ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

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

**Руководство разработчика по работе с приложением:**

**«Футбольный агент».**

Разработчики:

Данилов Игорь Евгеньевич

Петросян Гурген Аликович

Хомин Максим Вячеславович

Руководитель:

*Полякова М. В.*

Москва 2024 г.

Оглавление

[**Требования к характеристикам компьютера и операционной системе** 2](#_Toc169213581)

[**Версии интерпретатора и используемых библиотек** 2](#_Toc169213582)

[**Инструкция по установке приложения** 2](#_Toc169213583)

[**Инструкция по запуску и настройке приложения** 3](#_Toc169213584)

[**Описание структур БД** 3](#_Toc169213585)

[**Структура каталогов** 3](#_Toc169213586)

[**Архитектура приложения** 3](#_Toc169213587)

[**Листинг** 4](#_Toc169213588)

[**Main.py** 4](#_Toc169213589)

[**script\_graphics.py** 18](#_Toc169213590)

[**script\_text.py** 22](#_Toc169213591)

# **Требования к характеристикам компьютера и операционной системе**

Наличие на компьютере интерпретатора «Python» (вне зависимости от среды разработки)

Операционная система – Windows 7, 8, 10.

# **Версии интерпретатора и используемых библиотек**

Интерпретатор – Python 3.7+.

Используемые библиотеки:

|  |  |
| --- | --- |
| **Библиотека** | **Версия** |
| contourpy | 1.2.1+ |
| cycler | 0.12.1+ |
| fonttools | 4.53.0+ |
| kiwisolver | 1.4.5++ |
| matplotlib | 3.9.0+ |
| numpy | 1.26.4+ |
| packaging | 24.1+ |
| pandas | 2.2.2+ |
| pillow | 10.3.0+ |
| pyparsing | 3.1.2+ |
| python-dateuti | 2.9.0.post0+ |
| pytz | 2024.1+ |
| six | 1.16.0+ |
| tzdata | 2024.1+ |

# **Инструкция по установке приложения**

Загрузка zip-архива. Создание виртуального окружения “python3 -m venv venv”, активация виртуального окружения “source venv/bin/activate”, “pip install requirements.txt”

# **Инструкция по запуску и настройке приложения**

Для запуска приложения требуется выполнить команду “python3 main.py” в каталоге Work/scripts. Дополнительных настроек не требуется.

# **Описание структур БД**

Используется две базы данных: база данных со статистикой игроков и база данных со статистикой футбольных клубов.

База данных

# **Структура каталогов**

|  |  |
| --- | --- |
| **Первый уровень** | **Назначение** |
| Work | Основной каталог, содержащий программные модули модулями |

# **Архитектура приложения**

|  |  |  |
| --- | --- | --- |
| data | work | Папка, хранящая базы данных |
| graphics | work | Папка, хранящая графические отчеты |
| output | work | Папка, хранящая текстовые отчеты |
| scripts | work | Папка, хранящая скрипты |
| **Модуль** | **Расположение** | **Функция** |
| main.py | work\scripts | запуска программы и работы с базой данных |
| script\_graphics.py | work\scripts | Построение графических отчетов |
| script\_text.py | work\scripts | Построение текстовых отчетов |

# **Листинг**

## **Main.py**

from tkinter import simpledialog, StringVar, OptionMenu, filedialog, messagebox  
from script\_graphics import \*  
from script\_text import \*  
from tkinter import ttk  
  
# Чтение данных  
player\_data = pd.read\_csv(r'../data\player\_premier\_league\_stats.csv', delimiter=";").to\_numpy()  
teams\_data = pd.read\_csv(r'../data/squad\_premier\_league\_stats.csv', delimiter=',').to\_numpy()  
  
df = pd.read\_csv('../data/player\_premier\_league\_stats.csv', delimiter=";")  
df\_teams = pd.read\_csv('../data/squad\_premier\_league\_stats.csv', delimiter=',')  
  
teams = [i for i in df\_teams["Squad"]]  
POSITIONS = ["FW", "MF", "DF"]  
  
# Создание директорий, если не существует  
output\_dir = "../graphics/"  
if not os.path.exists(output\_dir):  
 os.makedirs(output\_dir)  
output\_dir = "../output/"  
if not os.path.exists(output\_dir):  
 os.makedirs(output\_dir)  
  
  
def select\_teams\_and\_plot(plot\_function):  
 *""" Функция выбора двух команд для генерации заданного графика  
 Автор: Хомин Максим  
  
 Parameters  
 ----------  
 plot\_function - график, который необходимо сгенерировать  
 Returns  
 -------  
 None  
 """* def on\_submit():  
 team\_1 = team\_var\_1.get()  
 team\_2 = team\_var\_2.get()  
 positions = ['DF', 'MF', 'FW']  
 if plot\_function == plot\_clustered\_bar\_chart:  
 plot\_function(player\_data, team\_1, team\_2, positions)  
 else:  
 plot\_function(player\_data, team\_1, team\_2)  
 select\_window.destroy()  
  
 select\_window = tk.Toplevel(root)  
 select\_window.title("Выбор команд")  
  
 team\_var\_1 = StringVar(select\_window)  
 team\_var\_2 = StringVar(select\_window)  
  
 team\_var\_1.set(teams[0])  
 team\_var\_2.set(teams[1])  
  
 tk.Label(select\_window, text="Команда 1:").pack(pady=5)  
 team\_menu\_1 = OptionMenu(select\_window, team\_var\_1, \*teams)  
 team\_menu\_1.pack(pady=5)  
  
 tk.Label(select\_window, text="Команда 2:").pack(pady=5)  
 team\_menu\_2 = OptionMenu(select\_window, team\_var\_2, \*teams)  
 team\_menu\_2.pack(pady=5)  
  
 submit\_button = tk.Button(select\_window, text="Подтвердить", command=on\_submit)  
 submit\_button.pack(pady=10)  
  
  
def select\_teams\_and\_plot\_five(plot\_function):  
 *""" Функция выбора пяти команд для генерации заданного графика  
 Автор: Петросян Гурген  
  
 Parameters  
 ----------  
 plot\_function - график, который необходимо сгенерировать  
 Returns  
 -------  
 None  
 """* def on\_submit():  
 team\_1 = team\_var\_1.get()  
 team\_2 = team\_var\_2.get()  
 team\_3 = team\_var\_3.get()  
 team\_4 = team\_var\_4.get()  
 team\_5 = team\_var\_5.get()  
 teams\_list = [team\_1, team\_2, team\_3, team\_4, team\_5]  
 if plot\_function == plot\_histogram:  
 plot\_histogram(teams\_data, teams\_list)  
 else:  
 plot\_boxplot(player\_data, teams\_list)  
 select\_window.destroy()  
  
 select\_window = tk.Toplevel(root)  
 select\_window.title("Выбор команд")  
  
 team\_var\_1 = StringVar(select\_window)  
 team\_var\_2 = StringVar(select\_window)  
 team\_var\_3 = StringVar(select\_window)  
 team\_var\_4 = StringVar(select\_window)  
 team\_var\_5 = StringVar(select\_window)  
  
 team\_var\_1.set(teams[0])  
 team\_var\_2.set(teams[1])  
 team\_var\_3.set(teams[2])  
 team\_var\_4.set(teams[3])  
 team\_var\_5.set(teams[4])  
  
 tk.Label(select\_window, text="Команда 1:").pack(pady=5)  
 team\_menu\_1 = OptionMenu(select\_window, team\_var\_1, \*teams)  
 team\_menu\_1.pack(pady=5)  
  
 tk.Label(select\_window, text="Команда 2:").pack(pady=5)  
 team\_menu\_2 = OptionMenu(select\_window, team\_var\_2, \*teams)  
 team\_menu\_2.pack(pady=5)  
  
 tk.Label(select\_window, text="Команда 3:").pack(pady=5)  
 team\_menu\_2 = OptionMenu(select\_window, team\_var\_3, \*teams)  
 team\_menu\_2.pack(pady=5)  
  
 tk.Label(select\_window, text="Команда 4:").pack(pady=5)  
 team\_menu\_2 = OptionMenu(select\_window, team\_var\_4, \*teams)  
 team\_menu\_2.pack(pady=5)  
  
 tk.Label(select\_window, text="Команда 5:").pack(pady=5)  
 team\_menu\_2 = OptionMenu(select\_window, team\_var\_5, \*teams)  
 team\_menu\_2.pack(pady=5)  
  
 submit\_button = tk.Button(select\_window, text="Подтвердить", command=on\_submit)  
 submit\_button.pack(pady=10)  
  
  
# Текстовый отчет №1  
def select\_position\_and\_age():  
 *""" Функция выбора позиции игрока и ввода максимального возраста для генерации голов  
 Автор: Данилов Игорь  
  
 Returns  
 -------  
 None  
 """* def on\_submit():  
 pos = str(position\_var.get())  
 max\_age = int(age\_entry.get())  
 goals(df, 'Pos', pos, 'Age', max\_age)  
 select\_window.destroy()  
  
 select\_window = tk.Toplevel()  
 select\_window.title("Выбор позиции и возраста")  
  
 positions = ['DF', 'MF', 'FW']  
  
 position\_var = StringVar(select\_window)  
 position\_var.set(positions[0])  
  
 tk.Label(select\_window, text="Выберите позицию:").pack(pady=5)  
 position\_menu = OptionMenu(select\_window, position\_var, \*positions)  
 position\_menu.pack(pady=5)  
  
 tk.Label(select\_window, text="Введите максимальный возраст игрока:").pack(pady=5)  
 age\_entry = tk.Entry(select\_window)  
 age\_entry.pack(pady=5)  
  
 submit\_button = tk.Button(select\_window, text="Подтвердить", command=on\_submit)  
 submit\_button.pack(pady=10)  
  
 select\_window.mainloop()  
  
  
# Текстовый отчет №3  
def select\_team\_and\_run\_xAG():  
 *""" Функция выбора команды и вызова функции xAG с выбранными параметрами  
 Автор: Петросян Гурген  
  
 Returns  
 -------  
 None  
 """* def on\_submit():  
 squad = squad\_var.get()  
 xAG(df, 'Squad', squad)  
 select\_window.destroy()  
  
 select\_window = tk.Toplevel()  
 select\_window.title("Выбор команды для xAG")  
  
 squads = df['Squad'].unique()  
  
 squad\_var = StringVar(select\_window)  
 squad\_var.set(squads[0])  
  
 tk.Label(select\_window, text="Выберите команду:").pack(pady=5)  
 squad\_menu = OptionMenu(select\_window, squad\_var, \*squads)  
 squad\_menu.pack(pady=5)  
  
 submit\_button = tk.Button(select\_window, text="Подтвердить", command=on\_submit)  
 submit\_button.pack(pady=10)  
  
 select\_window.mainloop()  
  
  
# Текстовый отчет №4  
def select\_aggregation\_column\_and\_create\_pivot\_table():  
 *""" Функция выбора столбца для агрегации и вызова функции create\_pivot\_table с выбранными параметрами  
 Автор: Хомин Максим  
  
 Returns  
 -------  
 None  
 """* def on\_submit():  
 variable = variable\_var.get()  
 create\_pivot\_table(df, 'Squad', 'Pos', variable, 'sum')  
 select\_window.destroy()  
  
 select\_window = tk.Toplevel()  
 select\_window.title("Выбор столбца для агрегации")  
  
 variables = ['Goals', 'Assist', '90s\_played', 'xG', 'xAG']  
  
 variable\_var = StringVar(select\_window)  
 variable\_var.set(variables[0])  
  
 tk.Label(select\_window, text="Выберите столбец для агрегации:").pack(pady=5)  
 variable\_menu = OptionMenu(select\_window, variable\_var, \*variables)  
 variable\_menu.pack(pady=5)  
  
 submit\_button = tk.Button(select\_window, text="Подтвердить", command=on\_submit)  
 submit\_button.pack(pady=10)  
  
 select\_window.mainloop()  
  
  
# Функции для работы с базой данных команд  
def add\_object\_to\_database\_squad():  
 *""" Функция для добавления объекта в базу данных  
 Автор: Хомин Максим  
  
 Returns  
 -------  
 None  
 """* # Запрос информации об объекте у пользователя с помощью интерфейса  
 df = pd.read\_csv('../data/squad\_premier\_league\_stats.csv', delimiter=",")  
 new\_object\_data = {}  
 root = tk.Tk()  
 root.withdraw() # Скрыть основное окно  
  
 for column in df.columns:  
 new\_value = simpledialog.askstring("Введите значение", f"Введите значение для столбца '{column}': ")  
 new\_object\_data[column] = new\_value  
  
 # Добавление нового объекта в базу данных  
 df.loc[len(df)] = new\_object\_data  
  
 # Сохранение изменений обратно в файл  
 df.to\_csv('../data/squad\_premier\_league\_stats.csv', index=False, sep=",")  
  
  
def delete\_object\_from\_database\_squad():  
 *""" Функция для удаления объекта из базы данных  
 Автор: Хомин Максим  
  
 Returns  
 -------  
 None  
 """* # Запрос информации о команде у пользователя с помощью интерфейса  
 df = pd.read\_csv('../data/squad\_premier\_league\_stats.csv', delimiter=",")  
 team\_name = simpledialog.askstring("Введите название команды", "Введите название команды для удаления: ")  
  
 # Удаление строки игрока из базы данных  
 df = df[df['Squad'] != team\_name]  
  
 # Сохранение изменений обратно в файл  
 df.to\_csv('../data/squad\_premier\_league\_stats.csv', index=False, sep=",")  
  
  
def edit\_value\_in\_database\_squad():  
 *""" Функция для изменения объекта в базе данных  
 Автор: Хомин Максим  
  
 Returns  
 -------  
 None  
 """* # Запрос информации о команде у пользователя с помощью интерфейса  
 df = pd.read\_csv('../data/squad\_premier\_league\_stats.csv', delimiter=",")  
 team\_name = simpledialog.askstring("Введите название команды", "Введите название команды для изменения значения: ")  
  
 # Запрос столбца и нового значения у пользователя  
 column\_name = simpledialog.askstring("Введите название столбца",  
 "Введите название столбца для изменения значения: ")  
 new\_value = simpledialog.askstring("Введите новое значение",  
 f"Введите новое значение для столбца '{column\_name}': ")  
  
 # Изменение значения для указанного игрока и столбца  
 mask = df['Squad'] == team\_name  
 df.loc[mask, column\_name] = new\_value  
  
 # Сохранение изменений обратно в файл  
 df.to\_csv('../data/squad\_premier\_league\_stats.csv', index=False, sep=",")  
  
  
def save\_data\_to\_bin\_squad():  
 *""" Функция для сохранения базы данных в двоичный формат  
 Автор: Петросян Гурген  
  
 Returns  
 -------  
 None  
 """* root = tk.Tk()  
 root.withdraw() # Скрыть основное окно  
 file\_path = filedialog.asksaveasfilename(defaultextension=".pkl", filetypes=[("Pickle files", "\*.pkl")])  
 if file\_path:  
 try:  
 df\_teams.to\_pickle(file\_path) # Сохранение базы данных в выбранный двоичный файл  
 messagebox.showinfo("Успех", "База данных успешно сохранена в двоичном формате.")  
 except Exception as e:  
 messagebox.showerror("Ошибка", f"Ошибка при сохранении базы данных в двоичном формате: {e}")  
  
  
def read\_data\_from\_bin\_squad():  
 *""" Функция для чтения базы данных из двоичного формата  
 Автор: Данилов Игорь  
  
 Returns  
 -------  
 None  
 """* root = tk.Tk()  
 root.withdraw() # Скрыть основное окно  
 file\_path = filedialog.askopenfilename(filetypes=[("Pickle files", "\*.pkl")])  
 if file\_path:  
 try:  
 df\_teams = pd.read\_pickle(file\_path) # Чтение базы данных из выбранного двоичного файла  
 df\_teams.to\_csv('../data/squad\_premier\_league\_stats.csv', index=False, sep=",") # Сохранение данных в формате CSV  
 messagebox.showinfo("Успех",  
 "База данных успешно восстановлена из двоичного файла и сохранена в формате CSV.")  
 except Exception as e:  
 messagebox.showerror("Ошибка", f"Ошибка при восстановлении базы данных из двоичного файла: {e}")  
  
# Функции для работы с базой данных игроков  
def add\_object\_to\_database():  
 *""" Функция для добавления объекта в базу данных  
 Автор: Хомин Максим  
  
 Returns  
 -------  
 None  
 """* # Запрос информации об объекте у пользователя с помощью интерфейса  
 df = pd.read\_csv('../data/player\_premier\_league\_stats.csv', delimiter=";")  
 new\_object\_data = {}  
 root = tk.Tk()  
 root.withdraw() # Скрыть основное окно  
  
 for column in df.columns:  
 new\_value = simpledialog.askstring("Введите значение", f"Введите значение для столбца '{column}': ")  
 new\_object\_data[column] = new\_value  
  
 # Добавление нового объекта в базу данных  
 df.loc[len(df)] = new\_object\_data  
  
 # Сохранение изменений обратно в файл  
 df.to\_csv('../data/player\_premier\_league\_stats.csv', index=False, sep=";")  
  
  
def delete\_object\_from\_database():  
 *""" Функция для удаления объекта из базы данных  
 Автор: Хомин Максим  
  
 Returns  
 -------  
 None  
 """* # Запрос информации об игроке у пользователя с помощью интерфейса  
 df = pd.read\_csv('../data/player\_premier\_league\_stats.csv', delimiter=";")  
 player\_name = simpledialog.askstring("Введите имя игрока", "Введите имя игрока для удаления: ")  
  
 # Удаление строки игрока из базы данных  
 df = df[df['Player'] != player\_name]  
  
 # Сохранение изменений обратно в файл  
 df.to\_csv('../data/player\_premier\_league\_stats.csv', index=False, sep=";")  
  
  
def edit\_value\_in\_database():  
 *""" Функция для изменения объекта в базе данных  
 Автор: Хомин Максим  
  
 Returns  
 -------  
 None  
 """* # Запрос информации об игроке у пользователя с помощью интерфейса  
 df = pd.read\_csv('../data/player\_premier\_league\_stats.csv', delimiter=";")  
 player\_name = simpledialog.askstring("Введите имя игрока", "Введите имя игрока для изменения значения: ")  
  
 # Запрос столбца и нового значения у пользователя  
 column\_name = simpledialog.askstring("Введите название столбца",  
 "Введите название столбца для изменения значения: ")  
 new\_value = simpledialog.askstring("Введите новое значение",  
 f"Введите новое значение для столбца '{column\_name}': ")  
  
 # Изменение значения для указанного игрока и столбца  
 mask = df['Player'] == player\_name  
 df.loc[mask, column\_name] = new\_value  
  
 # Сохранение изменений обратно в файл  
 df.to\_csv('../data/player\_premier\_league\_stats.csv', index=False, sep=";")  
  
  
def save\_data\_to\_bin():  
 *""" Функция для сохранения базы данных в двоичный формат  
 Автор: Петросян Гурген  
  
 Returns  
 -------  
 None  
 """* root = tk.Tk()  
 root.withdraw() # Скрыть основное окно  
 file\_path = filedialog.asksaveasfilename(defaultextension=".pkl", filetypes=[("Pickle files", "\*.pkl")])  
 if file\_path:  
 try:  
 df.to\_pickle(file\_path) # Сохранение базы данных в выбранный двоичный файл  
 messagebox.showinfo("Успех", "База данных успешно сохранена в двоичном формате.")  
 except Exception as e:  
 messagebox.showerror("Ошибка", f"Ошибка при сохранении базы данных в двоичном формате: {e}")  
  
  
def read\_data\_from\_bin():  
 *""" Функция для чтения базы данных из двоичного формата  
 Автор: Данилов Игорь  
  
 Returns  
 -------  
 None  
 """* root = tk.Tk()  
 root.withdraw() # Скрыть основное окно  
 file\_path = filedialog.askopenfilename(filetypes=[("Pickle files", "\*.pkl")])  
 if file\_path:  
 try:  
 df = pd.read\_pickle(file\_path) # Чтение базы данных из выбранного двоичного файла  
 df.to\_csv('../data/player\_premier\_league\_stats.csv', index=False, sep=";") # Сохранение данных в формате CSV  
 messagebox.showinfo("Успех",  
 "База данных успешно восстановлена из двоичного файла и сохранена в формате CSV.")  
 except Exception as e:  
 messagebox.showerror("Ошибка", f"Ошибка при восстановлении базы данных из двоичного файла: {e}")  
  
# Функции для вызова диалогов и запуска соответствующих функций  
def call\_plot\_clustered\_bar\_chart():  
 select\_teams\_and\_plot(plot\_clustered\_bar\_chart)  
  
  
def call\_plot\_histogram():  
 select\_teams\_and\_plot\_five(plot\_histogram)  
  
  
def call\_plot\_boxplot():  
 select\_teams\_and\_plot\_five(plot\_boxplot)  
  
  
def call\_plot\_scatter():  
 select\_teams\_and\_plot(plot\_scatter)  
  
  
def call\_edit\_database():  
 *""" Функция вызова редактирования базы данных игроков  
 Автор: Данилов Игорь  
  
 Returns  
 -------  
 None  
 """* edit\_window = tk.Toplevel()  
 edit\_window.title("Управление базой данных")  
  
 btn\_add = tk.Button(edit\_window, text="Добавить объект", command=add\_object\_to\_database)  
 btn\_add.pack(pady=10)  
  
 btn\_delete = tk.Button(edit\_window, text="Удалить объект", command=delete\_object\_from\_database)  
 btn\_delete.pack(pady=10)  
  
 btn\_edit = tk.Button(edit\_window, text="Редактировать объект", command=edit\_value\_in\_database)  
 btn\_edit.pack(pady=10)  
  
 btn\_save = tk.Button(edit\_window, text="Сохранить справочник в двоичном формате", command=save\_data\_to\_bin)  
 btn\_save.pack(pady=10)  
  
 btn\_load = tk.Button(edit\_window, text="Считать справочник из двоичного формата", command=read\_data\_from\_bin)  
 btn\_load.pack(pady=10)  
  
 edit\_window.mainloop()  
  
def call\_edit\_database\_squad():  
 *""" Функция вызова редактирования базы данных команд  
 Автор: Данилов Игорь  
  
 Returns  
 -------  
 None  
 """* edit\_window = tk.Toplevel()  
 edit\_window.title("Управление базой данных команд")  
  
 btn\_add = tk.Button(edit\_window, text="Добавить объект", command=add\_object\_to\_database\_squad)  
 btn\_add.pack(pady=10)  
  
 btn\_delete = tk.Button(edit\_window, text="Удалить объект", command=delete\_object\_from\_database\_squad)  
 btn\_delete.pack(pady=10)  
  
 btn\_edit = tk.Button(edit\_window, text="Редактировать объект", command=edit\_value\_in\_database\_squad)  
 btn\_edit.pack(pady=10)  
  
 btn\_save = tk.Button(edit\_window, text="Сохранить справочник в двоичном формате", command=save\_data\_to\_bin\_squad)  
 btn\_save.pack(pady=10)  
  
 btn\_load = tk.Button(edit\_window, text="Считать справочник из двоичного формата", command=read\_data\_from\_bin\_squad)  
 btn\_load.pack(pady=10)  
  
 edit\_window.mainloop()  
  
  
# Создание окна и кнопок  
root = tk.Tk()  
root.title("Футбольный агент")  
  
  
# Кнопки для редактирования баз данных  
btn\_edit = tk.Button(root, text="Взаимодействие с базой данных игроков", command=call\_edit\_database)  
btn\_edit.pack(pady=10)  
  
btn\_edit = tk.Button(root, text="Взаимодействие с базой данных команд", command=call\_edit\_database\_squad)  
btn\_edit.pack(pady=10)  
  
# Кнопки для генерации графиков  
btn1 = tk.Button(root, text="Cравнение эффективности игроков разных позиций", command=call\_plot\_clustered\_bar\_chart)  
btn1.pack(pady=10)  
tooltip = ttk.Label(root,  
 text="Функция создаёт график, сравнивающий количество голов по позициям для двух выбранных команд")  
tooltip.pack()  
  
btn2 = tk.Button(root, text="Cредний возраст игроков", command=call\_plot\_histogram)  
btn2.pack(pady=10)  
tooltip = ttk.Label(root,  
 text="Функция создаёт гистограмму по среднему возрасту игроков пяти выбранных команд")  
tooltip.pack()  
  
btn3 = tk.Button(root, text="Статистика «Гол+пас»", command=call\_plot\_boxplot)  
btn3.pack(pady=10)  
tooltip = ttk.Label(root,  
 text="Функция создаёт категоризированную диаграмму по системе 'гол+пас' для игроков пяти выбранных команд")  
tooltip.pack()  
  
btn4 = tk.Button(root, text="Категоризированная диаграмма рассеивания", command=call\_plot\_scatter)  
btn4.pack(pady=10)  
tooltip = ttk.Label(root,  
 text="Функция создаёт категоризированную диаграмму рассеивания по голам и помощам для пяти выбранных команд")  
tooltip.pack()  
  
# Кнопки для генерации текстовых отчетов  
btn5 = tk.Button(root, text="Текстовый отчет №1", command=select\_position\_and\_age)  
btn5.pack(pady=10)  
tooltip = ttk.Label(root,  
 text="Функция создаёт текстовый отчет о количестве голов для игроков выбранной позиции младше выбранного возраста")  
tooltip.pack()  
  
btn6 = tk.Button(root, text="Текстовый отчет №2", command=statistics)  
btn6.pack(pady=10)  
tooltip = ttk.Label(root,  
 text="Функция создаёт текстовый статистический отчет для количественных переменных базы данных")  
tooltip.pack()  
  
btn7 = tk.Button(root, text="Текстовый отчет №3", command=select\_team\_and\_run\_xAG)  
btn7.pack(pady=10)  
tooltip = ttk.Label(root,  
 text="Функция создаёт текстовый отчет, сравнивающий помощи игроков выбранной команды с их xAG")  
tooltip.pack()  
  
btn8 = tk.Button(root, text="Сводная таблица", command=select\_aggregation\_column\_and\_create\_pivot\_table)  
btn8.pack(pady=10)  
tooltip = ttk.Label(root,  
 text="Функция создаёт сводную таблицу по позициям для каждой команды, пользователь может выбрать столбец для агрегации")  
tooltip.pack()  
  
root.mainloop()

## **script\_graphics.py**

import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd  
import os  
  
output\_dir = "../graphics/"  
  
  
def plot\_clustered\_bar\_chart(player\_data, TEAM\_1, TEAM\_2, POSITIONS):  
 *""" Функция генерации кластеризованной столбчатой диаграммы для двух команд,  
 сравнивающая суммарное количество голов игроков по позициям среди этих команд.  
 Автор: Данилов Игорь  
  
 Parameters  
 ----------  
 player\_data : pd.DataFrame - исходная таблица с данными об игроках  
 TEAM\_1 : str - название команды 1  
 TEAM\_2 : str - название команды 2  
 POSITIONS : list - позиции, по которым идёт сравнение  
 Returns  
 -------  
 None  
 """* goals\_1, goals\_2 = [0 for \_ in range(len(POSITIONS))], [0 for \_ in range(len(POSITIONS))]  
  
 for i in range(len(player\_data)):  
 player = player\_data[i]  
 if player[3] == TEAM\_1:  
 for j in range(len(POSITIONS)):  
 if POSITIONS[j] in player[2]:  
 goals\_1[j] += int(player[7])  
 if player[3] == TEAM\_2:  
 for j in range(len(POSITIONS)):  
 if POSITIONS[j] in player[2]:  
 goals\_2[j] += int(player[7])  
  
 bar\_width = 0.32  
 x = range(len(POSITIONS))  
  
 plt.bar(x, goals\_1, width=bar\_width, label=TEAM\_1)  
 plt.bar([i + bar\_width for i in x], goals\_2, width=bar\_width, label=TEAM\_2)  
  
 plt.xlabel('Позиции игроков')  
 plt.ylabel('Суммарное количество голов')  
 plt.title('Кластеризованная столбчатая диаграмма')  
 plt.xticks([i + bar\_width / 2 for i in x], POSITIONS)  
 plt.legend()  
  
 plt.savefig(os.path.join(output\_dir, "1.png"))  
 plt.show()  
  
  
def plot\_histogram(teams\_data, teams):  
 *""" Функция генерации категоризированной гистограммы, показывающей средний  
 возраст игроков выбранных команд с добавлением значений на график.  
 Автор: Хомин Максим  
  
 Parameters  
 ----------  
 teams\_data : pd.DataFrame - исходная таблица с данными о командах  
 teams : list - список выбранных команд  
 Returns  
 -------  
 None  
 """* average\_ages = []  
 for team in teams\_data:  
 if team[0] in teams:  
 average\_ages.append(team[1])  
  
 plt.bar(teams, average\_ages, color='skyblue')  
  
 plt.title('Категоризированная гистограмма')  
 plt.xlabel('Команды')  
 plt.ylabel('Средний возраст')  
  
 for i in range(len(teams)):  
 plt.text(i, average\_ages[i], str(average\_ages[i]), ha='center', va='bottom')  
  
 plt.savefig(os.path.join(output\_dir, "2.png"))  
 plt.show()  
  
  
def plot\_boxplot(player\_data, teams):  
 *""" Функция генерации категоризированной диаграммы (box-and-whiskers)  
 для параметра "гол+пас" игроков из выбранных команд  
 Автор: Петросян Гурген  
  
 Parameters  
 ----------  
 player\_data : pd.DataFrame - исходная таблица с данными об игроках  
 teams : list - список выбранных команд  
 Returns  
 -------  
 None  
 """* team\_indices = {team: idx for idx, team in enumerate(teams)}  
 poss\_by\_mass = [[] for \_ in range(len(teams))]  
  
 for player in player\_data:  
 for team in teams:  
 if player[3] == team:  
 team\_idx = team\_indices[team]  
 poss\_by\_mass[team\_idx].append(player[8])  
 break  
  
 data = {team: ages for team, ages in zip(teams[:5], poss\_by\_mass[:5])}  
  
 box\_data = [ages for team, ages in data.items()]  
  
 plt.boxplot(box\_data)  
 plt.xticks(range(1, len(data) + 1), data.keys())  
 plt.xlabel("Команды")  
 plt.ylabel("Гол+пас")  
 plt.title("Категоризированная диаграмма")  
  
 plt.savefig(os.path.join(output\_dir, "3.png"))  
 plt.show()  
  
  
def plot\_scatter(player\_data, TEAM, TEAM\_2):  
 *""" Функция генерации категоризированной диаграммы рассеивания для  
 двух выбранных команд, показывающей количество помощей игроков  
 Автор: Данилов Игорь  
  
 Parameters  
 ----------  
 player\_data : pd.DataFrame - исходная таблица с данными об игроках  
 TEAM : str - название команды 1  
 TEAM\_2 : str - название команды 2  
 Returns  
 -------  
 None  
 """* goals, assist, current\_player = [], [], []  
 goals\_2, assist\_2, current\_player\_2 = [], [], []  
  
 for i in range(len(player\_data)):  
 if player\_data[i, 3] == TEAM:  
 current\_player.append(player\_data[i, 0])  
 goals.append(player\_data[i, 7])  
 assist.append(player\_data[i, 8])  
 if player\_data[i, 3] == TEAM\_2:  
 current\_player\_2.append(player\_data[i, 0])  
 goals\_2.append(player\_data[i, 7])  
 assist\_2.append(player\_data[i, 8])  
  
 plt.scatter(goals, assist, label=TEAM)  
 plt.scatter(goals\_2, assist\_2, label=TEAM\_2)  
  
 plt.xlabel("Количество голов")  
 plt.ylabel("Количество ассистов")  
 plt.title('Категоризированная диаграмма рассеивания')  
  
 plt.legend()  
 plt.savefig(os.path.join(output\_dir, "4.png"))  
 plt.show()

## **script\_text.py**

import pandas as pd  
import tkinter as tk  
  
df = pd.read\_csv('../data/player\_premier\_league\_stats.csv', delimiter=";")  
df\_teams = pd.read\_csv('../data/squad\_premier\_league\_stats.csv', delimiter=",")  
  
  
def goals(df: pd.DataFrame, cndname: str, cndval: str, intname: str, intval: int) -> pd.DataFrame:  
 *""" Функция генерации отчетов о количестве голов для одного строкового  
 и одного целочисленного (макс.) критериев с выводом на экран  
 Автор: Хомин Максим  
  
 Parameters  
 ----------  
 df : pd.DataFrame - исходная таблица с данными об игроках  
 cndname : str - атрибут критерия  
 cndval : str - значение критерия  
 intname : str - атрибут критерия  
 intval : int - максимальное значение критерия  
 Returns  
 -------  
 filtered\_df : pd.DataFrame - отчет.  
 """* filtered\_df = df[(df[cndname] == cndval) & (df[intname] <= intval)][['Player', intname, 'Squad', 'Goals']]  
 filtered\_text = filtered\_df.to\_string(index=False)  
  
 display\_window = tk.Toplevel()  
 display\_window.title("Результаты")  
  
 text\_widget = tk.Text(display\_window, wrap="none")  
 text\_widget.insert('1.0', filtered\_text)  
 text\_widget.pack(side="left", fill="both", expand=True)  
  
 scrollbar\_y = tk.Scrollbar(display\_window, command=text\_widget.yview)  
 scrollbar\_y.pack(side="right", fill="y")  
 text\_widget.config(yscrollcommand=scrollbar\_y.set)  
  
 scrollbar\_x = tk.Scrollbar(display\_window, orient="horizontal", command=text\_widget.xview)  
 scrollbar\_x.pack(side="bottom", fill="x")  
 text\_widget.config(xscrollcommand=scrollbar\_x.set)  
  
 filtered\_df.to\_csv('../output/report1.txt', sep=';', index=False)  
 return filtered\_df  
  
  
def statistics() -> pd.DataFrame:  
 *"""  
 Функция для генерации отчетов статистики для основных количественных переменных  
 Автор: Петросян Гурген  
  
 Parameters  
 ----------  
 count : количество непустых (непропущенных) значений в столбце.  
 mean : среднее арифметическое значение для всех значений в столбце.  
 std : стандартное отклонение, которое измеряет разброс значений от среднего значения. Большое стандартное  
 отклонение указывает на большой разброс, а маленькое - на то, что значения сгруппированы близко к среднему.  
 min : минимальное значение в столбце.  
 25% (первый квартиль) : значение, ниже которого попадает 25% значений столбца. Это также называется 25-м процентилем.  
 50% (медиана) : значение, ниже которого попадает 50% значений столбца. Это также называется медианой.  
 75% (третий квартиль) : значение, ниже которого попадает 75% значений столбца. Это также называется 75-м процентилем.  
 max : максимальное значение в столбце.  
 Returns  
 -------  
 statistics\_df : pd.DataFrame - отчет с основными статистическими показателями  
 """* variables = ['Age', 'Goals', 'Match\_Play', '90s\_played', 'Assist', 'xG']  
  
 statistics\_df = df[variables].describe().T  
  
 display\_window = tk.Toplevel()  
 display\_window.title("Результаты")  
  
 text\_widget = tk.Text(display\_window, wrap="none")  
 text\_widget.insert('1.0', statistics\_df.to\_string())  
 text\_widget.pack(side="left", fill="both", expand=True)  
  
 scrollbar\_y = tk.Scrollbar(display\_window, command=text\_widget.yview)  
 scrollbar\_y.pack(side="right", fill="y")  
 text\_widget.config(yscrollcommand=scrollbar\_y.set)  
  
 scrollbar\_x = tk.Scrollbar(display\_window, orient="horizontal", command=text\_widget.xview)  
 scrollbar\_x.pack(side="bottom", fill="x")  
 text\_widget.config(xscrollcommand=scrollbar\_x.set)  
  
 statistics\_df.to\_csv('../output/report2.txt', sep=';', index=False)  
 return statistics\_df  
  
  
def xAG(df: pd.DataFrame, cndname\_1: str, cndval\_1: str) -> pd.DataFrame:  
 *""" Функция генерации отчетов о сравнении реальных ассистов игрока с его xAG для  
 одного строкового критерия  
 Автор: Данилов Игорь  
  
 Parameters  
 ----------  
 df : pd.DataFrame - исходная таблица с данными об игроках  
 cndname\_1 : str - атрибут критерия 1  
 cndval\_1 : str - значение критерия 1  
 Returns  
 -------  
 filtered\_df : pd.DataFrame - отчет.  
 """* filtered\_df = df[df[cndname\_1] == cndval\_1][['Player', '90s\_played', 'Assist', 'xAG']]  
 filtered\_df.to\_csv('../output/report3.txt', sep=';', index=False)  
  
 filtered\_text = filtered\_df.to\_string(index=False)  
  
 display\_window = tk.Toplevel()  
 display\_window.title("Результаты")  
  
 text\_widget = tk.Text(display\_window, wrap="none")  
 text\_widget.insert('1.0', filtered\_text)  
 text\_widget.pack(side="left", fill="both", expand=True)  
  
 scrollbar\_y = tk.Scrollbar(display\_window, command=text\_widget.yview)  
 scrollbar\_y.pack(side="right", fill="y")  
 text\_widget.config(yscrollcommand=scrollbar\_y.set)  
  
 scrollbar\_x = tk.Scrollbar(display\_window, orient="horizontal", command=text\_widget.xview)  
 scrollbar\_x.pack(side="bottom", fill="x")  
 text\_widget.config(xscrollcommand=scrollbar\_x.set)  
 return filtered\_df  
  
def create\_pivot\_table(df: pd.DataFrame, index\_col1: str, index\_col2: str, values\_col: str,  
 agg\_func: str) -> pd.DataFrame:  
 *"""  
 Функция для создания сводной таблицы на основе входных параметров  
 Автор: Хомин Максим  
  
 Parameters  
 ----------  
 df: pd.DataFrame - исходная таблица с данными  
 index\_col1: str - название первого качественного столбца для использования в качестве индекса  
 index\_col2: str - название второго качественного столбца для использования в качестве индекса  
 values\_col: str - название количественного столбца для агрегации  
 agg\_func: str - метод агрегации  
  
 Returns  
 -------  
 pivot\_table : pd.DataFrame - Сводная таблица на основе входных параметров  
 """* pivot\_table = pd.pivot\_table(df, values=values\_col, index=index\_col1, columns=index\_col2, aggfunc=agg\_func,  
 fill\_value=0)  
  
 display\_window = tk.Toplevel()  
 display\_window.title("Результаты")  
  
 text\_widget = tk.Text(display\_window, wrap="none")  
 text\_widget.insert('1.0', pivot\_table.to\_string())  
 text\_widget.pack(side="left", fill="both", expand=True)  
  
 scrollbar\_y = tk.Scrollbar(display\_window, command=text\_widget.yview)  
 scrollbar\_y.pack(side="right", fill="y")  
 text\_widget.config(yscrollcommand=scrollbar\_y.set)  
  
 scrollbar\_x = tk.Scrollbar(display\_window, orient="horizontal", command=text\_widget.xview)  
 scrollbar\_x.pack(side="bottom", fill="x")  
 text\_widget.config(xscrollcommand=scrollbar\_x.set)  
  
 pivot\_table.to\_csv('../output/report4.txt', sep=';', index=True)  
 return pivot\_table