**

****

* Окно сделано с помощью виджета Tk
* Поле ввода сделано с помощью виджета Entry
* Кнопки сделаны с помощью виджета Button
* Надписи сделаны с помощью виджета Label

**Интерфейс отчёта "Необходимость поставки"**

****

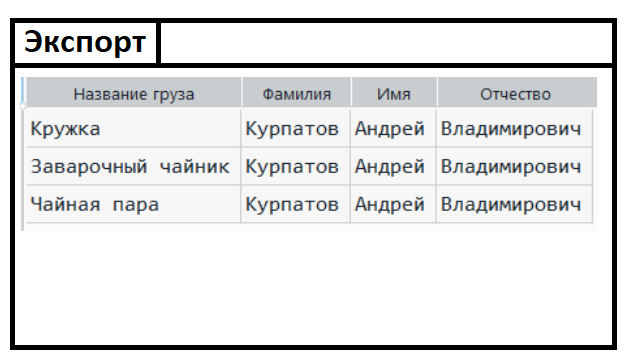
* Окно сделано с помощью виджета Tk
* Меню сделано с помощью виджета Menu
* Таблица сделана при помощи виджетов Entry
* При нажатии на "экспорт" всплывает отдельные меню

**Интерфейс отчёта "Загруженность стеллажа"**

****

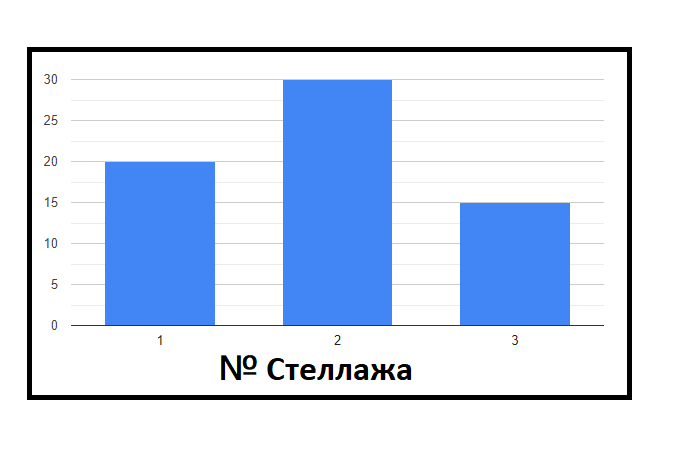
* Окно сделано с помощью виджета Tk
* Меню сделано с помощью виджета Menu
* Таблица сделана при помощи виджетов Entry
* При нажатии на "экспорт" всплывает отдельные меню

**Интерфейс отчёта "Продукты от владельца"**

****

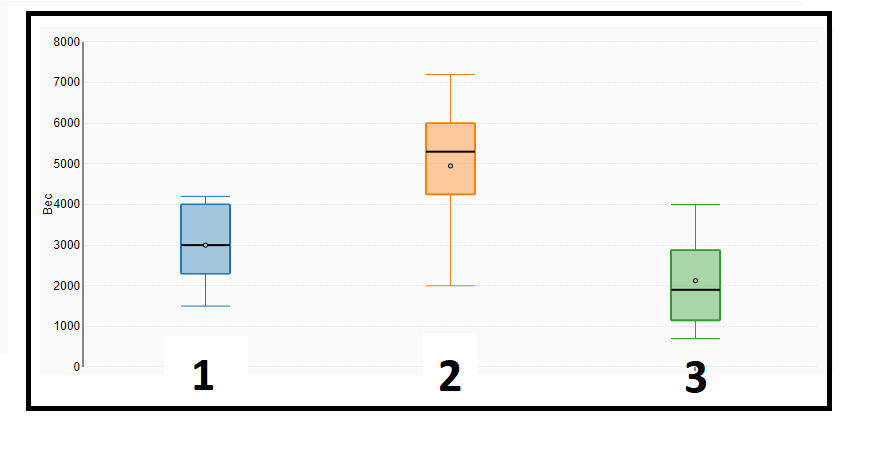
* Окно сделано с помощью виджета Tk
* Меню сделано с помощью виджета Menu
* Таблица сделана при помощи виджетов Entry
* При нажатии на "экспорт" всплывает отдельные меню

**Интерфейс отчёта "Товары на стеллаж"**

****

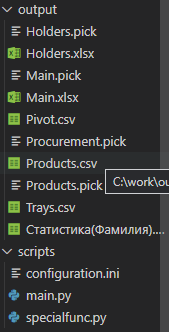
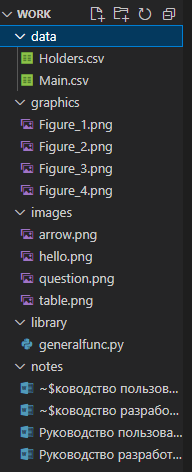
* Окно сделано с помощью виджета Tk
* Надписи сделаны с помощью виджета Label
* Диаграмма сделана с помощью виджета FigureCanvasTkAdd

**Интерфейс отчёта "Вес на стеллажах"**

****

* Окно сделано с помощью виджета Tk
* Надписи сделаны с помощью виджета Label
* Диаграмма сделана с помощью виджета FigureCanvasTkAdd

**Структура каталогов**

****

**Листинг**

**Модуль main.py**

import tkinter as tk

import pandas as pd

import configparser

import sys

sys.path.append("/work/library")

from generalfunc import create\_guide

from specialfunc import (

ask\_param,

ask\_stat,

graph\_weight,

graph\_goods,

ask\_pivot,

graph\_dist,

graph\_load,

)

# чтение данных из ini файла и присваивание значений переменной

config = configparser.ConfigParser()

config.read("/work/scripts/configuration.ini")

colorback = config["settings"]["colorback"]

colortext = config["settings"]["colortext"]

bt\_font = config["settings"]["bt\_font"].split(",")

bt\_font[1] = int(bt\_font[1])

bt\_font = tuple(bt\_font)

lb\_font = config["settings"]["lb\_font"].split(",")

lb\_font[1] = int(lb\_font[1])

lb\_font = tuple(lb\_font)

root = tk.Tk()

root.geometry("700x640+650+40")

root.focus\_force()

root.title("Главное окно")

root.resizable(False, False)

root.config(bg=colorback)

ph = tk.PhotoImage(file="/work/images/hello.png")

root.iconphoto(False, ph)

lb\_tables = tk.Label(root, text="Таблицы", fg=colortext, bg=colorback, font=lb\_font)

lb\_tables.place(x=270, y=0)

bt\_main = tk.Button(

root,

text="Основная",

fg=colorback,

bg=colortext,

width=19,

height=2,

command=lambda: create\_guide(

root, "Основной справочник", "600x250+30+30", "/work/data/Main.csv"

),

font=bt\_font,

relief="raised",

bd=1,

)

bt\_main.place(x=20, y=55)

bt\_hold = tk.Button(

root,

text="Владельцы",

fg=colorback,

bg=colortext,

width=18,

height=2,

command=lambda: create\_guide(

root, "Справочник владельцев", "500x150+30+350", "/work/data/Holders.csv"

),

font=bt\_font,

relief="raised",

bd=1,

)

bt\_hold.place(x=365, y=55)

lb\_text = tk.Label(

root, text="Текстовые отчёты", fg=colortext, bg=colorback, font=lb\_font

)

lb\_text.place(x=200, y=145)

bt\_prod = tk.Button(

root,

text="Продукты от\n владельца",

fg=colorback,

bg=colortext,

width=12,

height=2,

command=lambda: ask\_param(

root,

'Отчёт "продукты от владельца"',

"500x100+30+580",

pd.merge(

pd.read\_csv("/work/data/Main.csv", delimiter=",", encoding="utf8"),

pd.read\_csv("/work/data/Holders.csv", delimiter=",", encoding="utf8"),

on="Владелец №",

),

"/work/output/Products.csv",

"Введите № владельца:",

),

font=bt\_font,

relief="raised",

bd=1,

)

bt\_prod.place(x=20, y=200)

bt\_tray = tk.Button(

root,

text="Загруженность\nстеллажа",

fg=colorback,

bg=colortext,

width=12,

height=2,

command=lambda: ask\_param(

root,

'Отчёт "загруженность стеллажа"',

"350x100+156+484",

pd.merge(

pd.read\_csv("/work/data/Main.csv", delimiter=",", encoding="utf8"),

pd.read\_csv("/work/data/Holders.csv", delimiter=",", encoding="utf8"),

on="Владелец №",

),

"/work/output/Trays.csv",

"Введите № стеллажа:",

),

font=bt\_font,

relief="raised",

bd=1,

)

bt\_tray.place(x=235, y=200)

bt\_cure = tk.Button(

root,

text="Необходимость\nпоставки",

fg=colorback,

bg=colortext,

width=13,

height=2,

command=lambda: ask\_param(

root,

'Отчёт "необходимость поставки"',

"400x200+195+143",

pd.merge(

pd.read\_csv("/work/data/Main.csv", delimiter=",", encoding="utf8"),

pd.read\_csv("/work/data/Holders.csv", delimiter=",", encoding="utf8"),

on="Владелец №",

),

"/work/output/Procurement.csv",

"Минимальное кол-во:",

),

font=bt\_font,

relief="raised",

bd=1,

)

bt\_cure.place(x=450, y=200)

bt\_stat = tk.Button(

root,

text="Общая статистика",

fg=colorback,

bg=colortext,

width=19,

height=2,

font=bt\_font,

command=lambda: ask\_stat(

root,

pd.merge(

pd.read\_csv("/work/data/Main.csv", delimiter=",", encoding="utf8"),

pd.read\_csv("/work/data/Holders.csv", delimiter=",", encoding="utf8"),

on="Владелец №",

),

),

relief="raised",

bd=1,

)

bt\_stat.place(x=20, y=290)

bt\_pivot = tk.Button(

root,

text="Сводная таблица",

fg=colorback,

bg=colortext,

width=18,

height=2,

font=bt\_font,

command=lambda: ask\_pivot(

root,

pd.merge(

pd.read\_csv("/work/data/Main.csv", delimiter=",", encoding="utf8"),

pd.read\_csv("/work/data/Holders.csv", delimiter=",", encoding="utf8"),

on="Владелец №",

),

),

relief="raised",

bd=1,

)

bt\_pivot.place(x=365, y=290)

lb\_graph = tk.Label(

root, text="Графические отчёты", fg=colortext, bg=colorback, font=lb\_font

)

lb\_graph.place(x=185, y=380)

bt\_goods = tk.Button(

root,

text="Товары на стеллаж",

fg=colorback,

bg=colortext,

width=19,

height=2,

command=lambda: graph\_goods(

pd.merge(

pd.read\_csv("/work/data/Main.csv", delimiter=",", encoding="utf8"),

pd.read\_csv("/work/data/Holders.csv", delimiter=",", encoding="utf8"),

on="Владелец №",

)

),

font=bt\_font,

relief="raised",

bd=1,

)

bt\_goods.place(x=20, y=440)

bt\_weight = tk.Button(

root,

text="Вес на стеллажах",

fg=colorback,

bg=colortext,

width=18,

height=2,

command=lambda: graph\_weight(

pd.merge(

pd.read\_csv("/work/data/Main.csv", delimiter=",", encoding="utf8"),

pd.read\_csv("/work/data/Holders.csv", delimiter=",", encoding="utf8"),

on="Владелец №",

)

),

font=bt\_font,

relief="raised",

bd=1,

)

bt\_weight.place(x=365, y=440)

bt\_dist = tk.Button(

root,

text="Распределение весов",

fg=colorback,

bg=colortext,

width=19,

height=2,

command=lambda: graph\_dist(

pd.merge(

pd.read\_csv("/work/data/Main.csv", delimiter=",", encoding="utf8"),

pd.read\_csv("/work/data/Holders.csv", delimiter=",", encoding="utf8"),

on="Владелец №",

)

),

font=bt\_font,

relief="raised",

bd=1,

)

bt\_dist.place(x=20, y=530)

bt\_load = tk.Button(

root,

text="Загруженность склада",

fg=colorback,

bg=colortext,

width=18,

height=2,

command=lambda: graph\_load(

pd.merge(

pd.read\_csv("/work/data/Main.csv", delimiter=",", encoding="utf8"),

pd.read\_csv("/work/data/Holders.csv", delimiter=",", encoding="utf8"),

on="Владелец №",

)

),

font=bt\_font,

relief="raised",

bd=1,

)

bt\_load.place(x=365, y=530)

root.mainloop()

**Модуль specialfunc.py**

import tkinter as tk

import pandas as pd

import tkinter.messagebox as mb

import numpy as np

import configparser

import matplotlib.pyplot as plt

import sys

sys.path.append("/work/library")

from generalfunc import export, onFrameConfigure

from tkinter import ttk

# чтение данных из ini файла и присваивание значений переменной

config = configparser.ConfigParser()

config.read("/work/scripts/configuration.ini")

colorback = config["settings"]["colorback"]

colortext = config["settings"]["colortext"]

bt\_font = config["settings"]["bt\_font"].split(",")

bt\_font[1] = int(bt\_font[1])

bt\_font = tuple(bt\_font)

lb\_font = config["settings"]["lb\_font"].split(",")

lb\_font[1] = int(lb\_font[1])

lb\_font = tuple(lb\_font)

def create\_report(

root: tk.Tk,

parent: tk.Toplevel,

title: str,

coord: str,

NAME: pd.DataFrame,

param: str,

path: str,

):

"""\_summary\_

Args:

root (tk.Tk): корневое окно

parent (tk.Toplevel): ссылка на предыдущее окно (чтобы удалить)

title (str): название окна

coord (str): размеры окна

NAME (pd.DataFrame): DataFrame

param (str): ключевой параметр, по которому будет строиться 1 из 3 отчетов

path (str): путь к файлу

"""

flag = False

if title == 'Отчёт "продукты от владельца"':

if (

not param.isdigit()

or int(param) > NAME["Владелец №"].max()

or int(param) <= 0

):

mb.showerror(

"Неверный идентификатор владельца",

"Введите корректный номер владельца!",

)

parent.focus\_force()

else:

flag = True

SEL = NAME["Владелец №"] == int(param)

NAME = NAME.loc[SEL, ["Название", "Фамилия", "Имя", "Отчество"]]

elif title == 'Отчёт "загруженность стеллажа"':

if (

not param.isdigit()

or int(param) > NAME["Стеллаж №"].max()

or int(param) <= 0

):

mb.showerror(

"Неверный номер стеллажа", "Введите корректный номер стеллажа!"

)

parent.focus\_force()

else:

flag = True

SEL = NAME["Стеллаж №"] == int(param)

NAME = NAME.loc[SEL, ["Название", "Остаток", "Масса (кг)"]]

else:

if not param.isdigit() or int(param) <= 0:

mb.showerror(

"Неверное количество товара", "Введите корректный количество товара!"

)

parent.focus\_force()

else:

flag = True

SEL = NAME["Остаток"] <= int(param)

NAME = NAME.loc[SEL, ["Название", "Остаток", "Владелец №"]]

if flag:

parent.destroy()

win = tk.Toplevel(root)

win.geometry(coord)

mainmenu = tk.Menu(win)

win.config(menu=mainmenu)

win.focus\_force()

ph = tk.PhotoImage(file="/work/images/table.png")

win.iconphoto(False, ph)

win.resizable(False, True)

win.title(title)

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

frame = tk.Frame(canvas)

scroll = tk.Scrollbar(win, orient="vertical", command=canvas.yview)

canvas.configure(yscrollcommand=scroll.set)

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

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

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

frame.bind("<Configure>", lambda event, canvas=canvas: onFrameConfigure(canvas))

height = NAME.shape[0]

width = NAME.shape[1]

mainmenu.add\_command(label="Экспорт", command=lambda: export(win, NAME, path))

pnt = np.empty(shape=(height + 1, width), dtype="O")

vrs = np.empty(shape=(height + 1, width), dtype="O")

for i in range(height + 1):

for j in range(width):

if i == 0:

e = tk.Entry(frame, relief="ridge")

e.grid(row=0, column=j, sticky=tk.E)

e.insert(tk.END, NAME.columns[j])

e.config(state="readonly")

else:

vrs[i, j] = tk.StringVar()

pnt[i, j] = tk.Entry(frame, textvariable=vrs[i, j])

pnt[i, j].grid(row=i, column=j)

cnt = NAME.iloc[i - 1, j]

vrs[i, j].set(str(cnt))

pnt[i, j].config(state="readonly")

def ask\_param(

root: tk.Tk, title: str, coord: str, NAME: pd.DataFrame, path: str, msg: str

):

"""\_summary\_

Args:

root (tk.Tk): корневое окно

title (str): название окна

coord (str): размеры окна

NAME (pd.DataFrame): DataFrame

path (str): путь к файлу, куда идет сохранение

msg (str): сообщение, которое выводится пользователю

"""

win = tk.Toplevel(root)

win.geometry("500x150+700+200")

ph = tk.PhotoImage(file="/work/images/question.png")

win.iconphoto(False, ph)

win.resizable(False, False)

win.title("Окно выбора")

win.config(bg=colorback)

lb\_msg = tk.Label(win, text=msg, fg=colortext, bg=colorback, font=lb\_font)

lb\_msg.place(x=20, y=20)

entry = tk.Entry(win, font=("Arial", 18, "bold"), width=5)

entry.place(x=420, y=30)

btn = tk.Button(

win,

text="OK",

font=bt\_font,

bg=colortext,

fg=colorback,

command=lambda: create\_report(root, win, title, coord, NAME, entry.get(), path),

)

btn.place(x=200, y=80)

if title == 'Отчёт "продукты от владельца"':

mb.showinfo(

"Инструкция",

"Введите номер владельца, чтобы получить информацию о всех его товарах",

)

elif title == 'Отчёт "загруженность стеллажа"':

mb.showinfo(

"Инструкция",

"Введите номер стеллажа, чтобы получить информацию о товарах, расположенных на нём",

)

else:

mb.showinfo(

"Инструкция",

"Введите минимальное количество единиц товара, чтобы узнать, что нужно заказать",

)

entry.focus\_force()

def ask\_stat(root: tk.Tk, NAME: pd.DataFrame):

"""\_summary\_

Args:

root (tk.Tk): ссылка на корневое окно

NAME (pd.DataFrame): DatFrame

Функция узнает у пользователя, отчёт по какому атрибуту он хотел бы увидеть

"""

win = tk.Toplevel(root)

win.geometry("430x120+700+200")

ph = tk.PhotoImage(file="/work/images/question.png")

win.iconphoto(False, ph)

win.resizable(False, False)

win.title("Окно выбора")

win.config(bg=colorback)

lb\_msg = tk.Label(

win, text="Выберите переменную:", fg=colortext, bg=colorback, font=lb\_font

)

lb\_msg.place(x=10, y=0)

com\_col = ttk.Combobox(

win,

values=[

"Название",

"Владелец №",

"Остаток",

"Стеллаж №",

"Масса (кг)",

"Фамилия",

"Имя",

"Отчество",

],

font=bt\_font,

width=12,

state="readonly",

)

com\_col.current(0)

com\_col.place(x=30, y=60)

com\_col.focus\_force()

btn = tk.Button(

win,

text="OK",

font=bt\_font,

bg=colortext,

fg=colorback,

command=lambda: create\_stat(root, win, NAME, com\_col.get()),

)

btn.place(x=330, y=50)

def create\_stat(root: tk.Tk, parent: tk.Toplevel, NAME: pd.DataFrame, answer: str):

"""\_summary\_

Args:

root (tk.Tk): ссылка на корневое окно

parent (tk.Toplevel): ссылка на предыдущее окно

NAME (pd.DataFrame): DataFrame

answer (str): ответ пользователя, который получаем из функции ask\_stat()

"""

parent.destroy()

win = tk.Toplevel(root)

mainmenu = tk.Menu(win)

ph = tk.PhotoImage(file="/work/images/table.png")

win.iconphoto(False, ph)

win.config(menu=mainmenu)

win.focus\_force()

ph = tk.PhotoImage(file="/work/images/table.png")

win.iconphoto(False, ph)

win.resizable(True, True)

win.title(f'"{answer}"')

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

frame = tk.Frame(canvas)

scroll = tk.Scrollbar(win, orient="vertical", command=canvas.yview)

canvas.configure(yscrollcommand=scroll.set)

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

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

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

frame.bind("<Configure>", lambda event, canvas=canvas: onFrameConfigure(canvas))

mainmenu.add\_command(

label="Экспорт",

command=lambda: export(win, NAME, f"/work/data/Статистика({answer}).csv"),

)

if answer in ["Название", "Фамилия", "Имя", "Отчество", "Владелец №", "Стеллаж №"]:

win.geometry("350x200+100+100")

W = NAME[answer].value\_counts().to\_frame().reset\_index()

percents = []

for i in range(W.shape[0]):

percents.append((W[answer].loc[W.index[i]] / NAME.shape[0] \* 100).round(5))

W["Проценты (%)"] = percents

W.rename(

columns={"index": f"{answer}", f"{answer}": "Количество"}, inplace=True

)

height = W.shape[0]

width = W.shape[1]

pnt = np.empty(shape=(height + 1, width), dtype="O")

vrs = np.empty(shape=(height + 1, width), dtype="O")

for i in range(height + 1):

for j in range(width):

if i == 0:

e = tk.Entry(frame, relief="ridge")

e.grid(row=0, column=j, sticky=tk.E)

e.insert(tk.END, W.columns[j])

e.config(state="readonly")

else:

vrs[i, j] = tk.StringVar()

pnt[i, j] = tk.Entry(frame, textvariable=vrs[i, j])

pnt[i, j].grid(row=i, column=j)

cnt = W.iloc[i - 1, j]

vrs[i, j].set(str(cnt))

pnt[i, j].config(state="readonly")

else:

win.geometry("250x140+50+50")

NAME = NAME[answer].describe().to\_frame().reset\_index()

height = 8

width = 2

pnt = np.empty(shape=(height, width), dtype="O")

vrs = np.empty(shape=(height, width), dtype="O")

lst = [

"Количество",

"Среднее",

"СТД Отклонение",

"Минимум",

"25%",

"50%",

"75%",

"Максимум",

]

for i in range(height):

e = tk.Entry(frame, relief="ridge")

e.grid(row=i, column=0, sticky=tk.E)

e.insert(tk.END, lst[i])

e.config(state="readonly")

for i in range(height):

vrs[i, 1] = tk.StringVar()

pnt[i, 1] = tk.Entry(frame, textvariable=vrs[i, 1])

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

cnt = NAME.iloc[i, 1]

vrs[i, 1].set(str(cnt.round(5)))

pnt[i, 1].config(state="readonly")

def ask\_pivot(root: tk.Tk, NAME: pd.DataFrame):

"""\_summary\_

Args:

root (tk.Tk): ссылка на корневое окно

NAME (pd.DataFrame): DataFrame

Функция запрашивает у пользователя параметры для формирования сводной таблицы

"""

win = tk.Toplevel(root)

win.geometry("600x220+700+200")

ph = tk.PhotoImage(file="/work/images/question.png")

win.iconphoto(False, ph)

win.resizable(False, False)

win.title("Окно выбора")

win.config(bg=colorback)

lb\_msg = tk.Label(

win,

text="Выберите РАЗЛИЧНЫЕ переменные:",

fg=colortext,

bg=colorback,

font=bt\_font,

)

lb\_msg.place(x=10, y=0)

com\_1 = ttk.Combobox(

win,

values=["Название", "Владелец №", "Стеллаж №", "Фамилия", "Имя", "Отчество"],

font=bt\_font,

width=12,

state="readonly",

)

com\_1.current(0)

com\_1.place(x=30, y=50)

com\_1.focus\_force()

com\_2 = ttk.Combobox(

win,

values=["Название", "Владелец №", "Стеллаж №", "Фамилия", "Имя", "Отчество"],

font=bt\_font,

width=12,

state="readonly",

)

com\_2.current(1)

com\_2.place(x=330, y=50)

lb\_agg = tk.Label(

win,

text="Выберите агрегирующую функцию:",

fg=colortext,

bg=colorback,

font=bt\_font,

)

lb\_agg.place(x=10, y=100)

com\_agg = ttk.Combobox(

win,

values=[

"mean (Среднее)",

"median (Медиана)",

" min (Минимум)",

"max (Максимум)",

"std (СТД Отклонение)",

"sum (Сумма)",

],

font=bt\_font,

width=15,

state="readonly",

)

com\_agg.current(0)

com\_agg.place(x=70, y=160)

btn = tk.Button(

win,

text="OK",

font=bt\_font,

bg=colortext,

fg=colorback,

command=lambda: create\_pivot(

root, win, NAME, com\_1.get(), com\_2.get(), com\_agg.get()

),

)

btn.place(x=370, y=150)

def create\_pivot(

root: tk.Tk,

parent: tk.Toplevel,

NAME: pd.DataFrame,

par1: str,

par2: str,

aggr: str,

):

"""\_summary\_

Args:

root (tk.Tk): ссылка на корневое окно

parent (tk.Toplevel): ссылка на окно с выбором (чтобы удалить)

NAME (pd.DataFrame): DataFrame

par1 (str): первый параметр (строки)

par2 (str): второй параметр (столбцы)

aggr (str): аггрегирующая функция

"""

parent.destroy()

if par1 == par2:

mb.showerror("Ошибка выбора", "Введите РАЗЛИЧНЫЕ параметры!")

ask\_pivot(root, NAME)

else:

win = tk.Toplevel(root)

win.geometry("600x300+100+100")

ph = tk.PhotoImage(file="/work/images/table.png")

win.iconphoto(False, ph)

mainmenu = tk.Menu(win)

win.config(menu=mainmenu)

win.focus\_force()

ph = tk.PhotoImage(file="/work/images/table.png")

win.iconphoto(False, ph)

win.resizable(False, True)

win.title("Сводная таблица")

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

frame = tk.Frame(canvas)

scrollv = tk.Scrollbar(win, orient="vertical", command=canvas.yview)

canvas.configure(yscrollcommand=scrollv.set)

scrollg = tk.Scrollbar(win, orient="horizontal", command=canvas.xview)

canvas.configure(xscrollcommand=scrollg.set)

scrollg.pack(side="bottom", fill="x")

scrollv.pack(side="right", fill="y")

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

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

frame.bind("<Configure>", lambda event, canvas=canvas: onFrameConfigure(canvas))

NAME = NAME.pivot\_table(

index=par1,

columns=par2,

values=["Масса (кг)", "Остаток"],

aggfunc=aggr[: aggr.find("(") - 1],

)

height = NAME.shape[0]

width = NAME.shape[1]

mainmenu.add\_command(

label="Экспорт", command=lambda: export(win, NAME, "/work/data/Pivot.csv")

)

pnt = np.empty(shape=(height + 2, width + 1), dtype="O")

vrs = np.empty(shape=(height + 2, width + 1), dtype="O")

for i in range(height + 1):

e = tk.Entry(frame, relief="ridge")

e.grid(row=i, column=0, sticky=tk.E)

if i == 0:

e.insert(tk.END, "Мульти-индекс")

else:

e.insert(tk.END, NAME.index[i - 1])

e.config(state="readonly")

for j in range(1, width + 1):

if i == 0:

e = tk.Entry(frame, relief="ridge")

e.grid(row=0, column=j, sticky=tk.E)

e.insert(

tk.END,

str(NAME.columns[j - 1])

.replace("{", "")

.replace("}", "")

.replace("'", ""),

)

e.config(state="readonly")

else:

vrs[i, j] = tk.StringVar()

pnt[i, j] = tk.Entry(frame, textvariable=vrs[i, j])

pnt[i, j].grid(row=i, column=j)

cnt = NAME.iloc[i - 1, j - 1]

vrs[i, j].set(str(cnt.round(1)).replace("nan", "-"))

pnt[i, j].config(state="readonly")

def graph\_goods(NAME: pd.DataFrame):

"""\_summary\_

Args:

NAME (pd.DataFrame): DataFrame

Создаёт графический отчёт типа стобчатая диаграмма

"""

NAME = NAME.groupby(["Стеллаж №"]).agg({"Остаток": "sum"}).reset\_index()

trays = NAME["Стеллаж №"].to\_list()

number = NAME["Остаток"].to\_list()

fig, ax = plt.subplots()

bar\_container = ax.bar(trays, number)

ax.set(

ylabel="Количество",

title="Остаток товаров на стеллажах",

ylim=(0, max(number) + 30),

xlabel="Стеллаж №",

)

ax.bar\_label(bar\_container)

plt.show()

def graph\_weight(NAME: pd.DataFrame):

"""\_summary\_

Args:

NAME (pd.DataFrame): DataFrame

Создаёт графический отчёт типа диаграмма Бокса-Вискера

"""

values = []

for i in range(1, NAME["Стеллаж №"].max() + 1):

values.append(mass\_on\_each\_tray(NAME, i))

fig = plt.figure()

ax = fig.add\_subplot(111)

ax.boxplot(values)

ax.set\_xlabel("Стеллаж №")

ax.set\_ylabel("Масса (кг)")

plt.title("Загруженность стеллажей")

plt.show()

def mass\_on\_each\_tray(NAME: pd.DataFrame, num: int) -> list:

"""\_summary\_

Args:

NAME (pd.DataFrame): DataFrame

num (int): номер стеллажа, для которого нужно создать список с массами всех товаров, лежащих на нём

Returns:

list: список с массами товаров

"""

lst = []

for i in range(NAME.shape[0]):

if NAME["Стеллаж №"].loc[NAME.index[i]] == num:

lst.append(

NAME["Остаток"].loc[NAME.index[i]]

\* NAME["Масса (кг)"].loc[NAME.index[i]]

)

lst = list(np.around(np.array(lst), 1))

return lst

def graph\_dist(NAME: pd.DataFrame):

"""\_summary\_

Args:

NAME (pd.DataFrame): DataFrame

Создаёт графический отчёт типа гистограмма

"""

data = NAME["Масса (кг)"].to\_list()

fig, ax = plt.subplots()

bar\_container = ax.hist(data, bins=100)

plt.hist(data, bins=50)

ax.set(ylabel="Количество", xlabel="Масса (кг)")

plt.title("Распределение масс")

plt.show()

def graph\_load(NAME: pd.DataFrame):

"""\_summary\_

Args:

NAME (pd.DataFrame): DataFrame

Создаёт графический отчёт типа диаграмма рассеивания

"""

weights = NAME["Масса (кг)"].to\_list()

lefts = NAME["Остаток"].to\_list()

res\_list = [weights[i] \* lefts[i] for i in range(len(weights))]

fig, ax = plt.subplots()

bar\_container = ax.scatter(weights, lefts, s=res\_list, c="orange")

ax.set(ylabel="Остаток", xlabel="Масса (кг)")

plt.text(18, 100, "Размер круга = [кг\*шт]")

plt.title("Загруженность склада")

plt.show()

**Модуль generalfunc.py**

import tkinter as tk

import pandas as pd

from tkinter import ttk

import numpy as np

import configparser

import tkinter.messagebox as mb

# чтение данных из ini файла и присваивание значений переменной

config = configparser.ConfigParser()

config.read("/work/scripts/configuration.ini")

colorback = config["settings"]["colorback"]

colortext = config["settings"]["colortext"]

bt\_font = config["settings"]["bt\_font"].split(",")

bt\_font[1] = int(bt\_font[1])

bt\_font = tuple(bt\_font)

lb\_font = config["settings"]["lb\_font"].split(",")

lb\_font[1] = int(lb\_font[1])

lb\_font = tuple(lb\_font)

def export(root: tk.Tk, NAME: pd.DataFrame, path: str):

"""\_summary\_

Args:

root (tk.Tk): корневой элемент

NAME (pd.DataFrame): DataFrame

path (str): путь к исходному файлу, откуда взяли DataFrame

Функция собирает параметры экспорта, а по нажатию кнопки совершает его

"""

win = tk.Toplevel(root)

win.geometry("330x270+700+250")

win.title("Окно экспорта")

win.focus\_force()

win.resizable(False, False)

win.config(bg=colorback)

ph = tk.PhotoImage(file="/work/images/arrow.png")

win.iconphoto(False, ph)

lb\_sep = tk.Label(

win, text="Разделитель:", fg=colortext, bg=colorback, font=bt\_font

)

lb\_sep.place(x=20, y=20)

ent\_sep = tk.Entry(win, font=bt\_font, width=5)

ent\_sep.place(x=220, y=20)

lb\_dec = tk.Label(

win, text="Десятичный\n разделитель:", fg=colortext, bg=colorback, font=bt\_font

)

lb\_dec.place(x=10, y=70)

lb\_form = tk.Label(win, text="Формат:", fg=colortext, bg=colorback, font=bt\_font)

lb\_form.place(x=20, y=140)

com\_form = ttk.Combobox(

win, values=["Excel", "CSV", "Pickle"], font=bt\_font, width=12, state="readonly"

)

com\_form.current(0)

com\_form.place(x=140, y=140)

com\_dec = ttk.Combobox(

win, values=["Запятая", "Точка"], font=bt\_font, width=8, state="readonly"

)

com\_dec.current(0)

com\_dec.place(x=190, y=80)

bt\_OK = tk.Button(

win,

text="OK",

fg=colorback,

bg=colortext,

command=lambda: appr\_exp(

ent\_sep.get(), com\_form.get(), com\_dec.get(), NAME, path

),

font=lb\_font,

relief="raised",

)

bt\_OK.place(x=140, y=190)

def appr\_exp(sep: str, form: str, dec: str, NAME: pd.DataFrame, path: str):

"""\_summary\_

Args:

sep (str): разделитель между значениям

form (str): формат (csv, xlsx, pick)

dec (str): десятичный разделитель

NAME (pd.DataFrame): DataFrame

path (str): путь к исходному файлу, откуда взяли DataFrame

Функция выгружает DataFrame в файловую систему

"""

# выбор формата экспорта

if form == "Excel":

NAME.to\_excel(

path.replace("data", "output").replace("csv", "xlsx"), index=False

)

mb.showinfo(

"Успешное сохранение",

"Выходной файл (excel) успешно сохранён в каталоге output",

)

elif form == "CSV":

if len(sep) != 1:

mb.showerror("Ошибка ввода", 'Введите в поле "разделитель" один символ!')

else:

if dec == "Запятая":

NAME.to\_csv(

path.replace("data", "output"), index=False, decimal=",", sep=sep

)

mb.showinfo(

"Успешное сохранение",

"Выходной файл (csv) успешно сохранён в каталоге output",

)

else:

NAME.to\_csv(

path.replace("data", "output"), index=False, decimal=".", sep=sep

)

mb.showinfo(

"Успешное сохранение",

"Выходной файл (csv) успешно сохранён в каталоге output",

)

else:

NAME.to\_pickle(path.replace("data", "output").replace("csv", "pick"))

mb.showinfo(

"Успешное сохранение",

"Выходной файл (pickle) успешно сохранён в каталоге output",

)

def create\_guide(root: tk.Tk, title: str, coord: str, path: str):

"""\_summary\_

Args:

root (tk.Tk): корневое окно

title (str): название окна

coord (str): размеры окна

path (str): путь к исходному файлу

Функция создает окно для показа справочников

"""

win = tk.Toplevel(root)

win.geometry(coord)

mainmenu = tk.Menu(win)

win.config(menu=mainmenu)

win.focus\_force()

ph = tk.PhotoImage(file="/work/images/table.png")

win.iconphoto(False, ph)

win.resizable(False, True)

win.title(title)

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

frame = tk.Frame(canvas)

scroll = tk.Scrollbar(win, orient="vertical", command=canvas.yview)

canvas.configure(yscrollcommand=scroll.set)

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

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

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

frame.bind("<Configure>", lambda event, canvas=canvas: onFrameConfigure(canvas))

NAME = pd.read\_csv(path, delimiter=",", encoding="utf8")

height = NAME.shape[0]

width = NAME.shape[1]

mainmenu.add\_command(label="Экспорт", command=lambda: export(win, NAME, path))

actionmenu = tk.Menu(mainmenu, tearoff=0)

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

actionmenu.add\_command(

label="Редактирование",

command=lambda: edit\_data(frame, NAME, pnt, win, root, path, title, coord),

)

actionmenu.add\_command(

label="Удаление",

command=lambda: delete\_data(frame, NAME, pnt, win, root, path, title, coord),

)

actionmenu.add\_command(

label="Добавление",

command=lambda: add\_data(frame, NAME, vrs, pnt, win, root, path, title, coord),

)

pnt = np.empty(shape=(height + 50, width), dtype="O")

vrs = np.empty(shape=(height + 50, width), dtype="O")

for i in range(height + 1):

for j in range(width):

if i == 0:

e = tk.Entry(frame, relief="ridge")

e.grid(row=0, column=j, sticky=tk.E)

e.insert(tk.END, NAME.columns[j])

e.config(state="readonly")

else:

vrs[i, j] = tk.StringVar()

pnt[i, j] = tk.Entry(frame, textvariable=vrs[i, j])

pnt[i, j].grid(row=i, column=j)

cnt = NAME.iloc[i - 1, j]

vrs[i, j].set(str(cnt))

pnt[i, j].config(state="readonly")

def onFrameConfigure(canvas: tk.Canvas):

"""\_summary\_

Args:

canvas (tk.Canvas): для обработки скроллбара

"""

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

def delete\_data(

frame: tk.Frame,

NAME: pd.DataFrame,

pnt: np.array,

parent: tk.Toplevel,

root: tk.Tk,

path: str,

title: str,

coord: str,

):

"""\_summary\_

Args:

frame (tk.Frame): фрейм, где находятся виджеты entry

NAME (pd.DataFrame): DataFrame

pnt (np.array): массив указателей на виджеты Entry

parent (tk.Toplevel): предыдущее окно (чтобы удалить его)

root (tk.Tk): корневое окно

path (str): путь к исходому файлу

title (str): название окна

coord (str): размеры окна

"""

mb.showinfo(

"Инструкция",

"Сотрите все значения в строке, которую хотите удалить, затем нажмите ОК (под последней строкой)",

)

parent.focus\_force()

# здесь и далее focus\_force для фокусировки на определённом окне

height = NAME.shape[0]

width = NAME.shape[1]

for i in range(1, height):

for j in range(width):

pnt[i, j].config(state="normal")

btn = tk.Button(

frame,

text="OK",

font=bt\_font,

bg=colortext,

fg=colorback,

command=lambda: appr\_delete(btn, NAME, pnt, parent, root, path, title, coord),

width=10,

)

btn.grid(row=height + 1, columnspan=6)

def appr\_delete(

btn: tk.Button,

NAME: pd.DataFrame,

pnt: np.array,

parent: tk.Toplevel,

root: tk.Tk,

path: str,

title: str,

coord: str,

):

"""\_summary\_

Args:

btn (tk.Button): ссылка на кнопку (чтобы удалить)

NAME (pd.DataFrame): DataFrame

pnt (np.array): массив указателей на виджеты Entry

parent (tk.Toplevel): предыдущее окно (чтобы удалить его)

root (tk.Tk): корневое окно

path (str): путь к исходому файлу

title (str): название окна

coord (str): размеры окна

"""

btn.destroy()

height = NAME.shape[0]

width = NAME.shape[1]

for i in range(1, height):

lst = []

for j in range(width):

pnt[i, j].config(state="readonly")

lst.append(pnt[i, j].get())

# поиск пустой строки

if lst == ["", "", "", "", ""]:

for k in range(i, height - 1):

# цикл для удаления (переприсваивание)

NAME.loc[k] = NAME.iloc[k + 1]

NAME.drop(NAME.tail(1).index, inplace=True)

NAME.to\_csv(path, index=False, decimal=",", sep=",")

mb.showinfo("Успешные изменения", "Изменения успешно сохранены")

parent.destroy()

create\_guide(root, title, coord, path)

def edit\_data(

frame: tk.Frame,

NAME: pd.DataFrame,

pnt: np.array,

parent: tk.Toplevel,

root: tk.Tk,

path: str,

title: str,

coord: str,

):

"""\_summary\_

Args:

frame (tk.Frame): фрейм, где находятся виджеты entry

NAME (pd.DataFrame): DataFrame

pnt (np.array): массив указателей на виджеты Entry

parent (tk.Toplevel): предыдущее окно (чтобы удалить его)

root (tk.Tk): корневое окно

path (str): путь к исходому файлу

title (str): название окна

coord (str): размеры окна

"""

mb.showinfo(

"Инструкция",

"Отредактируйте необходимые записи и нажмите ОК (под последней строкой)",

)

parent.focus\_force()

height = NAME.shape[0]

width = NAME.shape[1]

for i in range(1, height):

for j in range(width):

# здесь и далее состояние normal - можно писать, readonly - только чтение

pnt[i, j].config(state="normal")

btn = tk.Button(

frame,

text="OK",

font=bt\_font,

bg=colortext,

fg=colorback,

command=lambda: appr\_edit(btn, NAME, pnt, parent, root, path, title, coord),

width=10,

)

btn.grid(row=height + 1, columnspan=6)

def appr\_edit(

btn: tk.Button,

NAME: pd.DataFrame,

pnt: np.array,

parent: tk.Toplevel,

root: tk.Tk,

path: str,

title: str,

coord: str,

):

"""\_summary\_

Args:

btn (tk.Button): ссылка на кнопку (чтобы удалить)

NAME (pd.DataFrame): DataFrame

pnt (np.array): массив указателей на виджеты Entry

parent (tk.Toplevel): предыдущее окно (чтобы удалить его)

root (tk.Tk): корневое окно

path (str): путь к исходому файлу

title (str): название окна

coord (str): размеры окна

"""

btn.destroy()

height = NAME.shape[0]

width = NAME.shape[1]

for i in range(1, height):

lst = []

for j in range(width):

pnt[i, j].config(state="readonly")

lst.append(pnt[i, j].get())

NAME.loc[i] = lst

NAME.to\_csv(path, index=False, decimal=",", sep=",")

mb.showinfo("Успешные изменения", "Изменения успешно сохранены")

parent.destroy()

create\_guide(root, title, coord, path)

def add\_data(

frame: tk.Frame,

NAME: pd.DataFrame,

vrs: np.array,

pnt: np.array,

parent: tk.Toplevel,

root: tk.Tk,

path: str,

title: str,

coord: str,

):

"""\_summary\_

Args:

frame (tk.Frame): фрейм, где находятся виджеты entry

NAME (pd.DataFrame): DataFrame

vrs (np.array): массив указателей на буферы

pnt (np.array): массив указателей на виджеты Entry

parent (tk.Toplevel): предыдущее окно (чтобы удалить его)

root (tk.Tk): корневое окно

path (str): путь к исходому файлу

title (str): название окна

coord (str): размеры окна

"""

mb.showinfo(

"Инструкция", "Добавьте одну запись в появившемся снизу поле и нажмите ОК"

)

parent.focus\_force()

height = NAME.shape[0]

width = NAME.shape[1]

for j in range(width):

vrs[height, j] = tk.StringVar()

pnt[height, j] = tk.Entry(frame, textvariable=vrs[height, j])

pnt[height, j].grid(row=height, column=j)

vrs[height, j].set("")

btn = tk.Button(

frame,

text="OK",

font=bt\_font,

bg=colortext,

fg=colorback,

command=lambda: appr\_add(btn, NAME, pnt, parent, root, path, title, coord),

width=10,

)

btn.grid(row=height + 2, columnspan=6)

def appr\_add(

btn: tk.Button,

NAME: pd.DataFrame,

pnt: np.array,

parent: tk.Toplevel,

root: tk.Tk,

path: str,

title: str,

coord: str,

):

"""\_summary\_

Args:

btn (tk.Button): ссылка на кнопку (чтобы удалить)

NAME (pd.DataFrame): DataFrame

pnt (np.array): массив указателей на виджеты Entry

parent (tk.Toplevel): предыдущее окно (чтобы удалить его)

root (tk.Tk): корневое окно

path (str): путь к исходому файлу

title (str): название окна

coord (str): размеры окна

"""

height = NAME.shape[0] + 1

width = NAME.shape[1]

lst = []

btn.destroy()

for i in range(1, height):

for j in range(width):

pnt[i, j].config(state="readonly")

for j in range(width):

lst.append(pnt[height - 1, j].get())

NAME.loc[len(NAME.index)] = lst

NAME.to\_csv(path, index=False, decimal=",", sep=",")

mb.showinfo("Успешные изменения", "Изменения успешно сохранены")

parent.destroy()

create\_guide(root, title, coord, path)