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

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

**Курс: Python в науке о данных**

Руководство разработчика к приложению по базе данных, предназначенной для оценки вулканической активности в мире

Разработчики:

Студент Подкопаева Полина БИВ191

Студент Баканов Глеб БИВ191

Студент Ковязин Владислав БИВ191

Руководитель: Полякова Марина Васильевна

**Москва 2020**

СОДЕРЖАНИЕ

Руководство

[Технические требования: 1](#_Toc42978192)

[Версии библиотек: 2](#_Toc42978193)

[Архитектура приложения 2](#_Toc42978194)

[Структура каталогов 3](#_Toc42978195)

[Листинг скрипта 4](#_Toc42978196)

[1) Модуль main.py 4](#_Toc42978197)

[2) Модуль base\_handling.py 8](#_Toc42978198)

[3) Модуль constants.py 11](#_Toc42978199)

[4) Модуль globalvars.py 12](#_Toc42978200)

[5) Модуль interface.py 19](#_Toc42978201)

[6) Модуль new\_map.py 55](#_Toc42978202)

[7) Модуль new\_table.py 62](#_Toc42978203)

[8) Модуль statistics.py 75](#_Toc42978204)

# Технические требования:

* Оперативная память – не менее 2 Гб
* Двухъядерный процессор с минимальной частотой 1,8 ГГц
* Свободная память на диске – не менее 200 Мб
* Операционная система – Windows 7 и выше
* Интерпретатор языка Python – Python 3.7 и выше

# Версии библиотек:

Используется небольшой набор популярных библиотек Python.

* Numpy версия 1.15.4
* Pandas версия 0.23.4
* Pickle версия 4.0
* Matplotlib версия 3.0.2
* Tkinter версия 8.6
* Sys версия 3.7.1
* Os версия 3.2
* Folium версия 0.11.0

# Архитектура приложения

Данное приложение состоит из модулей, находящиеся в каталоге Work/Scripts. Главный из них, main.py, содержит управляющий код программы, обработку функций кнопок, формирование базы и т.д.

В модуле constants.py находятся константные переменные. С ним могут быть связаны остальные модули, некоторые из которых так же могут быть взаимосвязаны между собой.

В модуле globalvars.py содержатся несколько важных глобальных переменных и функций. С ним могут быть связаны остальные модули, некоторые из которых так же могут быть взаимосвязаны между собой.

Модуль error\_edit\_windows.py отвечает за вывод соответствующих окон, говорящих об ошибках, предупреждениях, задающих вопросы и т.д.

Модуль statistics.py строит и иллюстрирует (диаграммы, графики) статистику. Графические отчёты сохраняются в каталоге Work/Graphics.

Модуль new\_table.py строит сводные таблицы. Отчёты в формате csv сохраняются в каталоге Work/Output.

Модуль new\_map.py предназначен для построения карты. Отчёты в формате html сохраняются в каталоге Work/Graphics.

Модуль base\_handling.py содержит базовые функции обработки базы данных (сохранение, изменение, поиск, удаление и т.д.).

Модуль interface.py формирует графический интерфейс программы.

Все модули тем или иным образом используют библиотеку, находящуюся в каталоге Work/Library (модуль stdlib.py). Она содержит общие функции, которые так же можно использовать и в других проектах.

В каталоге Work/Library находится конфигурационный файл config.ini, в нём можно настроить приложение, поправив некоторые параметры либо оставив их по умолчанию. С ним связаны все модули программы.

# Структура каталогов

* Work – основной каталог
* Data — содержит базу данных
* Graphics – содержит копии графических отчётов и картинки для виджетов приложения
* Library – содержит библиотеку стандартных (универсальных) функций и конфигурационный файл
* Notes – содержит документацию (руководство пользователя и руководство разработчика)
* Output – содержит копии текстовых отчётов
* Scripts – содержит main.py и дополнительные модули

# Листинг скрипта

## Модуль main.py

"""

Цель: Главный модуль, соединяющий остальные модули приложения

"""

import os

import sys

sys.path.insert(0, os.path.abspath('..'))

import tkinter as tk

import tkinter.ttk as ttk

from tkinter import filedialog

from Scripts import constants

from Scripts import base\_handling as hand\_base

from Scripts import globalvars as glob

from Scripts import interface as ui

from Library import error\_edit\_windows as err

def setup() -> tk.Tk:

"""

Автор: Ковязин В.

Цель: создание главного окна и расстановка всех его компонентов

Вход: Нет

Выход: объект главного окна

"""

root = tk.Tk()

style = ttk.Style()

style.configure('Selection.TCheckbutton', anchor="w")

glob.icons = {'save\_icon': tk.PhotoImage(file="../Graphics/save\_icon.gif",

master=root),

'add\_icon': tk.PhotoImage(file="../Graphics/add\_icon.gif",

master=root),

'edit\_icon': tk.PhotoImage(file="../Graphics/edit\_icon.gif",

master=root),

'load\_icon': tk.PhotoImage(file="../Graphics/load\_icon.gif",

master=root),

'close\_icon': tk.PhotoImage(file="../Graphics/close\_icon.gif",

master=root),

'add\_field\_icon': tk.PhotoImage(file="../Graphics/add\_field\_icon.gif",

master=root),

'del\_field\_icon': tk.PhotoImage(file="../Graphics/del\_field\_icon.gif",

master=root)}

root.title('Volcano Analyse')

pane = ttk.PanedWindow(root, orient=tk.HORIZONTAL, width=1)

# создаем и заполняем строчку меню

ui.create\_menu(root, load\_event)

# фрейм кнопочек

ui.create\_toolbar(root, pane, load\_event, save\_event, create\_event, glob.icons)

# лист для баз данных

frame = ui.create\_list4db(root, pane)

pane.add(frame, weight=1)

pls\_select\_frame = ui.show\_invitation(pane)

pane.add(pls\_select\_frame, weight=9)

pane.grid(row=1, column=0, columnspan=3, sticky="NSEW")

root.grid\_rowconfigure(0, weight=1)

root.grid\_rowconfigure(1, weight=99)

root.grid\_rowconfigure(3, weight=1)

root.grid\_columnconfigure(0, weight=1, minsize=150)

root.grid\_columnconfigure(1, weight=99)

return root

def load\_event(\*args):

"""

Автор: Баканов Г.

Цель: обработка события загрузки новой базы инструментами OS

Вход: Нет

Выход: нет

"""

path = filedialog.askopenfilename(initialdir="../Data/",

filetypes=(("Database files", "\*.csv"), ("All files", "\*.\*")))

path = path.replace('/', "\\")

try:

base\_name = hand\_base.read\_base(path)

glob.base\_list.insert(tk.END, base\_name)

except FileNotFoundError:

pass

except Exception as error:

message = str(error)

err.error(message[message.find('['):message.find(']') + 1] + " нет в Базе Данных")

return "break"

def create\_event(\*args):

"""

Автор: Подкопаева П.

Цель: создание новой базы, если база уже существует в рабочей директории

windows обеспечивает выбор, перезаписывать файл или нет.

Вход: Нет

Выход: объект главного окна

"""

new\_base\_path = filedialog.asksaveasfilename(initialdir="../Data/",

filetypes=(("Database files", "\*.csv"),

("All files", "\*.\*")))

new\_base\_name = hand\_base.create\_base(new\_base\_path)

# если имя открытой базы совпадает с именем новой базы

if glob.current\_base\_name == new\_base\_name:

glob.clear\_workspace()

glob.current\_base = glob.work\_list[new\_base\_name]

# если новой базы нет в листе - добавляем

elif new\_base\_name not in glob.base\_list.get(0, tk.END):

glob.base\_list.insert(tk.END, new\_base\_name)

# если имя открытой базы уже есть в листе, то пользователь все равно выбрал перезаписывать файл

# так что нам не нужно расматривать этот вариант и база уже перезаписалась на новую в work\_list

return "break"

def save\_event(\*args):

"""

Автор: Баканов Г.

Цель: Сохранение текущей базы в файл

Вход: Нет

Выход: Нет (файл)

"""

# открыта ли база?

if not glob.is\_db\_open():

return "break"

# сохранена ли база?

if not glob.is\_saved():

glob.unmark\_changes()

glob.work\_list[glob.current\_base\_name] = glob.current\_base

glob.update\_list()

# сохраняем в файл

hand\_base.save\_base()

ui.load\_event = load\_event

ui.create\_event = create\_event

ui.save\_event = save\_event

root = setup()

root.config(background="white")

screen\_width = root.winfo\_screenwidth()

screen\_height = root.winfo\_screenheight()

root.minsize(constants.style['app\_width'], constants.style['app\_height'])

root.maxsize(screen\_width, screen\_height)

root.geometry(str(constants.style['app\_width']) + 'x' + str(constants.style['app\_height']))

root.mainloop()

## Модуль base\_handling.py

"""

Цель: Модуль для обработки базы данных

"""

import pandas as pd

from Scripts import constants

from Scripts import globalvars as glob

# ['Year', 'Month', 'Day', 'Name', 'Location', 'Country', 'Latitude',

# 'Longitude', 'Elevation', 'Type', 'VEI', 'Agent',

# 'DEATHS', 'INJURIES', 'MISSING', 'DAMAGE\_MILLIONS\_DOLLARS', 'TSU', 'EQ']

bd = pd.read\_csv('../Data/volcano.csv', header=0)[['Year', 'Month', 'Day', 'Name', 'Location',

'Country', 'Latitude', 'Longitude', 'Elevation',

'Type', 'VEI', 'Agent',

'DEATHS', 'INJURIES', 'MISSING',

'DAMAGE\_MILLIONS\_DOLLARS', 'TSU', 'EQ']]

glob.work\_list['Volcano Eruption'] = glob.correct\_base\_values(bd)

def read\_base(path: str) -> str:

"""

Автор: Ковязин В.

Цель: загружает базу из файла

Вход: путь

Выход: новая база

"""

# если при создании базы возникло исключение, то перебрасываем исключение дальше

try:

# в прочитанной базе может не оказаться всех нужных нам столбцов

base = pd.read\_csv(path, header=0)[constants.origin\_columns]

except Exception:

raise

base\_name = path[path.rfind('\\') + 1:path.rfind('.')]

i = 0

# если база уже загружена в программу, то в программу добавляется ее копию с постфиксом

if base\_name in glob.work\_list.keys():

while base\_name + "(" + str(i) + ")" in glob.work\_list.keys():

i += 1

base\_name += "(" + str(i) + ")"

base = glob.correct\_base\_values(base)

glob.work\_list[base\_name] = base

return base\_name

def create\_base(path: str) -> str:

"""

Автор: Подкопаева П.

Цель: создает новую чистую базу

Вход: путь

Выход: новая база

"""

if ".csv" in path:

base\_name = path[path.rfind('/') + 1:path.rfind('.')]

else:

base\_name = path[path.rfind('/') + 1:]

path += ".csv"

new\_base = pd.DataFrame(columns=constants.origin\_columns)

new\_base = glob.correct\_base\_values(new\_base)

glob.work\_list[base\_name] = new\_base

new\_base.to\_csv(path, index=False)

return base\_name

def save\_base() -> None:

"""

Автор: Ковязин В.

Цель: сохраняем текущую базу

Вход: нет

Выход: нет

"""

glob.work\_list[glob.current\_base\_name].to\_csv("../Data/" + glob.current\_base\_name + ".csv",

index=False)

def searching(search\_data, filter\_option) -> pd.DataFrame:

"""

Автор: Ковязин В.

Цель: поиск по базе

Вход: запрос и фильтр

Выход: результат поиска

"""

if filter\_option != 'all':

founded = glob.current\_base.loc[glob.current\_base[filter\_option] == search\_data]

return founded

## Модуль constants.py

"""

Цель: Модуль констант приложения

Автор: Баканов Г., Ковязин В.

"""

origin\_columns = ['Year', 'Month', 'Day', 'Name', 'Location', 'Country', 'Latitude', 'Longitude',

'Elevation', 'Type', 'VEI', 'Agent', 'DEATHS', 'INJURIES', 'MISSING', 'DAMAGE\_MILLIONS\_DOLLARS',

'TSU', 'EQ']

"""

origin\_columns содержит название всех столбцов, которые должны быть в программе обязательно,

Если в загружаемой в программу БД нет столбца(ов), то (придумать что будет)

"""

tree\_rows\_number: int = 45

"""количество отображаемых на экране строчек таблицы"""

first\_form = ['Name', 'Latitude', 'Elevation', 'Longitude', 'Type']

second\_form = ['Location', 'Country', 'Latitude', 'Longitude']

third\_form = ['Year', 'Month', 'Day', 'Name', 'VEI', 'Agent', 'DEATHS', 'INJURIES', 'MISSING',

'DAMAGE\_MILLIONS\_DOLLARS', 'TSU', 'EQ']

quality\_columns = ['Name', 'Location', 'Country', 'Latitude', 'Longitude', 'Type', 'VEI', 'Agent', 'TSU', 'EQ']

quantity\_columns = ['Elevation', 'DEATHS', 'INJURIES', 'MISSING', 'DAMAGE\_MILLIONS\_DOLLARS']

with open("../Library/config.ini", 'r', encoding='utf-8') as f:

config = f.read()

style = eval(config)

## Модуль globalvars.py

"""

Цель: Модуль, содержащий основные глобальные переменные программы

Автор: Ковязин В., Баканов Г., Подкопаева П.

"""

from tkinter import BooleanVar

from tkinter import Listbox

from tkinter import PhotoImage

from tkinter.ttk import Treeview

from typing import Dict

from typing import List

import numpy as np

from pandas import DataFrame

from Library import error\_edit\_windows as err

from Scripts import constants

# просто типы данных для удобства исипользования в подсказках

Icons = Dict[str, PhotoImage]

BaseRecord = Dict[str, DataFrame]

SelectionDict = Dict[str, BooleanVar]

Columns = List[str]

columns: Columns = constants.origin\_columns

"""

В columns будут храниться все столбцы, отображаемые в данный момент в программе.

Программа будет автоматически отображать новые столбцы и убирать удаленные.

"""

work\_list: BaseRecord = {}

"""

Словарь со значениями (Имя БД: объект БД).

В нем хранятся все базы данных, которые были загруженны в программу.

При сохранении будет сохраняться именно базы из словаря.

"""

current\_base: DataFrame = None

"""Текущий dataframe, с которым работает пользователь."""

current\_base\_name: str = None

"""Имя текущего dataframe, с которым работает пользователь."""

current\_base\_list\_id: int = None

"""id текущего dataframe в ListBox, с которым работает пользователь."""

table4base: Treeview = None

"""

Объект TreeView, данные из которого отображаются в workspace.

В TreeView хранятся все данные из выбранной базы данных.

"""

base\_list: Listbox = None

"""

Объект Listbox, который отображается в программе.

в нем хранятся все имена загруженных баз

с помощью него осуществляется выбор пользователем

"""

columns\_selection: SelectionDict = None

"""

Словарь для выбора столбцов, которые должны отображаться в программе,\n

{Столбец: Значение}, где значение отвечает, будет ли отображаться столбец

"""

icons: Icons = {}

"""для иконок, нужна для того,

чтобы garbage collector не съедал их из-за отсутствия ссылки на них"""

sort = True

"""переключатель для сортировки таблицы"""

def is\_saved() -> bool:

"""

Автор: Ковязин В.

Цель: проверяет, сохранена ли база

Вход: нет

Выход: true, false

"""

global current\_base\_name

if "\*" in current\_base\_name:

return False

return True

def is\_db\_open() -> bool:

"""

Автор: Ковязин В.

Цель: проверка, окрыта ли база

Вход: Нет

Выход: true, false

"""

if current\_base is None:

err.error("База не выбранна!")

return False

return True

def delete\_current\_base():

"""

Автор: Подкопаева П.

Цель: удаляет базу из программы

Вход: нет

Выход: нет

"""

global work\_list, current\_base\_name, current\_base\_list\_id, current\_base

del work\_list[current\_base\_name]

current\_base = None

current\_base\_name = None

base\_list.delete(current\_base\_list\_id)

current\_base\_list\_id = None

def mark\_changes():

"""

Автор: Подкопаева П.

Цель: убирает пометку в имени текущей базы наличие несохраненных изменений

Вход: нет

Выход: нет

"""

global current\_base\_name

if is\_saved():

current\_base\_name += "\*"

def unmark\_changes():

"""

Автор: Подкопаева П.

Цель: помечает в имени текущей базы наличие несохраненных изменений

Вход: нет

Выход: нет

"""

global current\_base\_name

if not is\_saved():

current\_base\_name = current\_base\_name.replace('\*', '')

def correct\_base\_values(base: DataFrame) -> DataFrame:

"""

Автор: Ковязин В.

Цель: при добавлении в таблицу измененных пользователем данных могут возникнуть nan значения,

их мы меняем на пустые строки или на 0, так же nan меняет типы столбцов на другой,

здесь мы обратно приводим тип столбцов к нужному

Вход: нет

Выход: нет

"""

base = base.replace('', np.nan)

base[['Year', 'Month', 'Day']] = base[['Year', 'Month', 'Day']].replace(np.nan, 0)

base[['Name', 'Location', 'Country', 'Type', 'Agent']] = base[

['Name', 'Location', 'Country', 'Type', 'Agent']].replace(np.nan, " ")

base[['TSU']] = base[['TSU']].replace('TSU', '+')

base[['EQ']] = base[['EQ']].replace('EQ', '+')

base[['TSU']] = base[['TSU']].replace(np.nan, '-')

base[['EQ']] = base[['EQ']].replace(np.nan, '-')

base = base.astype({'Year': 'int32', 'Month': 'int32', 'Day': 'int32',

'Latitude': 'float64', 'Longitude': 'float64', 'VEI': 'float64',

'DEATHS': 'float64', 'INJURIES': 'float64', 'MISSING': 'float64',

'DAMAGE\_MILLIONS\_DOLLARS': 'float64'})

base[['Name', 'Location', 'Country',

'Type', 'Agent', 'TSU',

'EQ']] = base[['Name', 'Location', 'Country',

'Type', 'Agent', 'TSU',

'EQ']].astype(str)

base = base.astype({'Name': 'str', 'Location': 'str', 'Country': 'str',

'Type': 'str', 'Agent': 'str', 'TSU': 'str',

'EQ': 'str'})

return base

def update\_workspace():

"""

Автор: Баканов Г.

Цель: обновляет содержимое рабочего пространства

Вход: нет

Выход: нет

"""

global current\_base

global columns

assert current\_base is not None

for i in list(current\_base.index):

insert = current\_base.iloc[i, :]

for j in columns:

table4base.set(i, column=j, value=insert[j])

def clear\_workspace():

"""

Автор: Баканов Г.

Цель: очищает рабочее пространство

Вход: нет

Выход: нет

"""

global current\_base

global columns

assert current\_base is not None

table4base.delete(\*list(range(len(current\_base.index))))

def update\_list():

"""

Автор: Баканов Г.

Цель: заново добавляет текущую базу в base\_list

Вход: нет

Выход: нет

"""

global current\_base

global current\_base\_list\_id

base\_list.delete(current\_base\_list\_id)

base\_list.insert(current\_base\_list\_id, current\_base\_name)

## Модуль interface.py

"""

Цель: Модуль, отвечающий за интерфейс приложения

"""

import tkinter as tk

import tkinter.ttk as ttk

from tkinter import messagebox as mb, filedialog

import pandas as pd

from Library import error\_edit\_windows as err

from Scripts import base\_handling as hand\_base

from Scripts import constants

from Scripts import globalvars as glob

from Scripts import new\_map as mp

from Scripts import new\_table as tb

from Scripts import statistics as stat

# events ---------------------------------------------------------------------------------------

def close\_event(pane: ttk.Panedwindow, save):

"""

Автор: Подкопаева П.

Цель: закрывает открытую базу и показывает приглашение

к открытию новой на правой стороне pane,

save вызывается для сохранения базы, по решению пользователя

Вход: pane - растягивающийся виджет, save - объект функции save\_event из main

Выход: нет

"""

# открыта ли база?

if not glob.is\_db\_open():

return "break"

# сохранена ли база?

if not glob.is\_saved():

ans = err.yes\_no("Сохранить изменения?")

if ans:

save()

glob.delete\_current\_base()

pane.forget(1)

pls\_select\_frame = show\_invitation(pane)

pane.add(pls\_select\_frame, weight=9)

def remove\_inf():

"""

Автор: Подкопаева П.

Цель: удаляет строку из таблицы

Вход: корневое окно tkinter для создания окна редактирования, список активных столбцов таблицы

Выход: нет

"""

# открыта ли база?

if not glob.is\_db\_open():

return "break"

# пуста ли база?

if glob.current\_base.empty:

err.error("База пуста")

return "break"

ans = err.yes\_no("Вы точно хотите удалить данные?\n Это повлечёт полное удаление данных по выбранному вулкану.")

if ans:

index = glob.table4base.index(glob.table4base.selection())

glob.table4base.delete(list(glob.current\_base.index)[-1])

glob.current\_base = glob.current\_base.drop(index=index)

glob.current\_base.reset\_index(inplace=True, drop=True)

glob.mark\_changes()

glob.update\_workspace()

glob.update\_list()

def edit\_event(root: tk.Tk):

"""

Автор: Ковязин В.

Цель: обработчик события кнопки изменения поля таблицы, открывает окно для изменения данных

Вход: корневое окно tkinter для создания окна редактирования

Выход: нет

"""

# открыта ли база?

if not glob.is\_db\_open():

return "break"

# пуста ли база?

if glob.current\_base.empty:

err.error("База пуста")

return "break"

# получаем изменяемую строчку

index = glob.table4base.index(glob.table4base.selection())

curr\_item = glob.current\_base.iloc[index, :]

# создаем дочернее окно

edit\_win = tk.Toplevel(root)

edit\_win.resizable(0, 0)

edit\_win.title("Изменения данных поля таблицы")

# распологаем все необходимые элементы в этих фреймах

frame4check4labels = tk.Frame(edit\_win)

frame4check4entries = tk.Frame(edit\_win)

frame4check4button = tk.Frame(edit\_win)

list4changes = {}

for i in constants.origin\_columns:

# все значения будут строкой

text = tk.StringVar()

# если атрибут nan или 0, то вместо него отображаем пустую строчку

if pd.isna(curr\_item[i]) or (i in ['Year', 'Month', 'Day'] and curr\_item[i] == 0):

text.set("")

else:

text.set(curr\_item[i])

list4changes[i] = text

label = tk.Label(frame4check4labels, text=i + ":", anchor="e")

entry = tk.Entry(frame4check4entries, textvariable=text)

if i not in glob.columns:

label.configure(state=tk.DISABLED)

entry.configure(state='readonly')

entry.pack(side="top", fill="x", expand=True, pady=5)

label.pack(side="top", fill="x", expand=True, pady=5)

save\_changes\_button = tk.Button(frame4check4button, text="Сохранить")

save\_changes\_button.pack(expand=False)

save\_changes\_button.bind("<Button-1>",

lambda \*args: make\_changes\_event(edit\_win, index, list4changes))

edit\_win.rowconfigure(0, pad=5)

edit\_win.rowconfigure(1, pad=5)

edit\_win.columnconfigure(0, pad=5)

edit\_win.columnconfigure(1, pad=5)

frame4check4labels.grid(row=0, column=0, sticky="NSW")

frame4check4entries.grid(row=0, column=1, sticky="NSW")

frame4check4button.grid(row=1, column=0, columnspan=2, sticky="NSEW")

def make\_changes\_event(win: tk.Toplevel, index: int, new\_values: dict):

"""

Автор: Баканов Г.

Цель: обработчик события кнопки сохранения в окне редактирования поля таблицы

Вход: объект окна редактирования tkinter для его закрытия после нажатия кнопки сохранить,

текущий индекс выбранного поля таблицы,

новые значения для записи в поле таблицы.

Выход: нет

"""

# приводим все числа к числовому типу

glob.current\_base.iloc[index, :] = [x.get() for x in new\_values.values()]

# заменяем пустые строчки на nan и приводим тип всех столбцов таблицы к нужному типу

glob.current\_base = glob.correct\_base\_values(glob.current\_base)

glob.work\_list[glob.current\_base\_name] = glob.current\_base

item = glob.table4base.selection()

for key, value in new\_values.items():

glob.table4base.set(item, column=key, value=value.get())

glob.mark\_changes()

glob.update\_list()

glob.update\_workspace()

win.destroy()

def uncheck\_all\_event(\*args):

"""

Автор: Баканов Г.

Цель: снимает метки со всех значений columns\_selection

Вход: нет

Выход: нет

"""

[x.set(0) for x in glob.columns\_selection.values()]

def check\_all\_event(\*args):

"""

Автор: Баканов Г.

Цель: ставит метки на все значения columns\_selection

Вход: нет

Выход: нет

"""

[x.set(1) for x in glob.columns\_selection.values()]

def apply\_column\_selection(root: tk.Tk, win: tk.Toplevel, pane: ttk.Panedwindow):

"""

Автор: Баканов Г.

Цель: применяет к программме выбор столбцов (изменяет рабочее пространство)

Вход: главное окно, побочное окно выбора столбцов, растягивающийся виджет

Выход: нет

"""

if any(glob.columns\_selection.values()):

glob.columns = [x for x in glob.columns\_selection.keys() if glob.columns\_selection[x].get() == 1]

open\_base(root, pane, glob.current\_base\_list\_id)

win.destroy()

else:

err.error("Не выбран ни один столбец")

return "break"

def select\_columns\_event(root: tk.Tk, pane: ttk.Panedwindow):

"""

Автор: Ковязин В.

Цель: открывает окно для выбора столбцов, которые надо показать в программе

Вход: главное окно, растягивающийся виджет

Выход: нет

"""

# открыта ли база?

if not glob.is\_db\_open():

return "break"

win = tk.Toplevel(root)

win.title("Выберите стобцы")

glob.columns\_selection = {k: v

for k in constants.origin\_columns

for v in [tk.BooleanVar() for x in range(len(glob.constants.origin\_columns))]

}

frame4check = tk.Frame(win)

frame4button = tk.Frame(win)

i = 0

# раставляем checkbutton'ы и устанавливаем их в активное (отмеченное)

# положение по текущим показывающимся столбцам

for text, value in glob.columns\_selection.items():

ttk.Checkbutton(frame4check,

style="Selection.TCheckbutton", text=text, variable=value, onvalue=True,

offvalue=False).grid(row=i, column=1, sticky='NSEW')

value.set(True) if text in glob.columns else value.set(False)

i += 1

apply\_button = tk.Button(frame4button, text="Применить")

uncheck\_all\_button = tk.Button(frame4button, text="Снять выбор")

check\_all\_button = tk.Button(frame4button, text="Выбрать все")

apply\_button.bind("<Button-1>", lambda \*args: apply\_column\_selection(root, win, pane))

uncheck\_all\_button.bind("<Button-1>", uncheck\_all\_event)

check\_all\_button.bind("<Button-1>", check\_all\_event)

uncheck\_all\_button.grid(row=0, column=0, sticky='NSE', padx=5, pady=2)

check\_all\_button.grid(row=0, column=1, sticky='NSW', padx=5, pady=2)

apply\_button.grid(row=1, column=0, columnspan=2, sticky='NS', pady=2)

frame4check.pack(side="top", fill="both", expand=True, padx=10, pady=5)

frame4button.pack(side="top", fill="both", expand=True, padx=10, pady=5)

def open\_base(root: tk.Tk, pane: ttk.Panedwindow, selected: int):

"""

Автор: Баканов Г.

Цель: открывает загруженную базу данных и

создает для нее таблицу с полями, добавляя ее на главный экран

Вход: объект главного окна,

объект растягиваемого виджета,

индекс базы в Listbox

Выход: нет

"""

glob.current\_base\_list\_id = selected

glob.current\_base, glob.current\_base\_name = glob.work\_list.get(

glob.base\_list.get(selected).replace('\*', '')), glob.base\_list.get(selected)

work\_frame4check = create\_workspace(root, pane)

pane.forget(1)

pane.add(work\_frame4check, weight=10000)

def workspace\_onclick\_event(root, event, mode: str):

"""

Автор: Баканов Г.

Цель: обработчик события нажатия на рабочее пространство таблицы данных

Вход: объект главного окна,

информация события,

вид нажатия (одинарное, двойное)

Выход: нет

"""

glob.sort = not glob.sort

tree = glob.table4base

# одиночное нажатие по заголовку - сортировка

if mode == "Single":

if tree.identify\_region(event.x, event.y) == "heading":

column = tree.identify\_column(event.x)

index4column = int(column[1:])

glob.current\_base = glob.current\_base.sort\_values(by=glob.columns[index4column - 1],

axis=0,

ascending=glob.sort, ignore\_index=True)

glob.current\_base = glob.correct\_base\_values(glob.current\_base)

glob.update\_workspace()

# двойное нажатие по полю - редактирование

elif mode == "Double":

edit\_event(root)

def show\_invitation(pane: ttk.Panedwindow) -> tk.Frame:

"""

Автор: Ковязин В.

Цель: создание фрейма с приглашением

Вход: объект растягивающегося виджета

Выход: фрейм с приглашением

"""

# label приглашение к выбору

pls\_select\_frame4check = tk.Frame(pane, bg=constants.style['bg'])

lbl\_select\_pls = tk.Label(pls\_select\_frame4check,

text="Пожалуйста, выберите базу данных", bg=constants.style['bg'])

lbl\_select\_pls.pack(expand=True, fill="both")

return pls\_select\_frame4check

def show\_form(root, pane, selector, form: str, save):

"""

Автор: Подкопаева П.

Цель: Выбор вида таблицы

Вход: Нет

Выход: нет

"""

if not glob.is\_db\_open():

selector.current(0)

return

if not glob.is\_saved():

ans = err.yes\_no("Сохранить изменения?")

if ans:

save()

if form == "Общий вид":

glob.columns = constants.origin\_columns

elif form == "Вид первый":

glob.columns = constants.first\_form

elif form == "Вид второй":

glob.columns = constants.second\_form

elif form == "Вид третий":

glob.columns = constants.third\_form

open\_base(root, pane, glob.current\_base\_list\_id)

def select\_statistics\_event(root: tk.Tk, pane: ttk.Panedwindow):

"""

Автор: Подкопаева П., Ковязин В.

Цель: открывает окно для выбора данных, для которых нужно показать общую статистику

Вход: главное окно, растягивающийся виджет

Выход: нет

"""

# открыта ли база?

if not glob.is\_db\_open():

return "break"

win = tk.Toplevel(root)

win.title("Выбор")

win.geometry(constants.style['popup\_width'] + 'x' + constants.style['popup\_height'] + "+500+200")

win.minsize(400, 600)

background = tk.Frame(win, bg=constants.style['bg'])

background.place(x=0, y=0, relwidth=1, relheight=1)

lang = tk.StringVar()

win.geometry("400x600+500+300")

lbl = tk.Label(win)

lbl.configure(bg=constants.style['bg'],

text="Выберите данные для общей статистики")

lbl.place(relx=0.05, rely=0.02)

Elevation\_checkbutton = tk.Radiobutton(background,

text="Высота", value="Elevation",

bg=constants.style['bg'],

variable=lang)

Elevation\_checkbutton.place(relx=0.25, rely=0.05)

DEATHS\_checkbutton = tk.Radiobutton(background,

text="Количество смертей",

value="DEATHS",

variable=lang,

bg=constants.style['bg'])

DEATHS\_checkbutton.place(relx=0.25, rely=0.1)

DAMAGE\_checkbutton = tk.Radiobutton(background,

text="Ущерб в млн долларов",

value="DAMAGE\_MILLIONS\_DOLLARS",

variable=lang,

bg=constants.style['bg'])

DAMAGE\_checkbutton.place(relx=0.25, rely=0.15)

MISSING\_checkbutton = tk.Radiobutton(background,

text="Количество пропавших",

value="MISSING",

variable=lang,

bg=constants.style['bg'])

MISSING\_checkbutton.place(relx=0.25, rely=0.2)

INJURIES\_checkbutton = tk.Radiobutton(background,

text="Количество раненных",

value="INJURIES",

variable=lang,

bg=constants.style['bg'])

INJURIES\_checkbutton.place(relx=0.25, rely=0.25)

ttk.Separator(background, orient='horizontal').place(relx=0, rely=0.29,

relheight=0,

relwidth=1)

Name\_checkbutton = tk.Radiobutton(background,

text="Имя", value="Name",

bg=constants.style['bg'], variable=lang)

Name\_checkbutton.place(relx=0.25, rely=0.3)

Location\_checkbutton = tk.Radiobutton(background, text="Расположение",

value="Location",

variable=lang,

bg=constants.style['bg'])

Location\_checkbutton.place(relx=0.25, rely=0.35)

Country\_checkbutton = tk.Radiobutton(background,

text="Страна", value="Country",

variable=lang,

bg=constants.style['bg'])

Country\_checkbutton.place(relx=0.25, rely=0.4)

Latitude\_checkbutton = tk.Radiobutton(background,

text="Широта", value="Latitude",

variable=lang,

bg=constants.style['bg'])

Latitude\_checkbutton.place(relx=0.25, rely=0.45)

Longitude\_checkbutton = tk.Radiobutton(background,

text="Долгота", value="Longitude",

variable=lang,

bg=constants.style['bg'])

Longitude\_checkbutton.place(relx=0.25, rely=0.5)

Type\_checkbutton = tk.Radiobutton(background,

text="Тип", value="Type",

bg=constants.style['bg'], variable=lang)

Type\_checkbutton.place(relx=0.25, rely=0.55)

VEI\_checkbutton = tk.Radiobutton(background,

text="Индекс взрывоопасности",

value="VEI", variable=lang,

bg=constants.style['bg'])

VEI\_checkbutton.place(relx=0.25, rely=0.6)

Agent\_checkbutton = tk.Radiobutton(background, text="Причина", value="Agent",

variable=lang, bg=constants.style['bg'])

Agent\_checkbutton.place(relx=0.25, rely=0.65)

TSU\_checkbutton = tk.Radiobutton(background, text="Было ли цунами?", value="TSU",

variable=lang, bg=constants.style['bg'])

TSU\_checkbutton.place(relx=0.25, rely=0.7)

EQ\_checkbutton = tk.Radiobutton(background,

text="Было ли землетрясение?", value="EQ",

variable=lang, bg=constants.style['bg'])

EQ\_checkbutton.place(relx=0.25, rely=0.75)

apply\_button = tk.Button(background, text="Выбрать",

font=3, bg=constants.style['apply\_button'])

apply\_button.bind("<Button-1>",

lambda \*args:

stat.statistics\_base(root,

pane,

lang.get()) if lang.get() in constants.quantity\_columns

else show\_stat\_report(root, lang.get()))

apply\_button.place(relx=0.5, rely=0.8, relheight=0.1, relwidth=0.25)

background.pack(side="top", fill="both", expand=True, padx=10, pady=5)

def analysis\_window\_event(root: tk.Tk, pane: ttk.Panedwindow):

"""

Автор: Ковязин В.

Цель: Окно выбора метода анализа

Вход: главное окно, растягивающийся виджет

Выход: Нет

"""

if not glob.is\_db\_open():

return "break"

win = tk.Toplevel(root)

win.title("Выбор")

win.geometry("400x400+500+200")

background = tk.Frame(win, bg="#F8F8FF")

background.place(x=0, y=0, relwidth=1, relheight=1)

backgroundlabel = tk.Label(background, bg="#F8F8FF")

backgroundlabel.place(relx=0.025, rely=0.025, relwidth=0.95, relheight=0.95)

button\_statistics = tk.Button(background, text='Сводная таблица', width=20,

height=2, font=11, bg=constants.style['pivot\_button'])

button\_statistics.bind("<Button-1>", lambda \*args: tb.choice\_table(root, pane))

button\_statistics.place(relx=0.25, rely=0.15, relheight=0.1, relwidth=0.5)

button\_map = tk.Button(background, bg=constants.style['map\_button'],

text='Построение карты', font=11, width=20, height=2)

button\_map.bind("<Button-1>", lambda \*args: mp.choice\_map(root, pane))

button\_map.place(relx=0.25, rely=0.45, relheight=0.1, relwidth=0.5)

button\_graphics = tk.Button(background, bg=constants.style['plotsndiagrams\_button'],

text='Построение диаграмм\n и графиков', font=11,

width=20, height=2)

button\_graphics.bind("<Button-1>", lambda \*args: stat.graphics\_choice(root, pane))

button\_graphics.place(relx=0.2, rely=0.75, relheight=0.2, relwidth=0.65)

background.pack(side="top", fill="both", expand=True, padx=10, pady=5)

def save\_report\_event(table, columns, i):

"""

Автор: Ковязин В.

Цель: Сохранение в файл таблицы

Вход: Таблица с данными

Выход: Файл

"""

data = pd.DataFrame(columns=columns)

for index in range(i):

data = data.append(table.set(index), ignore\_index=True)

path = filedialog.asksaveasfilename(initialdir="../Data/",

filetypes=(("Database files", "\*.csv"),

("All files", "\*.\*")))

if ".csv" not in path:

path += ".csv"

data.to\_csv(path, index=False, encoding='cp1251')

return "break"

def show\_stat\_report(root: tk.Tk, target: str):

"""

Автор: Ковязин В.

Цель: Формирование статистики по качественным переменным

Вход: Выбранный параметр типа строка, главное окно

Выход: Нет

"""

win = tk.Toplevel(root)

win.title(target)

if target not in ['TSU', 'EQ']:

columns = ["Атрибут " + target, "Частота", "Процент от общего числа"]

else:

columns = ["Было ли " + ('цунами' if target == 'TSU' else 'землетрясение'),

"Частота", "Процент от общего числа"]

table = ttk.Treeview(win, height=20, show='headings', columns=columns)

[table.heading('#' + str(x + 1), text=columns[x]) for x in range(3)]

freq, whole = stat.stat\_report(root, target)

i = 0

for name, number in freq.items():

table.insert('', 'end', iid=i,

values=[name, number, str(round((number / whole) \* 100, 2)) + '%'])

i += 1

vsb = ttk.Scrollbar(win, orient="vertical", command=table.yview)

table.configure(yscrollcommand=vsb.set)

save\_button = tk.Button(win, text='Сохранить')

save\_button.bind("<Button-1>", lambda \*args: save\_report\_event(table, columns, i))

vsb.pack(side='right', fill='both')

save\_button.pack(side='top')

table.pack(side='top', fill='x')

def stat\_report\_event(root: tk.Tk):

"""

Автор: Ковязин В.

Цель: Показ статистики по качественным переменным

Вход: главное окно

Выход: нет

"""

if not glob.is\_db\_open():

return "break"

win = tk.Toplevel(root)

win.title("Статистический отчет")

win.geometry('300x300')

target = tk.StringVar()

dropdown = ttk.Combobox(win, values=constants.quality\_columns, state='readonly',

textvariable=target)

dropdown.current(0)

show\_result = tk.Button(win, text="Показать")

show\_result.bind("<Button-1>", lambda \*args: show\_stat\_report(root, target.get()))

tk.Label(win, text="Атрибут").pack(side="top")

dropdown.pack(side="top")

show\_result.pack(side="top")

# ---------------------------------------------------------------------------------------

# frame4checks ===================================================================================

def create\_toolbar(root: tk.Tk, pane: ttk.Panedwindow, load, save, create, icons: glob.Icons):

"""

Автор: Подкопаева П., Баканов Г.

Цель: создание панели инструментов в главном окне

Вход: объект главного окна,

объект растягивающегося виджета,

объекты функций load\_event, save\_event, create\_event

словарь для иконок

Выход: нет

"""

tools\_frame4check = tk.Frame(root, bg=constants.style['bg\_accent'])

add\_button = tk.Button(tools\_frame4check, image=icons['add\_icon'], relief="groove", bd=0,

bg=constants.style['bg\_accent'])

save\_button = tk.Button(tools\_frame4check, image=icons['save\_icon'], relief="groove", bd=0,

bg=constants.style['bg\_accent'])

edit\_button = tk.Button(tools\_frame4check, image=icons['edit\_icon'], relief="groove", bd=0,

bg=constants.style['bg\_accent'])

load\_button = tk.Button(tools\_frame4check, image=icons['load\_icon'], relief="groove", bd=0,

bg=constants.style['bg\_accent'])

add\_field\_button = tk.Button(tools\_frame4check,

image=icons['add\_field\_icon'], relief="groove", bd=0,

bg=constants.style['bg\_accent'])

del\_field\_button = tk.Button(tools\_frame4check, image=icons['del\_field\_icon'],

relief="groove", bd=0,

bg=constants.style['bg\_accent'])

select\_columns = tk.Button(tools\_frame4check, text="Столбцы", relief="raised", bd=2,

bg=constants.style['bg\_accent'])

close\_button = tk.Button(tools\_frame4check, image=icons['close\_icon'], relief="groove", bd=0,

bg=constants.style['bg\_accent'])

table\_normal\_forms\_selector = ttk.Combobox(tools\_frame4check, state='readonly',

values=["Общий вид",

"Вид первый", "Вид второй", "Вид третий"])

statistics\_select = tk.Button(tools\_frame4check,

text="Общая статистика за период наблюдений", relief="raised",

bd=2, bg=constants.style['bg\_accent'])

analysis\_window = tk.Button(tools\_frame4check, text="Анализ данных", relief="raised", bd=2,

bg=constants.style['analyse\_button'])

global CHOSEN\_option, GL\_REQUEST

search\_data = tk.StringVar(value="Поиск")

request = tk.Entry(tools\_frame4check, textvariable=search\_data, relief="raised", bd=2)

request\_menu = ttk.Combobox(tools\_frame4check, state='readonly',

values=["Без фильтра",

"По названию", "По типу", "По стране", "По цунами",

"По землетрясению"])

button\_save\_filter = tk.Button(tools\_frame4check,

relief="groove", bd=0, bg=constants.style['bg\_accent'])

analysis\_window.bind("<Button-1>", lambda \*args: analysis\_window\_event(root, pane))

statistics\_select.bind("<Button-1>", lambda \*args: select\_statistics\_event(root, pane))

table\_normal\_forms\_selector.current(0)

request\_menu.current(0)

add\_button.bind("<Button-1>", create)

save\_button.bind("<Button-1>", save)

edit\_button.bind("<Button-1>", lambda \*args: edit\_event(root))

load\_button.bind("<Button-1>", load)

add\_field\_button.bind("<Button-1>",

lambda \*args: add\_inf(root, table\_normal\_forms\_selector.get(), save))

del\_field\_button.bind("<Button-1>", lambda \*args: remove\_inf())

table\_normal\_forms\_selector.bind("<<ComboboxSelected>>",

lambda event: show\_form(root, pane,

table\_normal\_forms\_selector,

table\_normal\_forms\_selector.get(),

save))

request\_menu.bind("<<ComboboxSelected>>",

lambda event: search\_call(root, pane, request\_menu,

request\_menu.get(), search\_data.get(),

save, create))

select\_columns.bind("<Button-1>", lambda \*args: select\_columns\_event(root, pane))

close\_button.bind("<Button-1>", lambda \*args: close\_event(pane, save))

# button\_save\_filter.bind("<Button-1>", lambda \*args: )

add\_button.grid(row=0, column=0, padx=2, pady=2, sticky="NSEW")

load\_button.grid(row=0, column=1, padx=2, pady=2, sticky="NSEW")

save\_button.grid(row=0, column=2, padx=2, pady=2, sticky="NSEW")

edit\_button.grid(row=0, column=3, padx=2, pady=2, sticky="NSEW")

add\_field\_button.grid(row=0, column=4, padx=2, pady=2, sticky="NSEW")

del\_field\_button.grid(row=0, column=5, padx=2, pady=2, sticky="NSEW")

table\_normal\_forms\_selector.grid(row=0, column=6, padx=2, pady=2, sticky="NSEW")

select\_columns.grid(row=0, column=7, padx=2, pady=2, sticky="NSEW")

close\_button.grid(row=0, column=8, padx=2, pady=2, sticky="NSEW")

button\_save\_filter.grid(row=0, column=18, padx=2, pady=2, sticky="NSEW")

statistics\_select.grid(row=0, column=16, padx=2, pady=2, sticky="NSEW")

analysis\_window.grid(row=2, column=16, padx=2, pady=2, sticky="NSEW")

request.grid(row=0, column=17, padx=2, pady=2, sticky="NSEW")

request\_menu.grid(row=2, column=17, padx=2, pady=2, sticky="NSEW")

tools\_frame4check.grid\_rowconfigure(0, minsize=20)

tools\_frame4check.grid(row=0, column=0, columnspan=12, sticky="NSEW")

def search\_call(root: tk.Tk, pane: ttk.Panedwindow, selector, option: str,

search\_data, save, create):

"""

Автор: Подкопаева П.

Цель: Поиск в базе данных по ключевому слову

Вход: Ключевое слово типа строка

Выход: dataframe

"""

if not glob.is\_db\_open():

selector.current(0)

return

if not glob.is\_saved():

ans = err.yes\_no("Сохранить изменения?")

if ans:

save()

filter\_option = ''

if option == "Без фильтра":

filter\_option = 'all'

elif option == "По названию":

filter\_option = 'Name'

elif option == "По типу":

filter\_option = 'Type'

elif option == "По стране":

filter\_option = 'Country'

elif option == "По цунами":

filter\_option = 'TSU'

elif option == "По землетрясению":

filter\_option = 'EQ'

search\_result = hand\_base.searching(search\_data, filter\_option)

if search\_result.empty:

err.warning('Ничего не найдено', True)

return

search\_result.reset\_index(inplace=True, drop=True)

create()

glob.work\_list[glob.base\_list.get(glob.base\_list.size() - 1)] = search\_result

open\_base(root, pane, glob.base\_list.size() - 1)

glob.mark\_changes()

save()

def create\_list4db(root: tk.Tk, pane: ttk.Panedwindow) -> tk.LabelFrame:

"""

Автор: Баканов Г.

Цель: создание виджета Listbox для выбора базы

Вход: объект главного окна,

объект растягивающегося виджета

Выход: фрейм с Listbox

"""

list\_frame4check = tk.LabelFrame(root, labelanchor='n',

text='Базы данных', bd=0, padx=5, pady=5, relief=tk.RIDGE,

bg='white')

lsb\_base = tk.Listbox(list\_frame4check, selectmode='browse')

for name, base in glob.work\_list.items():

lsb\_base.insert(tk.END, name)

glob.base\_list = lsb\_base

lsb\_base.bind('<Double-Button-1>',

lambda \*args: open\_base(root, pane, lsb\_base.curselection()))

lsb\_base.pack(side="left", fill="both", expand=True)

return list\_frame4check

def create\_menu(root: tk.Tk, load):

"""

Автор: Ковязин В.

Цель: создает меню на главном окне

Вход: объект главного окна, объект функции load

Выход: нет

"""

menubar = tk.Menu(root)

file = tk.Menu(menubar, tearoff=0)

file.add\_command(label="Load", command=load)

file.add\_command(label="Exit", command=root.destroy)

menubar.add\_cascade(label="File", menu=file)

root.config(menu=menubar)

def create\_workspace(root: tk.Tk, pane: ttk.Panedwindow) -> tk.LabelFrame:

"""

Автор: Ковязин В.

Цель: создает рабочее пространство таблицы

Вход: объект главного окна,

объект растягивающегося виджета,

Выход: фрейм с таблицей

"""

# создаем и заполняем нашу таблицу

title = glob.columns

frame = tk.LabelFrame(pane, labelanchor='n',

text='Данные', bd=0, pady=5, padx=5, relief=tk.RIDGE, bg='white')

tree = ttk.Treeview(frame, columns=title,

height=constants.tree\_rows\_number, show="headings", selectmode='browse')

[tree.heading('#' + str(x + 1), text=title[x]) for x in range(len(title))]

for i in list(glob.current\_base.index):

insert = list(glob.current\_base[glob.columns].iloc[i, :])

tree.insert('', 'end', iid=i, values=insert)

# меняем ширину столбца для красоты

for i in range(1, len(title) + 1):

tree.column('#' + str(i), width=100, stretch=False)

# скроллбары для нее

vsb = ttk.Scrollbar(frame, orient="vertical", command=tree.yview)

hsb = ttk.Scrollbar(frame, orient="horizontal", command=tree.xview)

tree.bind("<Button-1>", lambda event, mode="Single": workspace\_onclick\_event(root, event, mode))

tree.bind("<Double-Button-1>", lambda event,

mode="Double": workspace\_onclick\_event(root, event, mode))

tree.configure(yscrollcommand=vsb.set)

tree.configure(xscrollcommand=hsb.set)

# пакуем все в фрейм, а его по сетке в окно

glob.table4base = tree

hsb.pack(side='bottom', fill='both')

vsb.pack(side='right', fill='both')

tree.pack(side='top', fill='x')

return frame

# =======================================================================================

def add\_inf(win: tk.Tk, form: str, save):

"""

Автор: Подкопаева П., Баканов Г.

Цель: Добавление новых элементов в базу данных (окно)

Вход: Нет

Выход: Нет

"""

if not glob.is\_db\_open():

return "break"

root = tk.Toplevel(win)

root.title("Окно ввода данных")

if form == "Общий вид":

Year = tk.IntVar()

Year\_label = tk.Label(root, text="Год извержения:")

Year\_label.grid(row=0, column=0, sticky="w")

Year\_entry = tk.Entry(root, textvariable=Year)

Year\_entry.grid(row=0, column=1, padx=5, pady=5)

cmb\_month = ttk.Combobox(root)

Month\_label = tk.Label(root, text="Месяц извержения:")

Month\_label.grid(row=1, column=0, sticky="w", padx=5, pady=5)

cmb\_month['values'] = ('1', '2', '3', '4', '5', '6', '7',

'8', '9', '10', '11', '12')

cmb\_month.current(0)

cmb\_month.grid(column=1, row=1)

Day = tk.IntVar()

Day\_label = tk.Label(root, text="День извержения:")

Day\_label.grid(row=2, column=0, sticky="w")

Day\_entry = tk.Entry(root, textvariable=Day)

Day\_entry.grid(row=2, column=1, padx=5, pady=5)

name = tk.StringVar()

name\_label = tk.Label(root, text="Название вулкана:")

name\_label.grid(row=3, column=0, sticky="w")

name\_entry = tk.Entry(root, textvariable=name)

name\_entry.grid(row=3, column=1, padx=5, pady=5)

Type = tk.StringVar()

Type\_label = tk.Label(root, text="Тип вулкана:")

Type\_label.grid(row=4, column=0, sticky="w")

Type\_entry = tk.Entry(root, textvariable=Type)

Type\_entry.grid(row=4, column=1, padx=5, pady=5)

Height = tk.IntVar()

Height\_label = tk.Label(root, text="Высота вулкана (в метрах):")

Height\_label.grid(row=5, column=0, sticky="w")

Height\_entry = tk.Entry(root, textvariable=Height)

Height\_entry.grid(row=5, column=1, padx=5, pady=5)

country = tk.StringVar()

country\_label = tk.Label(root, text="Страна:")

country\_label.grid(row=6, column=0, sticky="w")

country\_entry = tk.Entry(root, textvariable=country)

country\_entry.grid(row=6, column=1, padx=5, pady=5)

location = tk.StringVar()

location\_label = tk.Label(root, text="Расположение вулкана:")

location\_label.grid(row=7, column=0, sticky="w")

location\_entry = tk.Entry(root, textvariable=location)

location\_entry.grid(row=7, column=1, padx=5, pady=5)

Latitude = tk.IntVar()

Latitude\_label = tk.Label(root, text="Широта:")

Latitude\_label.grid(row=8, column=0, sticky="w")

Latitude\_entry = tk.Entry(root, textvariable=Latitude)

Latitude\_entry.grid(row=8, column=1, padx=5, pady=5)

Longtitude = tk.IntVar()

Longtitude\_label = tk.Label(root, text="Долгота:")

Longtitude\_label.grid(row=9, column=0, sticky="w")

Longtitude\_entry = tk.Entry(root, textvariable=Longtitude)

Longtitude\_entry.grid(row=9, column=1, padx=5, pady=5)

cmb\_VEI = ttk.Combobox(root)

VEI\_label = tk.Label(root, text="Индекс взрывоопасности:")

VEI\_label.grid(row=10, column=0, sticky="w", padx=5, pady=5)

cmb\_VEI['values'] = ('0', '1', '2', '3', '4', '5', '6', '7', '8')

cmb\_VEI.current(0)

cmb\_VEI.grid(column=1, row=10)

cmb\_agent = ttk.Combobox(root)

Agent\_label = tk.Label(root, text="Причина извержения:")

Agent\_label.grid(row=10, column=3, sticky="w", padx=5, pady=5)

cmb\_agent['values'] = ('A', 'E', 'F', 'G', 'I', 'L', 'M', 'm', 'P', 'S', 'T', 'W')

cmb\_agent.current(0)

cmb\_agent.grid(column=4, row=10)

Deaths = tk.IntVar()

Deaths\_label = tk.Label(root, text="Количество смертей:")

Deaths\_label.grid(row=13, column=0, sticky="w")

Deaths\_entry = tk.Entry(root, textvariable=Deaths)

Deaths\_entry.grid(row=13, column=1, padx=5, pady=5)

Injured = tk.IntVar()

Injured\_label = tk.Label(root, text="Количество пострадавших:")

Injured\_label.grid(row=14, column=0, sticky="w")

Injured\_entry = tk.Entry(root, textvariable=Injured)

Injured\_entry.grid(row=14, column=1, padx=5, pady=5)

Lost = tk.IntVar()

Lost\_label = tk.Label(root, text="Количество пропавших:")

Lost\_label.grid(row=15, column=0, sticky="w")

Lost\_entry = tk.Entry(root, textvariable=Lost)

Lost\_entry.grid(row=15, column=1, padx=5, pady=5)

Damage = tk.IntVar()

Damage\_label = tk.Label(root, text="Ущерб в млн долларов:")

Damage\_label.grid(row=16, column=0, sticky="w")

Damage\_entry = tk.Entry(root, textvariable=Damage)

Damage\_entry.grid(row=16, column=1, padx=5, pady=5)

TSU = tk.BooleanVar()

TSU.set(False)

TSU1 = ttk.Checkbutton(root, text="Было цунами?", var=TSU)

TSU1.grid(column=0, row=17)

EQ = tk.BooleanVar()

EQ.set(False)

EQ1 = ttk.Checkbutton(root, text="Было землетрясение?", var=EQ)

EQ1.grid(column=2, row=17)

list4values = {'Year': Year, 'Month': cmb\_month,

'Day': Day, 'Name': name, 'Location': location,

'Country': country,

'Latitude': Latitude, 'Longitude': Longtitude,

'Elevation': Height, 'Type': Type, 'VEI': cmb\_VEI,

'Agent': cmb\_agent, 'DEATHS': Deaths, 'INJURIES': Injured, 'MISSING': Lost,

'DAMAGE\_MILLIONS\_DOLLARS': Damage, 'TSU': TSU, 'EQ': EQ}

message\_button = tk.Button(root, text="Ввести",

command=lambda \*args: accept(root, list4values))

message\_button.grid(row=19, column=3, padx=5, pady=5, sticky="e")

elif form == "Вид первый":

name = tk.StringVar()

name\_label = tk.Label(root, text="Название вулкана:")

name\_label.grid(row=1, column=0, sticky="w")

name\_entry = tk.Entry(root, textvariable=name)

name\_entry.grid(row=1, column=1, padx=5, pady=5)

Type = tk.StringVar()

Type\_label = tk.Label(root, text="Тип вулкана:")

Type\_label.grid(row=2, column=0, sticky="w")

Type\_entry = tk.Entry(root, textvariable=Type)

Type\_entry.grid(row=2, column=1, padx=5, pady=5)

Height = tk.IntVar()

Height\_label = tk.Label(root, text="Высота вулкана (в метрах):")

Height\_label.grid(row=3, column=0, sticky="w")

Height\_entry = tk.Entry(root, textvariable=Height)

Height\_entry.grid(row=3, column=1, padx=5, pady=5)

Latitude = tk.IntVar()

Latitude\_label = tk.Label(root, text="Широта:")

Latitude\_label.grid(row=4, column=0, sticky="w")

Latitude\_entry = tk.Entry(root, textvariable=Latitude)

Latitude\_entry.grid(row=4, column=1, padx=5, pady=5)

Longtitude = tk.IntVar()

Longtitude\_label = tk.Label(root, text="Долгота:")

Longtitude\_label.grid(row=5, column=0, sticky="w")

Longtitude\_entry = tk.Entry(root, textvariable=Longtitude)

Longtitude\_entry.grid(row=5, column=1, padx=5, pady=5)

list4values = {'Name': name, 'Latitude': Latitude,

'Longitude': Longtitude, 'Elevation': Height, 'Type': Type}

message\_button = tk.Button(root, text="Ввести",

command=lambda \*args: accept(root, list4values))

message\_button.grid(row=7, column=3, padx=5, pady=5, sticky="e")

elif form == "Вид второй":

country = tk.StringVar()

country\_label = tk.Label(root, text="Страна:")

country\_label.grid(row=1, column=0, sticky="w")

country\_entry = tk.Entry(root, textvariable=country)

country\_entry.grid(row=1, column=1, padx=5, pady=5)

location = tk.StringVar()

location\_label = tk.Label(root, text="Расположение вулкана:")

location\_label.grid(row=2, column=0, sticky="w")

location\_entry = tk.Entry(root, textvariable=location)

location\_entry.grid(row=2, column=1, padx=5, pady=5)

Latitude = tk.IntVar()

Latitude\_label = tk.Label(root, text="Широта:")

Latitude\_label.grid(row=3, column=0, sticky="w")

Latitude\_entry = tk.Entry(root, textvariable=Latitude)

Latitude\_entry.grid(row=3, column=1, padx=5, pady=5)

Longtitude = tk.IntVar()

Longtitude\_label = tk.Label(root, text="Долгота:")

Longtitude\_label.grid(row=4, column=0, sticky="w")

Longtitude\_entry = tk.Entry(root, textvariable=Longtitude)

Longtitude\_entry.grid(row=4, column=1, padx=5, pady=5)

list4values = {'Location': location, 'Country': country,

'Latitude': Latitude, 'Longitude': Longtitude}

message\_button = tk.Button(root, text="Ввести",

command=lambda \*args: accept(root, list4values))

message\_button.grid(row=6, column=3, padx=5, pady=5, sticky="e")

elif form == "Вид третий":

Year = tk.IntVar()

Year\_label = tk.Label(root, text="Год извержения:")

Year\_label.grid(row=0, column=0, sticky="w")

Year\_entry = tk.Entry(root, textvariable=Year)

Year\_entry.grid(row=0, column=1, padx=5, pady=5)

cmb\_month = ttk.Combobox(root)

Month\_label = tk.Label(root, text="Месяц извержения:")

Month\_label.grid(row=1, column=0, sticky="w", padx=5, pady=5)

cmb\_month['values'] = ('1', '2', '3', '4', '5', '6', '7',

'8', '9', '10', '11', '12')

cmb\_month.current(0)

cmb\_month.grid(column=1, row=1)

Day = tk.IntVar()

Day\_label = tk.Label(root, text="День извержения:")

Day\_label.grid(row=2, column=0, sticky="w")

Day\_entry = tk.Entry(root, textvariable=Day)

Day\_entry.grid(row=2, column=1, padx=5, pady=5)

name = tk.StringVar()

name\_label = tk.Label(root, text="Название вулкана:")

name\_label.grid(row=3, column=0, sticky="w")

name\_entry = tk.Entry(root, textvariable=name)

name\_entry.grid(row=3, column=1, padx=5, pady=5)

cmb\_VEI = ttk.Combobox(root)

VEI\_label = tk.Label(root, text="Индекс взрывоопасности:")

VEI\_label.grid(row=4, column=0, sticky="w", padx=5, pady=5)

cmb\_VEI['values'] = ('0', '1', '2', '3', '4', '5', '6', '7', '8')

cmb\_VEI.current(0)

cmb\_VEI.grid(column=1, row=4)

cmb\_agent = ttk.Combobox(root)

Agent\_label = tk.Label(root, text="Причина извержения:")

Agent\_label.grid(row=4, column=3, sticky="w", padx=5, pady=5)

cmb\_agent['values'] = ('A', 'E', 'F', 'G', 'I', 'L', 'M', 'm', 'P', 'S', 'T', 'W')

cmb\_agent.current(0)

cmb\_agent.grid(column=4, row=4)

Deaths = tk.IntVar()

Deaths\_label = tk.Label(root, text="Количество смертей:")

Deaths\_label.grid(row=5, column=0, sticky="w")

Deaths\_entry = tk.Entry(root, textvariable=Deaths)

Deaths\_entry.grid(row=5, column=1, padx=5, pady=5)

Injured = tk.IntVar()

Injured\_label = tk.Label(root, text="Количество пострадавших:")

Injured\_label.grid(row=6, column=0, sticky="w")

Injured\_entry = tk.Entry(root, textvariable=Injured)

Injured\_entry.grid(row=6, column=1, padx=5, pady=5)

Lost = tk.IntVar()

Lost\_label = tk.Label(root, text="Количество пропавших:")

Lost\_label.grid(row=7, column=0, sticky="w")

Lost\_entry = tk.Entry(root, textvariable=Lost)

Lost\_entry.grid(row=7, column=1, padx=5, pady=5)

Damage = tk.IntVar()

Damage\_label = tk.Label(root, text="Ущерб в млн долларов:")

Damage\_label.grid(row=8, column=0, sticky="w")

Damage\_entry = tk.Entry(root, textvariable=Damage)

Damage\_entry.grid(row=8, column=1, padx=5, pady=5)

TSU = tk.BooleanVar()

TSU.set(False)

TSU1 = ttk.Checkbutton(root, text="Было цунами?", var=TSU)

TSU1.grid(column=0, row=9)

EQ = tk.BooleanVar()

EQ.set(False)

EQ1 = ttk.Checkbutton(root, text="Было землетрясение?", var=EQ)

EQ1.grid(column=2, row=9)

list4values = {'Year': Year, 'Month': cmb\_month, 'Day': Day, 'Name': name, 'VEI': cmb\_VEI,

'Agent': cmb\_agent, 'DEATHS': Deaths, 'INJURIES': Injured, 'MISSING': Lost,

'DAMAGE\_MILLIONS\_DOLLARS': Damage, 'TSU': TSU, 'EQ': EQ}

message\_button = tk.Button(root, text="Ввести",

command=lambda \*args: accept(root, list4values))

message\_button.grid(row=11, column=3, padx=5, pady=5, sticky="e")

def accept(root, list4values):

"""

Автор: Баканов Г., Подкопаева П.

Цель: Проверка правильности введённых данных в окно

Вход: Значение переменной любого типа

Выход: нет

"""

flag = True

if (list4values['Day'].get() > 29) and (list4values['Month'].get() == 2):

flag = False

elif list4values['Day'].get() > 31 or list4values['Day'].get() < 1:

flag = False

if list4values['Elevation'].get() > 6887:

flag = False

if (list4values['Latitude'].get() > 180) or (list4values['Latitude'].get() < -180):

flag = False

if (list4values['Longitude'].get() > 180) or (list4values['Longitude'].get() < -180):

flag = False

if flag:

glob.current\_base = glob.current\_base.append({k: v.get() for k,

v in list4values.items()}, ignore\_index=True)

glob.current\_base = glob.correct\_base\_values(glob.current\_base)

glob.work\_list[glob.current\_base\_name] = glob.current\_base

new\_item = glob.table4base.insert('', 'end', iid=len(glob.current\_base.index) - 1)

for i in glob.columns:

glob.table4base.set(new\_item, column=i, value=list4values[i].get())

mb.showinfo("Сообщение", "Занесено в базу")

glob.mark\_changes()

glob.update\_list()

root.destroy()

else:

err.error("Данные введены некорректно, повторите попытку")

## Модуль new\_map.py

"""

Цель: Модуль для построения карты

"""

import tkinter as tk

import tkinter.ttk as ttk

from tkinter import messagebox as mb

import folium

from folium.plugins import MarkerCluster

from Scripts import base\_handling as hand\_base

from Scripts import globalvars as glob

from Scripts import constants

def choice\_map(root: tk.Tk, pane: ttk.Panedwindow):

"""

Автор: Подкопаева П.

Цель: Функция для выбора типа карты

Вход: Главное окно, растягивающий виджет

Выход: Нет

"""

global CHOSEN\_VALUE\_MAP

if not glob.is\_db\_open():

return "break"

win = tk.Toplevel(root)

win.title("Выбор")

win.geometry("300x300+500+200")

background = tk.Frame(win, bg=constants.style['bg'])

background.place(x=0, y=0, relwidth=1, relheight=1)

choice = ("По высоте", "По смертности", "По ущербу")

make\_map = tk.Label(background, text='Это окно для построения карт', bg=constants.style['bg'])

make\_map.place(relx=0.1, rely=0.1)

make\_map.pack()

CHOSEN\_VALUE\_MAP = tk.StringVar(value='Выберите тип карты')

make\_table\_op = tk.OptionMenu(background, CHOSEN\_VALUE\_MAP, \*choice)

make\_table\_op.place(relx=0.25, rely=0.4)

make\_table\_op.pack()

button\_statistics = tk.Button(background, text='Сохранить', bg=constants.style['plots\_button'])

button\_statistics.bind("<Button-1>", lambda \*args: new\_map())

button\_statistics.place(relx=0.2, rely=0.5, relheight=0.1, relwidth=0.6)

background.pack(side="top", fill="both", expand=True, padx=10, pady=5)

def new\_map():

"""

Автор: Подкопаева П.

Цель: Функция для построения карты

Вход: Нет

Выход: Нет

"""

global CHOSEN\_VALUE\_MAP

lat = hand\_base.bd['Latitude']

lon = hand\_base.bd['Longitude']

if CHOSEN\_VALUE\_MAP.get() == 'Выберите тип карты':

mb.showerror("Ошибка!", "Сначала выберите вариант карты!")

elif CHOSEN\_VALUE\_MAP.get() == 'По высоте':

map\_elevation(lat, lon)

mb.showinfo("Инфо", "Карта высот сохранена")

elif CHOSEN\_VALUE\_MAP.get() == 'По смертности':

map\_deaths(lat, lon)

mb.showinfo("Инфо", "Карта смертности сохранена")

elif CHOSEN\_VALUE\_MAP.get() == 'По ущербу':

map\_damage(lat, lon)

mb.showinfo("Инфо", "Карта ущерба сохранена")

# для высоты

def map\_elevation(lat, lon):

"""

Автор: Баканов Г.

Цель: Функция для построения карты по высоте

Вход: dataframe

Выход: нет (файл)

"""

elevation = hand\_base.bd['Elevation']

name = hand\_base.bd['Name']

def color\_change(elevation):

"""

Автор: Баканов Г.

Цель: Функция, устанавливающая цвета

Вход: Int высота

Выход: Строка с цветом

"""

if elevation < 500:

return 'green'

if 500 <= elevation < 1000:

return 'yellow'

if 1000 <= elevation < 3000:

return 'red'

if elevation > 3000:

return 'black'

map = folium.Map(location=[-6.2146200, 106.8451300], zoom\_start=6, titles="Mapbox bright")

marker\_cluster = MarkerCluster().add\_to(map)

for lat1, lon1, elevation, name in zip(lat, lon, elevation, name):

folium.Marker(location=[lat1, lon1], radius=9,

popup='Name: ' + str(name) + '\n' + 'Elevation: ' + str(elevation) + " m",

fill\_color=color\_change(elevation),

color="gray", fill\_opacity=0.9).add\_to(marker\_cluster)

map.save("../Output/map\_elevation.html")

# для смертности

def map\_deaths(lat, lon):

"""

Автор: Ковязин В.

Цель: Функция для построения карты по смертям

Вход: dataframe

Выход: нет(файл)

"""

deaths = hand\_base.bd['DEATHS'].fillna(0)

name = hand\_base.bd['Name']

def color\_change(deaths):

"""

Автор: Ковязин В.

Цель: Функция, устанавливающая цвета

Вход: Int смерти

Выход: Строка с цветом

"""

if deaths < 500:

return 'green'

if 500 <= deaths < 1000:

return 'yellow'

if 1000 <= deaths < 2000:

return 'red'

if deaths > 2000:

return 'black'

map = folium.Map(location=[-6.2146200, 106.8451300], zoom\_start=6, titles="Mapbox bright")

marker\_cluster = MarkerCluster().add\_to(map)

for lat1, lon1, deaths, name in zip(lat, lon, deaths, name):

folium.Marker(location=[lat1, lon1], radius=9,

popup='Name: ' + str(name) + '\n' + 'Deaths: ' + str(deaths),

fill\_color=color\_change(deaths),

color="gray", fill\_opacity=0.9).add\_to(marker\_cluster)

map.save("../Output/map\_deaths.html")

# для ущерба

def map\_damage(lat, lon):

"""

Автор: Подкопаева П.

Цель: Функция для построения карты по ущербу

Вход: dataframe

Выход: нет (файл)

"""

damage = hand\_base.bd['DAMAGE\_MILLIONS\_DOLLARS'].fillna(0)

name = hand\_base.bd['Name']

def color\_change(damage):

"""

Автор: Подкопаева П.

Цель: Функция, устанавливающая цвета

Вход: Int ущерб

Выход: Строка с цветом

"""

if damage < 50:

return 'green'

if 50 <= damage < 100:

return 'yellow'

if 100 <= damage < 200:

return 'red'

if damage > 200:

return 'black'

map = folium.Map(location=[-6.2146200, 106.8451300], zoom\_start=6, titles="Mapbox bright")

marker\_cluster = MarkerCluster().add\_to(map)

for lat1, lon1, damage, name in zip(lat, lon, damage, name):

folium.Marker(location=[lat1, lon1], radius=9,

popup='Name: ' + str(name) + '\n' + 'Damage: ' + str(damage),

fill\_color=color\_change(damage),

color="gray", fill\_opacity=0.9).add\_to(marker\_cluster)

map.save("../Output/map\_damage.html")

## Модуль new\_table.py

"""

Цель: Модуль для построения сводных таблиц баз данных

"""

import tkinter as tk

import tkinter.ttk as ttk

from tkinter import messagebox as mb

import pandas as pd

from Scripts import base\_handling as hand\_base

from Scripts import globalvars as glob

def statoutput(bd):

"""

Автор: Подкопаева П.

Цель: Функция, создающая список из datframe для использования при выводе

Вход: dataframe

Выход: список из элементов dataframe

"""

shape = bd.shape

row = shape[0]

full\_list = []

for i in range(0, row):

list\_row = []

for j in bd.iloc[i]:

list\_row.append(j)

full\_list.append(list\_row)

return full\_list

def choice\_table(root: tk.Tk, pane: ttk.Panedwindow):

"""

Автор: Баканов Г.

Цель: Выбор фильтров для сводной таблицы

Вход: Главное окно, растягивающий виджет

Выход: Нет

"""

global CHOSEN\_VALUE

if not glob.is\_db\_open():

return "break"

win = tk.Toplevel(root)

win.title("Выбор")

win.geometry("300x300+500+200")

background = tk.Frame(win, bg="#F8F8FF")

background.place(x=0, y=0, relwidth=1, relheight=1)

choice = ("Выберите фильтры", "Страна - Средняя смертность", "Страна - Средняя высота вулкана",

"Расположение - Средняя смертность",

"Тип вулкана - Средняя смертность", "Тип вулкана - Количество пропавших",

"Название вулкана - Количество раненных")

# make\_table = tk.Label(background)

CHOSEN\_VALUE = tk.StringVar(value='Выберите фильтры')

make\_table\_op = tk.OptionMenu(background, CHOSEN\_VALUE, \*choice)

make\_table\_op.place(relx=0.25, rely=0.25)

make\_table\_op.pack()

button\_statistics = tk.Button(background, text='Посмотреть статистику', bg="#AFEEEE")

button\_statistics.bind("<Button-1>", lambda \*args: stat\_table\_window(root, pane))

button\_statistics.place(relx=0.25, rely=0.5, relheight=0.1, relwidth=0.6)

background.pack(side="top", fill="both", expand=True, padx=10, pady=5)

def name\_injuries():

"""

Автор: Ковязин В.

Цель: Создание dataframe по фильтрам

Вход: нет

Выход: dataframe

"""

general = pd.DataFrame(hand\_base.bd)

general['INJURIES'] = general['INJURIES'].apply(pd.to\_numeric, errors='coerce')

rat = general.groupby(['Name']).agg({'INJURIES': "sum"})

rat.rename(columns={'Name': 'Название', 'INJURIES': 'Раненые'}, inplace=True)

rat = rat.reset\_index()

rat = rat.round({'Injuries': 2})

rat.fillna(0)

# print(rat)

return rat.fillna(0)

def country\_deaths\_mean():

"""

Автор:Ковязин В.

Цель: Подсчёт средней смертности для каждой страны

Входные параметры: DataFrame

Возвращает: нет

"""

general = pd.DataFrame(hand\_base.bd)

general['DEATHS'] = general['DEATHS'].apply(pd.to\_numeric, errors='coerce')

rat = general.groupby(['Country']).agg({'DEATHS': "mean"})

rat.rename(columns={'Country': 'Страна', 'DEATHS': 'Средняя смертность'}, inplace=True)

rat = rat.reset\_index()

rat = rat.round({'Average number of deaths': 2})

rat.fillna(0)

# print(rat)

return rat.fillna(0)

def country\_elevation\_mean():

"""

Автор:Подкопаева П.

Цель: Создание dataframe по фильтрам

Вход: нет

Выход: dataframe

"""

general = pd.DataFrame(hand\_base.bd)

general['Elevation'] = general['Elevation'].apply(pd.to\_numeric, errors='coerce')

rat = general.groupby(['Country']).agg({'Elevation': "mean"})

rat.rename(columns={'Country': 'Страна', 'Elevation': 'Средняя высота вулкана'}, inplace=True)

rat = rat.reset\_index()

rat = rat.round({'Average elevation': 2})

# print(rat)

return rat.fillna(0)

def location\_deaths\_mean():

"""

Автор: Подкопаева П.

Цель: Создание dataframe по фильтрам

Вход: нет

Выход: dataframe

"""

general = pd.DataFrame(hand\_base.bd)

general['DEATHS'] = general['DEATHS'].apply(pd.to\_numeric, errors='coerce')

rat = general.groupby(['Location']).agg({'DEATHS': "mean"})

rat.rename(columns={'Location': 'Местность', 'DEATHS': 'Средняя смертность'}, inplace=True)

rat = rat.reset\_index()

rat = rat.round({'Average number of deaths': 2})

rat.fillna(0)

# print(rat)

return rat.fillna(0)

def type\_deaths\_mean():

"""

Автор: Баканов Г.

Цель: Создание dataframe по фильтрам

Вход: нет

Выход: dataframe

"""

general = pd.DataFrame(hand\_base.bd)

general['DEATHS'] = general['DEATHS'].apply(pd.to\_numeric, errors='coerce')

rat = general.groupby(['Type']).agg({'DEATHS': "mean"})

rat.rename(columns={'Type': 'Тип вулкана', 'DEATHS': 'Средняя смертность'}, inplace=True)

rat = rat.reset\_index()

rat = rat.round({'Average number of deaths': 2})

# print(rat.fillna(0))

return rat.fillna(0)

def type\_missing\_num():

"""

Автор: Баканов Г.

Цель: Создание dataframe по фильтрам

Вход: нет

Выход: dataframe

"""

general = pd.DataFrame(hand\_base.bd)

general['MISSING'] = general['MISSING'].apply(pd.to\_numeric, errors='coerce')

num = general.groupby(['Type']).agg({'MISSING': "sum"})

num.rename(columns={'Type': 'Тип вулкана',

'MISSING': 'Количество известных пропавших'}, inplace=True)

num = num.reset\_index()

num = num.round({'Number of missing': 2})

# print(num.fillna(0))

return num

def stat\_table\_window(root: tk.Tk, pane: ttk.Panedwindow):

"""

Автор: Подкопаева П., Ковязин В.

Цель: Функция, создающая окно для просмотра сводной таблицы

Вход: нет

Выход: нет

"""

global CHOSEN\_VALUE

win = tk.Toplevel(root)

win.title("Таблица")

win.geometry("600x500+500+200")

background = tk.Frame(win, bg="#F8F8FF")

background.place(x=0, y=0, relwidth=1, relheight=1)

name\_lab = tk.Label(background, font=1,

text='Таблица по выбранному фильтру\n' + CHOSEN\_VALUE.get(), bg="#F5F5F5")

name\_lab.place(relx=0.01, rely=0.01, relwidth=0.95, relheight=0.1)

answerfr = tk.Frame(background, bg="#F5F5F5")

answerfr.place(relx=0.025, rely=0.13, relwidth=0.95, relheight=0.8)

answerfr\_1 = tk.Frame(answerfr, bg="#F5F5F5")

answerfr\_1.place(relx=0.01, rely=0.01, relwidth=0.98, relheight=0.98)

if CHOSEN\_VALUE.get() == 'Выберите фильтры':

mb.showerror("Ошибка!", "Сначала выберите фильтр!")

win.destroy()

elif CHOSEN\_VALUE.get() == 'Страна - Средняя смертность':

# выбор параметра и строится соответствующая таблица

bd\_series = country\_deaths\_mean()

bd = bd\_series

list\_ = statoutput(bd)

table = ttk.Treeview(answerfr\_1, columns=('Страна', 'Средняя смертность'),

height=15, show='headings')

table.column('Страна', anchor=tk.CENTER)

table.column('Средняя смертность', anchor=tk.CENTER)

table.heading('Страна', text='Страна')

table.heading('Средняя смертность', text='Средняя смертность')

for row in list\_:

table.insert('', tk.END, values=row)

table.place(relwidth=1, relheight=1)

# кнопка сохранения

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save\_button(bd))

button\_save.place(relx=0.755, rely=0.9375, relheight=0.0495, relwidth=0.22)

scrolltable = tk.Scrollbar(answerfr\_1, orient=tk.VERTICAL, command=table.yview)

table.config(yscrollcommand=scrolltable.set)

scrolltable.pack(side=tk.RIGHT, fill=tk.Y)

elif CHOSEN\_VALUE.get() == 'Название вулкана - Количество раненных':

# выбор параметра и строится соответствующая таблица

bd\_series = name\_injuries()

bd = bd\_series

list\_ = statoutput(bd)

table = ttk.Treeview(answerfr\_1, columns=('Название', 'Раненые'),

height=15, show='headings')

table.column('Название', anchor=tk.CENTER)

table.column('Раненые', anchor=tk.CENTER)

table.heading('Название', text='Название')

table.heading('Раненые', text='Раненые')

for row in list\_:

table.insert('', tk.END, values=row)

table.place(relwidth=1, relheight=1)

# кнопка сохранения

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save\_button(bd))

button\_save.place(relx=0.755, rely=0.9375, relheight=0.0495, relwidth=0.22)

scrolltable = tk.Scrollbar(answerfr\_1, orient=tk.VERTICAL, command=table.yview)

table.config(yscrollcommand=scrolltable.set)

scrolltable.pack(side=tk.RIGHT, fill=tk.Y)

elif CHOSEN\_VALUE.get() == 'Страна - Средняя высота вулкана':

# выбор параметра и соответсвующая параметру таблица строится

bd\_series = country\_elevation\_mean()

bd = bd\_series

list\_ = statoutput(bd)

table = ttk.Treeview(answerfr\_1, columns=('Страна',

'Средняя высота вулкана'),

height=15, show='headings')

table.column('Страна', anchor=tk.CENTER)

table.column('Средняя высота вулкана', anchor=tk.CENTER)

table.heading('Страна', text='Страна')

table.heading('Средняя высота вулкана', text='Средняя высота вулкана')

for row in list\_:

table.insert('', tk.END, values=row)

table.place(relwidth=1, relheight=1)

# кнопка сохранения

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save\_button(bd))

button\_save.place(relx=0.755, rely=0.9375, relheight=0.0495, relwidth=0.22)

scrolltable = tk.Scrollbar(answerfr\_1, orient=tk.VERTICAL, command=table.yview)

table.config(yscrollcommand=scrolltable.set)

scrolltable.pack(side=tk.RIGHT, fill=tk.Y)

elif CHOSEN\_VALUE.get() == 'Расположение - Средняя смертность':

# выбор параметра и соответсвующая параметру таблица строится

bd\_series = location\_deaths\_mean()

bd = bd\_series

list\_ = statoutput(bd)

table = ttk.Treeview(answerfr\_1, columns=('Расположение',

'Средняя смертность'),

height=15, show='headings')

table.column('Расположение', anchor=tk.CENTER)

table.column('Средняя смертность', anchor=tk.CENTER)

table.heading('Расположение', text='Расположение')

table.heading('Средняя смертность', text='Средняя смертность')

for row in list\_:

table.insert('', tk.END, values=row)

table.place(relwidth=1, relheight=1)

# кнопка сохранения

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save\_button(bd))

button\_save.place(relx=0.755, rely=0.9375, relheight=0.0495, relwidth=0.22)

scrolltable = tk.Scrollbar(answerfr\_1, orient=tk.VERTICAL, command=table.yview)

table.config(yscrollcommand=scrolltable.set)

scrolltable.pack(side=tk.RIGHT, fill=tk.Y)

elif CHOSEN\_VALUE.get() == 'Тип вулкана - Средняя смертность':

# выбор параметра и соответсвующая параметру таблица строится

bd\_series = type\_deaths\_mean()

bd = bd\_series

list\_ = statoutput(bd)

table = ttk.Treeview(answerfr\_1, columns=('Тип вулкана',

'Средняя смертность'),

height=15, show='headings')

table.column('Тип вулкана', anchor=tk.CENTER)

table.column('Средняя смертность', anchor=tk.CENTER)

table.heading('Тип вулкана', text='Тип вулкана')

table.heading('Средняя смертность', text='Средняя смертность')

for row in list\_:

table.insert('', tk.END, values=row)

table.place(relwidth=1, relheight=1)

# кнопка сохранения

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save\_button(bd))

button\_save.place(relx=0.755, rely=0.9375, relheight=0.0495, relwidth=0.22)

scrolltable = tk.Scrollbar(answerfr\_1, orient=tk.VERTICAL, command=table.yview)

table.config(yscrollcommand=scrolltable.set)

scrolltable.pack(side=tk.RIGHT, fill=tk.Y)

elif CHOSEN\_VALUE.get() == 'Тип вулкана - Количество пропавших':

# выбор параметра и соответсвующая параметру таблица строится

bd\_series = type\_missing\_num()

bd = bd\_series

list\_ = statoutput(bd)

table = ttk.Treeview(answerfr\_1, columns=('Тип вулкана',

'Количество пропавших'), height=15,

show='headings')

table.column('Тип вулкана', anchor=tk.CENTER)

table.column('Количество пропавших', anchor=tk.CENTER)

table.heading('Тип вулкана', text='Тип вулкана')

table.heading('Количество пропавших', text='Количество пропавших')

for row in list\_:

table.insert('', tk.END, values=row)

scrolltable = tk.Scrollbar(answerfr\_1, orient=tk.VERTICAL, command=table.yview)

table.config(yscrollcommand=scrolltable.set)

scrolltable.pack(side=tk.RIGHT, fill=tk.Y)

table.place(relwidth=1, relheight=1)

# кнопка сохранения

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save\_button(bd))

button\_save.place(relx=0.755, rely=0.9375, relheight=0.0495, relwidth=0.22)

def save\_button(bd):

"""

Автор: Подкопаева П.

Цель: Функция для сохранения таблицы

Вход: dataframe

Выход: Нет

"""

global CHOSEN\_VALUE

name = CHOSEN\_VALUE.get()

bd.to\_csv("../Output/" + name + ".csv", index=False)

mb.showinfo("Информация", "Таблица " + name + " успешно сохранена")

## Модуль statistics.py

"""

Цель: Модуль, содержащий функции для статистического анализа баз данных

"""

import math as m

import tkinter as tk

import tkinter.ttk as ttk

from tkinter import messagebox as mb, filedialog

import matplotlib.pyplot as plt

import pandas as pd

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

from Scripts import base\_handling as hand\_base

from Scripts import globalvars as glob

from Scripts import new\_table as tb

from Scripts import constants

def statistics\_base(root: tk.Toplevel, pane: ttk.Panedwindow, string):

"""

Автор: Баканов Г.

Цель: Формирует и сохраняет отчёт по основным статистикам для количественных переменных

Вход: dataframe

Выход: Нет (файл)

"""

win = tk.Toplevel(root, bg=constants.style['bg'])

win.title("Статистические данные: " + string)

win.geometry('600x400+500+300')

bd = glob.current\_base

sortedSample = bd[string].sort\_values()[bd[string] == bd[string]] # get rid of nan

average = m.ceil(sortedSample.mean())

med = sortedSample.median()

mode = sortedSample.mode().values[0]

if average == med:

s = 'симметрия'

elif average > med:

s = 'acимметрия вправо'

else:

s = 'acимметрия влево'

disp = (sortedSample.apply(lambda x: (x - average) \*\* 2).sum()) / (len(sortedSample) - 1)

tree = ttk.Treeview(win, height=20, show='headings')

tree["columns"] = ("one", "two")

tree.column("#0", width=270, minwidth=270, stretch=tk.NO)

tree.column("one", width=300, minwidth=300, stretch=tk.NO)

tree.column("two", width=300, minwidth=300, stretch=tk.NO)

i = 0

tree.heading("#0", anchor=tk.W)

tree.heading("one", text="Характеристика", anchor=tk.W)

tree.heading("two", text="Значение", anchor=tk.W)

tree.insert('', tk.END, iid=i, values=["Среднее арифметическое", average])

i += 1

tree.insert('', tk.END, iid=i, values=["Медиана", med])

i += 1

tree.insert('', tk.END, iid=i, values=["Мода", mode])

i += 1

tree.insert('', tk.END, iid=i, values=["Форма плотности распределения", s])

i += 1

tree.insert('', tk.END, iid=i, values=["Выборочная дисперсия", disp])

i += 1

tree.insert('', tk.END, iid=i, values=["Нижняя квартиль", sortedSample.quantile(0.25)])

i += 1

tree.insert('', tk.END, iid=i, values=["Верхняя квартиль", sortedSample.quantile(0.75)])

i += 1

tree.insert('', tk.END, iid=i, values=["Межквартильный размах",

sortedSample.quantile(0.75) - sortedSample.quantile(0.25)])

i += 1

tree.insert('', tk.END, iid=i, values=["Максимум", sortedSample.max()])

i += 1

tree.insert('', tk.END, iid=i, values=["Минимум", sortedSample.min()])

i += 1

tree.insert('', tk.END, iid=i, values=["Размах", (sortedSample.max() - sortedSample.min())])

i += 1

tree.insert('', tk.END, iid=i, values=["Стандартное отклонение (СКО)", sortedSample.std()])

i += 1

tree.insert('', tk.END, iid=i, values=["Коэффициент вариации",

sortedSample.std() / sortedSample.mean()])

vsb = ttk.Scrollbar(win, orient="vertical", command=tree.yview)

tree.configure(yscrollcommand=vsb.set)

vsb.pack(side='right', fill='both')

save\_button = tk.Button(win, text='Сохранить', bg=constants.style['apply\_button'])

save\_button.bind("<Button-1>", lambda \*args: save\_stat(tree, i))

save\_button.pack(side='top')

tree.pack(side=tk.TOP, fill=tk.X)

def save\_stat(tree, i):

"""

Автор: Баканов Г.

Цель: Функция сохраниения статистического отчёта в файл

Вход: Статистический отчёт (tree)

Выход: нет (файл)

"""

data = pd.DataFrame()

for index in range(i):

data = data.append(tree.set(index), ignore\_index=True)

path = filedialog.asksaveasfilename(initialdir="../Output/",

filetypes=(("Database files", "\*.csv"),

("All files", "\*.\*")))

if ".csv" not in path:

path += ".csv"

data.to\_csv(path, index=False, encoding='cp1251')

return "break"

def graphics\_choice(root: tk.Tk, pane: ttk.Panedwindow):

"""

Автор: Подкопаева П.

Цель: Выбор фильтров для построения графического отчёта

Вход: главное окно, растягивающийся виджет

Выход: нет

"""

global CHOSEN\_VALUE1, CHOSEN\_VALUE2, CHOSEN\_VALUE3, CHOSEN\_VALUE4, CHOSEN\_VALUE5, CHOSEN\_VALUE6

if not glob.is\_db\_open():

return "break"

win = tk.Toplevel(root)

win.title("Выбор")

win.geometry("700x500+500+200")

background = tk.Frame(win, bg=constants.style['bg'])

background.place(x=0, y=0, relwidth=1, relheight=1)

choice\_graph = ("Фильтры для линейного графика", "Год - Средняя смертность")

CHOSEN\_VALUE1 = tk.StringVar(value='Фильтры для линейного графика')

make\_gr = tk.OptionMenu(background, CHOSEN\_VALUE1, \*choice\_graph)

make\_gr.place(relx=0.1, rely=0.1)

make\_gr.pack()

button\_graph = tk.Button(background,

text='Построить график',

bg=constants.style['plots\_button'])

button\_graph.bind("<Button-1>", lambda \*args: draw\_graph(root, pane))

button\_graph.place(relx=0.0, rely=0.5, relheight=0.1, relwidth=0.2)

choice\_diagram = (

"Фильтры для столбчатой диаграммы",

"Страна - Средняя смертность",

"Страна - Средняя высота вулкана",

"Расположение - Средняя смертность",

"Тип вулкана - Средняя смертность",

"Тип вулкана - Количество пропавших")

CHOSEN\_VALUE2 = tk.StringVar(value='Фильтры для столбчатой диаграммы')

make\_dgrm = tk.OptionMenu(background, CHOSEN\_VALUE2, \*choice\_diagram)

make\_dgrm.place(relx=0.3, rely=0.1)

make\_dgrm.pack()

button\_diagram = tk.Button(background,

text='Построить диаграмму',

bg=constants.style['plots\_button'])

button\_diagram.bind("<Button-1>", lambda \*args: draw\_diagram(root, pane))

button\_diagram.place(relx=0.4, rely=0.5, relheight=0.1, relwidth=0.2)

choice\_pie = (

"Фильтры для круговой диаграммы",

"Страна - Средняя смертность",

"Тип - Средняя смертность", "Тип - Ранения")

CHOSEN\_VALUE3 = tk.StringVar(value='Фильтры для круговой диаграммы')

make\_pie = tk.OptionMenu(background, CHOSEN\_VALUE3, \*choice\_pie)

make\_pie.place(relx=0.5, rely=0.1)

make\_pie.pack()

button\_pie = tk.Button(background, text='Построить "пирог"', bg=constants.style['plots\_button'])

button\_pie.bind("<Button-1>", lambda \*args: draw\_pie(root, pane))

button\_pie.place(relx=0.8, rely=0.5, relheight=0.1, relwidth=0.2)

choice\_viskers = ("Фильтры для диаграммы Бокса-Вискера", "Высота - Тип")

CHOSEN\_VALUE4 = tk.StringVar(value='Фильтры для диаграммы Бокса-Вискера')

make\_box = tk.OptionMenu(background, CHOSEN\_VALUE4, \*choice\_viskers)

make\_box.place(relx=0.7, rely=0.1)

make\_box.pack()

button\_box = tk.Button(background,

text='Построить \n "Бокса-Вискера"',

bg=constants.style['plots\_button'])

button\_box.bind("<Button-1>", lambda \*args: draw\_box(root, pane))

button\_box.place(relx=0.0, rely=0.7, relheight=0.1, relwidth=0.2)

choice\_scatter = (

"Фильтры для диаграммы рассеяния",

"Высота - Смерти - Индекс взрывоопасности", "Высота - Смерти - Тип")

CHOSEN\_VALUE5 = tk.StringVar(value='Фильтры для диаграммы рассеяния')

make\_scatter = tk.OptionMenu(background, CHOSEN\_VALUE5, \*choice\_scatter)

make\_scatter.place(relx=0.7, rely=0.1)

make\_scatter.pack()

button\_scatter = tk.Button(background,

text='Построить \n диаграмму \n рассеяния',

bg=constants.style['plots\_button'])

button\_scatter.bind("<Button-1>", lambda \*args: draw\_scatter(root, pane))

button\_scatter.place(relx=0.4, rely=0.7, relheight=0.1, relwidth=0.2)

choice\_hist = ("Фильтры для гистограммы", "Высота - Количество смертей", "Высота - Ущерб")

CHOSEN\_VALUE6 = tk.StringVar(value='Фильтры для гистограммы')

make\_hist = tk.OptionMenu(background, CHOSEN\_VALUE6, \*choice\_hist)

make\_hist.place(relx=0.8, rely=0.1)

make\_hist.pack()

button\_scatter = tk.Button(background,

text='Построить гистограмму', bg=constants.style['plots\_button'])

button\_scatter.bind("<Button-1>", lambda \*args: draw\_hist(root, pane))

button\_scatter.place(relx=0.8, rely=0.7, relheight=0.1, relwidth=0.2)

background.pack(side="top", fill="both", expand=True, padx=2, pady=3)

def draw\_graph(root: tk.Tk, pane: ttk.Panedwindow):

"""

Автор: Ковязин В.

Цель: Функция для построения линейного графика

Вход: главное окно, растягивающийся виджет

Выход: Нет (файл)

"""

global CHOSEN\_VALUE1

win = tk.Toplevel(root)

win.title("График")

win.geometry("600x500+500+200")

background = tk.Frame(win, bg=constants.style['bg'])

background.place(x=0, y=0, relwidth=1, relheight=1)

name\_lab = tk.Label(background, text='График\n' + CHOSEN\_VALUE1.get(), bg=constants.style['bg'])

name\_lab.place(relx=0.01, rely=0.01, relwidth=0.95, relheight=0.1)

if CHOSEN\_VALUE1.get() == 'Фильтры для линейного графика':

mb.showerror("Ошибка!", "Сначала выберите фильтр! (графики)")

win.destroy()

elif CHOSEN\_VALUE1.get() == 'Год - Средняя смертность':

fig = plt.Figure(figsize=(7, 5), dpi=100)

ax1 = fig.add\_subplot(1, 1, 1)

# win.fillna(0)

hand\_base.bd = hand\_base.bd.fillna(0)

graph = FigureCanvasTkAgg(fig, background)

graph.get\_tk\_widget().pack(side=tk.LEFT, fill=tk.BOTH)

df1 = hand\_base.bd[["Year", "DEATHS"]].groupby("Year").mean()

df1.plot(kind='line', legend=True,

ax=ax1, color='r', title="График \n Год-Средняя смертность")

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save('../Graphics/year\_deaths\_graph.png'))

def save(string):

"""

Автор: Ковязин В.

Цель: Сохранение в файл

Вход: Путь формата строка

Выход: нет

"""

fig.savefig(string)

mb.showinfo("Инфо!", "Картинка сохранена")

button\_save.place(relx=0.1, rely=0.0, relheight=0.0495, relwidth=0.22)

def draw\_diagram(root: tk.Tk, pane: ttk.Panedwindow):

"""

Автор: Баканов Г.

Цель: Функция для построения столбчатой диаграмммы

Вход: главное окно, растягивающийся виджет

Выход: Нет (файл)

"""

global CHOSEN\_VALUE2

win = tk.Toplevel(root)

win.title("Диаграммы")

win.geometry("600x500+500+200")

background = tk.Frame(win, bg=constants.style['bg'])

background.place(x=0, y=0, relwidth=1, relheight=1)

name\_lab = tk.Label(background,

text='Диаграмма\n' + CHOSEN\_VALUE2.get(),

bg=constants.style['bg'])

name\_lab.place(relx=0.01, rely=0.01, relwidth=0.95, relheight=0.1)

if CHOSEN\_VALUE2.get() == 'Фильтры для столбчатой диаграммы':

mb.showerror("Ошибка!", "Сначала выберите фильтр (столбчатые диаграммы)!")

win.destroy()

elif CHOSEN\_VALUE2.get() == 'Тип вулкана - Средняя смертность':

bd = tb.type\_deaths\_mean()

fig = plt.figure(figsize=(7, 5), dpi=70)

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

fig.suptitle('Тип вулкана -\n Средняя смертность')

# fig.set\_figwidth(3)

bd.plot(kind='bar', ax=ax, y='Средняя смертность', x='Type', rot=45, fontsize=9)

# тут он вставка в интерфейс

CANVAS = FigureCanvasTkAgg(fig, background)

CANVAS.draw()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# toolbar = plt.NavigationToolbar2Tk(CANVAS\_1, background)

# toolbar.update()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save('../Graphics/type\_deaths\_bar.png'))

def save(string):

"""

Автор: Баканов Г.

Цель: Сохранение в файл

Вход: Путь формата строка

Выход: нет

"""

fig.savefig(string)

mb.showinfo("Инфо!", "Картинка сохранена")

button\_save.place(relx=0.1, rely=0.0, relheight=0.0495, relwidth=0.22)

elif CHOSEN\_VALUE2.get() == 'Страна - Средняя смертность':

bd = tb.country\_deaths\_mean()

fig = plt.figure(figsize=(4, 5), dpi=70)

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

fig.suptitle('Страна -\n Средняя смертность')

bd.plot(kind='bar', ax=ax, y='Средняя смертность', x='Country', rot=45, fontsize=9)

# тут он вставка в интерфейс

CANVAS = FigureCanvasTkAgg(fig, background)

CANVAS.draw()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# toolbar = NavigationToolbar2Tk(CANVAS\_1, background)

# toolbar.update()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save('../Graphics/country\_deaths\_bar.png'))

def save(string):

"""

Автор: Баканов Г.

Цель: Сохранение в файл

Вход: Путь формата строка

Выход: нет

"""

fig.savefig(string)

mb.showinfo("Инфо!", "Картинка сохранена")

button\_save.place(relx=0.1, rely=0.0, relheight=0.0495, relwidth=0.22)

elif CHOSEN\_VALUE2.get() == 'Страна - Средняя высота вулкана':

bd = tb.country\_elevation\_mean()

fig = plt.figure(figsize=(4, 5), dpi=70)

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

fig.suptitle('Страна -\n Средняя высота вулкана')

bd.plot(kind='bar', ax=ax, y='Средняя высота вулкана', x='Country', rot=45, fontsize=9)

# тут он вставка в интерфейс

CANVAS = FigureCanvasTkAgg(fig, background)

CANVAS.draw()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# toolbar = NavigationToolbar2Tk(CANVAS\_1, background)

# toolbar.update()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save('../Graphics/country\_elevation\_bar.png'))

def save(string):

"""

Автор: Баканов Г.

Цель: Сохранение в файл

Вход: Путь формата строка

Выход: нет

"""

fig.savefig(string)

mb.showinfo("Инфо!", "Картинка сохранена")

button\_save.place(relx=0.1, rely=0.0, relheight=0.0495, relwidth=0.22)

elif CHOSEN\_VALUE2.get() == 'Расположение - Средняя смертность':

bd = tb.location\_deaths\_mean()

fig = plt.figure(figsize=(4, 5), dpi=70)

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

fig.suptitle('Расположение -\n Средняя смертность')

bd.plot(kind='bar', ax=ax, y='Средняя смертность', x='Location', rot=45, fontsize=9)

# тут он вставка в интерфейс

CANVAS = FigureCanvasTkAgg(fig, background)

CANVAS.draw()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# toolbar = NavigationToolbar2Tk(CANVAS\_1, background)

# toolbar.update()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save('../Graphics/location\_deaths\_bar.png'))

def save(string):

"""

Автор: Баканов Г.

Цель: Сохранение в файл

Вход: Путь формата строка

Выход: нет

"""

fig.savefig(string)

mb.showinfo("Инфо!", "Картинка сохранена")

button\_save.place(relx=0.1, rely=0.0, relheight=0.0495, relwidth=0.22)

elif CHOSEN\_VALUE2.get() == 'Тип вулкана - Количество пропавших':

bd = tb.type\_missing\_num()

fig = plt.figure(figsize=(4, 5), dpi=70)

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

fig.suptitle('Тип вулкана -\n Количество пострадавших')

bd.plot(kind='bar', ax=ax, y='Количество известных пропавших', x='Type', rot=45, fontsize=9)

# тут он вставка в интерфейс

CANVAS = FigureCanvasTkAgg(fig, background)

CANVAS.draw()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# toolbar = NavigationToolbar2Tk(CANVAS\_1, background)

# toolbar.update()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save('../Graphics/type\_missing\_bar.png'))

def save(string):

"""

Автор: Баканов Г.

Цель: Сохранение в файл

Вход: Путь формата строка

Выход: нет

"""

fig.savefig(string)

mb.showinfo("Инфо!", "Картинка сохранена")

button\_save.place(relx=0.1, rely=0.0, relheight=0.0495, relwidth=0.22)

def elevation\_type():

"""

Автор: Подкопаева П.

Цель: Создание dataframe по фильтрам

Вход: нет

Выход: dataframe

"""

General = pd.DataFrame(hand\_base.bd)

General['Elevation'] = General['Elevation'].apply(pd.to\_numeric, errors='coerce')

rat = General.groupby(['Type']).agg({'Elevation': "mean"})

rat.rename(columns={'Type': 'Тип вулкана', 'Elevation': 'Высота вулкана'}, inplace=True)

rat = rat.reset\_index()

rat = rat.round({'Elevation': 2})

# print(rat.fillna(0))

return rat.fillna(0)

def draw\_box(root: tk.Tk, pane: ttk.Panedwindow):

"""

Автор: Подкопаева П., Баканов Г.

Цель: Построение диаграммы размаха Бокса-Вискера

Вход: главное окно, растягивающийся виджет

Выход: нет (файл)

"""

global CHOSEN\_VALUE4

win = tk.Toplevel(root)

win.title("Диаграммы размаха (Бокса-Вискера)")

win.geometry("600x500+500+200")

background = tk.Frame(win, bg=constants.style['bg'])

background.place(x=0, y=0, relwidth=1, relheight=1)

name\_lab = tk.Label(background,

text='Диаграмма размаха\n' + CHOSEN\_VALUE4.get(),

bg=constants.style['bg'])

name\_lab.place(relx=0.01, rely=0.01, relwidth=2, relheight=0.1)

if CHOSEN\_VALUE4.get() == 'Фильтры для диаграммы Бокса-Вискера':

mb.showerror("Ошибка!", "Сначала выберите фильтр для диаграммы Бокса-Вискера!")

win.destroy()

elif CHOSEN\_VALUE4.get() == 'Высота - Тип':

type1 = hand\_base.bd.loc[hand\_base.bd['Type'] == 'Stratovolcano', 'Elevation'].values

type2 = hand\_base.bd.loc[hand\_base.bd['Type'] == 'Shield volcano', 'Elevation'].values

type3 = hand\_base.bd.loc[hand\_base.bd['Type'] == 'Caldera', 'Elevation'].values

type4 = hand\_base.bd.loc[hand\_base.bd['Type'] == 'Pyroclastic shield', 'Elevation'].values

type5 = hand\_base.bd.loc[hand\_base.bd['Type'] == 'Submarine volcano', 'Elevation'].values

type6 = hand\_base.bd.loc[hand\_base.bd['Type'] == 'Complex volcano', 'Elevation'].values

fig = plt.figure(figsize=(4, 5), dpi=70)

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

fig.suptitle('Высота - Тип')

ax.boxplot([type1, type2, type3, type4, type5, type6],

labels=['Stratovolcano', 'Shield \n volcano', 'Caldera', 'Pyroclastic \n shield',

'Submarine \n volcano', 'Complex \n volcano'])

# тут вставка в интерфейс

CANVAS = FigureCanvasTkAgg(fig, background)

CANVAS.draw()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save('../Graphics/type\_elevation\_box.png'))

def save(string):

"""

Автор: Подкопаева П.

Цель: Сохранение в файл

Вход: Путь формата строка

Выход: нет

"""

fig.savefig(string)

mb.showinfo("Инфо!", "Картинка сохранена")

button\_save.place(relx=0.1, rely=0.0, relheight=0.0495, relwidth=0.22)

def draw\_pie(root: tk.Tk, pane: ttk.Panedwindow):

"""

Автор: Ковязин В., Подкопаева П.

Цель: Построение круговой диаграммы

Вход: главное окно, растягивающийся виджет

Выход: нет (файл)

"""

global CHOSEN\_VALUE3

win = tk.Toplevel(root)

win.title("Круговая диаграмма")

win.geometry("600x500+500+200")

background = tk.Frame(win, bg=constants.style['bg'])

background.place(x=0, y=0, relwidth=1, relheight=1)

name\_lab = tk.Label(background,

text='Круговая диаграмма\n' + CHOSEN\_VALUE3.get(),

bg=constants.style['bg'])

name\_lab.place(relx=0.01, rely=0.01, relwidth=0.95, relheight=0.1)

if CHOSEN\_VALUE3.get() == 'Фильтры для круговой диаграммы':

mb.showerror("Ошибка!", "Сначала выберите фильтр для круговой диаграммы!")

win.destroy()

elif CHOSEN\_VALUE3.get() == 'Тип - Средняя смертность':

dictionary = {}

dictionary['others'] = 0

for x in set(hand\_base.bd.Type):

deaths = hand\_base.bd['DEATHS'][hand\_base.bd['Type'] == x].mean()

if deaths > 200:

dictionary[x] = deaths

else:

dictionary['others'] = deaths

fig = plt.figure(figsize=(4, 5), dpi=70)

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

fig.suptitle('Тип - Средняя смертность')

ax.pie(dictionary.values(), labels=dictionary.keys(), autopct='%1.1f%%')

# тут он вставка в интерфейс

CANVAS = FigureCanvasTkAgg(fig, background)

CANVAS.draw()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# toolbar = NavigationToolbar2Tk(CANVAS\_1, background)

# toolbar.update()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save('../Graphics/type\_deaths\_pie.png'))

def save(string):

"""

Автор: Ковязин В.

Цель: Сохранение в файл

Вход: Путь формата строка

Выход: нет

"""

fig.savefig(string)

mb.showinfo("Инфо!", "Картинка сохранена")

button\_save.place(relx=0.1, rely=0.0, relheight=0.0495, relwidth=0.22)

elif CHOSEN\_VALUE3.get() == 'Страна - Средняя смертность':

dictionary = {}

dictionary['others'] = 0

for x in set(hand\_base.bd.Country):

deaths = hand\_base.bd['DEATHS'].fillna(0)[hand\_base.bd['Country'] == x].mean()

if deaths > 500:

dictionary[x] = deaths

else:

dictionary['others'] = deaths

fig = plt.figure(figsize=(4, 5), dpi=70)

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

fig.suptitle('Страна - Средняя смертность')

ax.pie(dictionary.values(), labels=dictionary.keys(), autopct='%1.1f%%')

# тут он вставка в интерфейс

CANVAS = FigureCanvasTkAgg(fig, background)

CANVAS.draw()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# toolbar = NavigationToolbar2Tk(CANVAS\_1, background)

# toolbar.update()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save('../Graphics/country\_deaths\_pie.png'))

def save(string):

"""

Автор: Подкопаева П.

Цель: Сохранение в файл

Вход: Путь формата строка

Выход: нет

"""

fig.savefig(string)

mb.showinfo("Инфо!", "Картинка сохранена")

button\_save.place(relx=0.1, rely=0.0, relheight=0.0495, relwidth=0.22)

elif CHOSEN\_VALUE3.get() == 'Тип - Ранения':

dictionary = {}

dictionary['others'] = 0

for x in set(hand\_base.bd.Type):

deaths = hand\_base.bd['INJURIES'][hand\_base.bd['Type'] == x].sum()

if deaths > 200:

dictionary[x] = deaths

fig = plt.figure(figsize=(4, 5), dpi=70)

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

fig.suptitle('Тип - Ранения')

ax.pie(dictionary.values(), labels=dictionary.keys(), autopct='%1.1f%%')

# тут он вставка в интерфейс

CANVAS = FigureCanvasTkAgg(fig, background)

CANVAS.draw()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# toolbar = NavigationToolbar2Tk(CANVAS\_1, background)

# toolbar.update()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save('../Graphics/type\_injuries\_pie.png'))

def save(string):

"""

Автор: Ковязин В.

Цель: Сохранение в файл

Вход: Путь формата строка

Выход: нет

"""

fig.savefig(string)

mb.showinfo("Инфо!", "Картинка сохранена")

button\_save.place(relx=0.1, rely=0.0, relheight=0.0495, relwidth=0.22)

def draw\_scatter(root: tk.Tk, pane: ttk.Panedwindow):

"""

Автор: Баканов Г., Подкопаева П.

Цель: Построение диаграммы рассеяния

Вход: главное окно, растягивающийся виджет

Выход: нет

"""

global CHOSEN\_VALUE5

if CHOSEN\_VALUE5.get() == 'Фильтры для диаграммы рассеяния':

mb.showerror("Ошибка!", "Сначала выберите фильтр для диаграммы рассеяния")

elif CHOSEN\_VALUE5.get() == 'Высота - Смерти - Индекс взрывоопасности':

win = tk.Toplevel(root)

win.title("Диаграммы рассеяния")

win.geometry("600x500+500+200")

background = tk.Frame(win, bg=constants.style['bg'])

background.place(x=0, y=0, relwidth=1, relheight=1)

name\_lab = tk.Label(background,

text='Диаграмма рассеяния\n' + CHOSEN\_VALUE5.get(),

bg=constants.style['bg'])

name\_lab.place(relx=0.01, rely=0.01, relwidth=0.95, relheight=0.1)

fig = plt.figure(figsize=(4, 5), dpi=70)

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

fig.suptitle('Высота - Смерти - Индекс \n взрывоопасности')

flag1 = flag2 = flag3 = flag4 = flag5 = flag6 = flag7 = flag8 = False

for n in range(0, 808):

if (hand\_base.bd['VEI'][n] == 7.0) and (flag1 is False):

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='red', label="7.0")

flag1 = True

elif hand\_base.bd['VEI'][n] == 7.0:

ax.scatter(hand\_base.bd['Elevation'][n], hand\_base.bd['DEATHS'][n], color='red')

elif hand\_base.bd['VEI'][n] == 6.0 and (flag2 is False):

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='blue', label="6.0")

flag2 = True

elif hand\_base.bd['VEI'][n] == 6.0:

ax.scatter(hand\_base.bd['Elevation'][n], hand\_base.bd['DEATHS'][n], color='blue')

elif hand\_base.bd['VEI'][n] == 5.0 and (flag3 is False):

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='green', label="5.0")

flag3 = True

elif hand\_base.bd['VEI'][n] == 5.0:

ax.scatter(hand\_base.bd['Elevation'][n], hand\_base.bd['DEATHS'][n], color='green')

elif hand\_base.bd['VEI'][n] == 4.0 and (flag4 is False):

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='#FF69B4', label="4.0")

flag4 = True

elif hand\_base.bd['VEI'][n] == 4.0:

ax.scatter(hand\_base.bd['Elevation'][n], hand\_base.bd['DEATHS'][n], color='#FF69B4')

elif hand\_base.bd['VEI'][n] == 3.0 and flag5 is False:

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='#00FFFF', label="3.0")

flag5 = True

elif hand\_base.bd['VEI'][n] == 3.0:

ax.scatter(hand\_base.bd['Elevation'][n], hand\_base.bd['DEATHS'][n], color='#00FFFF')

elif hand\_base.bd['VEI'][n] == 2.0 and flag6 is False:

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='#FF7F50', label="2.0")

flag6 = True

elif hand\_base.bd['VEI'][n] == 2.0:

ax.scatter(hand\_base.bd['Elevation'][n], hand\_base.bd['DEATHS'][n], color='#FF7F50')

elif hand\_base.bd['VEI'][n] == 1.0 and flag7 is False:

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='#7B68EE', label="1.0")

flag7 = True

elif hand\_base.bd['VEI'][n] == 1.0 and flag7:

ax.scatter(hand\_base.bd['Elevation'][n], hand\_base.bd['DEATHS'][n], color='#7B68EE')

elif flag8 is False:

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='#008080', label="Others")

flag8 = True

else:

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='#008080')

CANVAS = FigureCanvasTkAgg(fig, background)

CANVAS.draw()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

ax.legend(loc="upper right", title="", frameon=True)

ax.set\_xlabel('Высота')

ax.set\_ylabel('Количество смертей')

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>",

lambda \*args: save('../Graphics/type\_deaths\_elevation\_scat.png'))

def save(string):

"""

Автор: Подкопаева П.

Цель: Сохранение в файл

Вход: Путь формата строка

Выход: нет

"""

fig.savefig(string)

mb.showinfo("Инфо!", "Картинка сохранена")

button\_save.place(relx=0.1, rely=0.0, relheight=0.0495, relwidth=0.22)

elif CHOSEN\_VALUE5.get() == 'Высота - Смерти - Тип':

win = tk.Toplevel(root)

win.title("Диаграммы рассеяния")

win.geometry("600x500+500+200")

background = tk.Frame(win, bg=constants.style['bg'])

background.place(x=0, y=0, relwidth=1, relheight=1)

name\_lab = tk.Label(background,

text='Диаграмма рассеяния\n' + CHOSEN\_VALUE5.get(),

bg=constants.style['bg'])

name\_lab.place(relx=0.01, rely=0.01, relwidth=0.95, relheight=0.1)

fig = plt.figure(figsize=(4, 5), dpi=70)

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

fig.suptitle('Высота - Смерти - Тип')

flag1 = flag2 = flag3 = flag4 = flag5 = flag6 = flag7 = flag8 = False

for n in range(0, 808):

if (hand\_base.bd['Type'][n] == 'Shield volcano') and (flag1 is False):

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='red', label="Shield volcano")

flag1 = True

elif hand\_base.bd['Type'][n] == 'Shield volcano':

ax.scatter(hand\_base.bd['Elevation'][n], hand\_base.bd['DEATHS'][n], color='red')

elif (hand\_base.bd['Type'][n] == 'Caldera') and (flag2 is False):

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='blue', label="Caldera")

flag2 = True

elif hand\_base.bd['Type'][n] == 'Caldera':

ax.scatter(hand\_base.bd['Elevation'][n], hand\_base.bd['DEATHS'][n], color='blue')

elif (hand\_base.bd['Type'][n] == 'Maar') and (flag3 is False):

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='green', label="Maar")

flag3 = True

elif hand\_base.bd['Type'][n] == 'Maar':

ax.scatter(hand\_base.bd['Elevation'][n], hand\_base.bd['DEATHS'][n], color='green')

elif (hand\_base.bd['Type'][n] == 'Lava dome') and (flag4 is False):

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='#FF69B4', label="Lava dome")

flag4 = True

elif hand\_base.bd['Type'][n] == 'Lava dome':

ax.scatter(hand\_base.bd['Elevation'][n], hand\_base.bd['DEATHS'][n], color='#FF69B4')

elif (hand\_base.bd['Type'][n] == 'Pyroclastic shield') and (flag5 is False):

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='#00FFFF', label="Pyroclastic shield")

flag5 = True

elif hand\_base.bd['Type'][n] == 'Pyroclastic shield':

ax.scatter(hand\_base.bd['Elevation'][n], hand\_base.bd['DEATHS'][n], color='#00FFFF')

elif (hand\_base.bd['Type'][n] == 'Subglacial volcano') and (flag6 is False):

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='#FF7F50', label="Subglacial volcano")

flag6 = True

elif hand\_base.bd['Type'][n] == 'Subglacial volcano':

ax.scatter(hand\_base.bd['Elevation'][n], hand\_base.bd['DEATHS'][n], color='#FF7F50')

elif (hand\_base.bd['Type'][n] == 'Complex volcano') and (flag7 is False):

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='#7B68EE', label="Complex volcano")

flag7 = True

elif hand\_base.bd['Type'][n] == 'Complex volcano':

ax.scatter(hand\_base.bd['Elevation'][n], hand\_base.bd['DEATHS'][n], color='#7B68EE')

elif flag8 is False:

ax.scatter(hand\_base.bd['Elevation'][n],

hand\_base.bd['DEATHS'][n], color='#008080', label='Others')

flag8 = True

else:

ax.scatter(hand\_base.bd['Elevation'][n], hand\_base.bd['DEATHS'][n], color='#008080')

CANVAS = FigureCanvasTkAgg(fig, background)

CANVAS.draw()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

ax.legend(loc="upper right", title="", frameon=True)

ax.set\_xlabel('Высота')

ax.set\_ylabel('Количество смертей')

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>",

lambda \*args: save('../Graphics/type\_deaths\_elevation\_scat.png'))

def save(string):

"""

Автор: Баканов Г.

Цель: Сохранение в файл

Вход: Путь формата строка

Выход: нет

"""

fig.savefig(string)

mb.showinfo("Инфо!", "Картинка сохранена")

button\_save.place(relx=0.1, rely=0.0, relheight=0.0495, relwidth=0.22)

def elevation\_deaths():

"""

Автор: Баканов Г.

Цель: Создание dataframe по фильтрам

Вход: нет

Выход: dataframe

"""

General = pd.DataFrame(hand\_base.bd)

General['DEATHS'] = General['DEATHS'].apply(pd.to\_numeric, errors='coerce')

rat = General.groupby(['Elevation']).agg({'DEATHS': "sum"})

rat.rename(columns={'Elevation': 'Высота вулкана',

'DEATHS': 'Количество смертей'}, inplace=True)

rat = rat.reset\_index()

rat = rat.round({'Number of deaths': 2})

# print(rat.fillna(0))

return rat.fillna(0)

def elevation\_damage():

"""

Автор: Баканов Г.

Цель: Создание dataframe по фильтрам

Вход: нет

Выход: dataframe

"""

General = pd.DataFrame(hand\_base.bd)

General['DAMAGE\_MILLIONS\_DOLLARS'] = General['DAMAGE\_MILLIONS\_DOLLARS'].apply(pd.to\_numeric,

errors='coerce')

rat = General.groupby(['Elevation']).agg({'DAMAGE\_MILLIONS\_DOLLARS': "sum"})

rat.rename(columns={'Elevation': 'Высота вулкана',

'DAMAGE\_MILLIONS\_DOLLARS': 'Ущерб'}, inplace=True)

rat = rat.reset\_index()

rat = rat.round({'Damage in million dollars': 2})

# print(rat.fillna(0))

return rat.fillna(0)

def draw\_hist(root: tk.Tk, pane: ttk.Panedwindow):

"""

Автор: Ковязин В.

Цель: Построение гистограммы

Вход: главное окно, растягивающийся виджет

Выход: нет

"""

global CHOSEN\_VALUE6

win = tk.Toplevel(root)

win.title("Гистограммы")

win.geometry("600x500+500+200")

background = tk.Frame(win, bg=constants.style['bg'])

background.place(x=0, y=0, relwidth=1, relheight=1)

def save(string):

"""

Автор: Ковязин В.

Цель: Сохранение в файл

Вход: Путь формата строка

Выход: нет

"""

fig.savefig(string)

mb.showinfo("Инфо!", "Картинка сохранена")

name\_lab = tk.Label(background, text='Гистограмма\n' + CHOSEN\_VALUE6.get(),

bg=constants.style['bg'])

name\_lab.place(relx=0.01, rely=0.01, relwidth=0.95, relheight=0.1)

if CHOSEN\_VALUE6.get() == 'Фильтры для гистограммы':

mb.showerror("Ошибка!", "Сначала выберите фильтр (гистограммы)!")

win.destroy()

if CHOSEN\_VALUE6.get() == 'Высота - Количество смертей':

bd = elevation\_deaths()

fig = plt.figure(figsize=(4, 5), dpi=70)

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

fig.suptitle('Высота - Количество смертей')

bd.plot(kind='hist', ax=ax, y='Количество смертей', x='Elevation', rot=45, fontsize=9)

# тут он вставка в интерфейс

CANVAS = FigureCanvasTkAgg(fig, background)

CANVAS.draw()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# toolbar = NavigationToolbar2Tk(CANVAS\_1, background)

# toolbar.update()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save('../Graphics/elevation\_deaths\_hist.png'))

button\_save.place(relx=0.1, rely=0.0, relheight=0.0495, relwidth=0.22)

if CHOSEN\_VALUE6.get() == 'Высота - Ущерб':

bd = elevation\_damage()

fig = plt.figure(figsize=(4, 5), dpi=70)

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

fig.suptitle('Высота - Ущерб')

bd.plot(kind='hist', ax=ax, y='Ущерб', x='Elevation', rot=45, fontsize=9)

# тут он вставка в интерфейс

CANVAS = FigureCanvasTkAgg(fig, background)

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# toolbar = NavigationToolbar2Tk(CANVAS\_1, background)

# toolbar.update()

CANVAS.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

button\_save = tk.Button(background, font=12, text='Сохранить')

button\_save.bind("<Button-1>", lambda \*args: save('../Graphics/elevation\_damage\_hist.png'))

button\_save.place(relx=0.1, rely=0.0, relheight=0.0495, relwidth=0.22)

def stat\_report(root: tk.Tk, target: str) -> tuple:

"""

Автор: Ковязин В.

Цель: Подсчёт количества вхождений атрибута target

Вход: главное окно, target типа строка

Выход: количество вхождений и значение процента

"""

freq = glob.current\_base[target].value\_counts()

whole = 0

for x in freq.values:

whole += x

return freq, whole