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: «Анализ космических миссий»

Авторы:

Маркова Эвелина БИВ201

Тарасенко Илья БИВ201

Горбачева Валерия БИВ201

Москва 2021 г.

## Требования к характеристикам компьютера и операционной системе

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

Операционная система – Windows 7 и новее.

## Версии интерпретатора и используемых библиотек

Интерпретатор – Python 3.7 и новее.

Используемые библиотеки:

|  |  |
| --- | --- |
| **Библиотека** | **Версия** |
| pandas | 1.2.4 и новее |
| matplotlib | 3.4.2 и новее |
| numpy | 1.20.0 и новее |
| openpyxl | 3.0.7 и новее |
| tkinter | 8.6 и новее |

А также используются стандартные модули sys и os.

## Описание структуры базы данных

За основу была взята база данных Space\_corrected.csv, после измененная в space\_missions.csv:

Информация «Unnamed: 0» (Нумерация строки) – натуральное число, была удалена из начальной таблицы.  
Поле «Datum» (Дата) – строка, было разделено на поля «Year», «Month», «Day of Week», «Time in Min» и «Date».  
Из «Location» был выделен столбец «Country», после чего «Location» был удален

База данных Space\_missions.csv имеет поля:

Company Name (Название компании) – строка

Detail (Тип ракеты) – строка

Status Rocket (Статус ракеты) – строка

Rocket (Цена запуска ракеты в $млн) – вещественное число

Status Mission (Статус миссии) – строка

Year (Год) – натуральное число

Month (Месяц) – строка

Day of Week (День недели) – строка

Date (Число) – натуральное число

Time in Min (Время в минутах) – вещественное число

Country (Страна) – строка

## Описание интерфейса программы

Интерфейс программы подробно описан в руководстве пользователя

## Листинг основного скрипта и всех модулей

|  |  |
| --- | --- |
| **Модуль** | **Содержимое** |
| config.py  (файл, содержащий константы) | **import** os  path = os.path.abspath(os.path.join(os.getcwd(), **'..'**, **'Data'**)) path\_to\_csv = path + **'/Space\_Corrected.csv'** path\_to\_new\_csv = path + **'/space\_missions.csv'** path\_graph = os.path.abspath(os.path.join(os.getcwd(), **'..'**, **'Graphics'**)) path\_out = os.path.abspath(os.path.join(os.getcwd(), **'..'**, **'Output'**)) |
| gui.py  (создание интерфейса) | **from** tkinter **import** \* **import** tkinter **as** tk **import** tkinter.ttk **as** ttk **import** csv **from** tkinter **import** messagebox **import** sys  sys.path.append(**'../'**)  **from** Scripts.main **import** \* **from** Scripts.config **import** \* **from** Library.lib **import** \*   **def** main\_window():  *"""  Функция создает главное окно приложения и описывает функции удаления, вставки, изменения строк  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* window = Tk()  window.title(**"Python analysis"**)   df = reading(path\_to\_new\_csv)  df.drop(df.columns[0], axis=1, inplace=**True**)  tree = ttk.Treeview()   df\_col = df.columns.values.tolist()  tree[**"columns"**] = df\_col  counter = len(df)   **for** x **in** range(len(df\_col)):  tree.column(df\_col[x], width=100)  tree.heading(df\_col[x], text=df\_col[x])   row\_labels = df.index.tolist()   **for** i **in** range(counter):  tree.insert(**''**, i, text=row\_labels[i], values=df.iloc[i, :].tolist())   ysb = ttk.Scrollbar(orient=tk.VERTICAL, command=tree.yview)  xsb = ttk.Scrollbar(orient=tk.HORIZONTAL, command=tree.xview)  tree.configure(yscroll=ysb.set, xscroll=xsb.set)  ysb.pack(side=tk.RIGHT, fill=tk.Y)  xsb.pack(side=tk.TOP, fill=tk.X)   **def** save():  **with** open(path\_out + **'/New.csv'**, **'w'**, newline=**''**, encoding=**'utf-8'**) **as** file:  writer = csv.writer(file, delimiter=**','**)  **for** row\_id **in** tree.get\_children():  row = tree.item(row\_id)[**'values'**]  writer.writerow(row)  messagebox.showinfo(**'Сделано!'**, **'Вы можете открыть новую базу данных '  'в папке Output в файле под названием New.csv'**)   **def** delete():  *"""  Функция удаляет строку из базы данных  Входные данные: нет  Выходные данные: нет  Автор: Маркова Э.  """* item = tree.selection()[0] *## get selected item* tree.delete(item)   **def** add():  *"""  Функция вставляет новую строку в базу данных  Входные данные: нет  Выходные данные: нет  Автор: Маркова Э.  """* **def** clicked():  *"""  Функция вставляет новую строку в базу данных  Входные данные: нет  Выходные данные: нет  Автор: Маркова Э.  """* space\_missions = reading(path\_to\_csv)  to\_3nf(space\_missions)  company\_name = txt1.get()  detail = txt2.get()  status\_rocket = txt3.get()  rocket = txt4.get()  status\_mission = txt5.get()  year = txt6.get()  month = txt7.get()  dow = txt8.get()  date = txt9.get()  time\_in\_min = txt10.get()  country = txt11.get()  value = [company\_name, detail, status\_rocket, rocket, status\_mission, year, month, dow, date, time\_in\_min,  country]  tree.insert(**''**, tk.END, value=value)  df.loc[-1] = value   window1 = Tk()  window1.title(**"Insert"**)  lbl = Label(window1, text=**"Рекомендуется вводить данные на английском языке и в соответствии"  " со следующими советами."**)  lbl1 = Label(window1, text=**"Status Rocket вводится как либо StatusActive, либо StatusRetired."**)  lbl2 = Label(window1, text=**"Rocket представляет собой число типа float."**)  lbl3 = Label(window1, text=**"Status Mission вводится как либо Success, либо Failure, "  "либо Partial Failure, либо Prelaunch Failure."**)  lbl4 = Label(window1, text=**"Месяц рекомендуется вводить в формате Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, "  "Oct, Nov, Dec."**)  lbl5 = Label(window1, font=**'bold'**, text=**"Перед тем, как скачать новую базу данных, обязательно нажмите кнопку 'Сохранить'"**)  lbl5.place(x=10, y=10)  lbl.place(x=10, y=40)  lbl1.place(x=10, y=60)  lbl2.place(x=10, y=80)  lbl3.place(x=10, y=100)  lbl4.place(x=10, y=120)  lbl\_info1 = Label(window1, text=**"Company Name"**)  lbl\_info2 = Label(window1, text=**"Detail"**)  lbl\_info3 = Label(window1, text=**"Status Rocket"**)  lbl\_info4 = Label(window1, text=**"Rocket"**)  lbl\_info5 = Label(window1, text=**"Status Mission"**)  lbl\_info6 = Label(window1, text=**"Year"**)  lbl\_info7 = Label(window1, text=**"Month"**)  lbl\_info8 = Label(window1, text=**"Day of Week"**)  lbl\_info9 = Label(window1, text=**"Date"**)  lbl\_info10 = Label(window1, text=**"Time in Min"**)  lbl\_info11 = Label(window1, text=**"Country"**)  txt1 = Entry(window1, width=80)  txt2 = Entry(window1, width=80)  txt3 = Entry(window1, width=80)  txt4 = Entry(window1, width=80)  txt5 = Entry(window1, width=80)  txt6 = Entry(window1, width=80)  txt7 = Entry(window1, width=80)  txt8 = Entry(window1, width=80)  txt9 = Entry(window1, width=80)  txt10 = Entry(window1, width=80)  txt11 = Entry(window1, width=80)  lbl\_info1.place(x=10, y=160)  txt1.place(x=110, y=160)  lbl\_info2.place(x=10, y=190)  txt2.place(x=110, y=190)  lbl\_info3.place(x=10, y=220)  txt3.place(x=110, y=220)  lbl\_info4.place(x=10, y=250)  txt4.place(x=110, y=250)  lbl\_info5.place(x=10, y=280)  txt5.place(x=110, y=280)  lbl\_info6.place(x=10, y=310)  txt6.place(x=110, y=310)  lbl\_info7.place(x=10, y=340)  txt7.place(x=110, y=340)  lbl\_info8.place(x=10, y=370)  txt8.place(x=110, y=370)  lbl\_info9.place(x=10, y=400)  txt9.place(x=110, y=400)  lbl\_info10.place(x=10, y=430)  txt10.place(x=110, y=430)  lbl\_info11.place(x=10, y=460)  txt11.place(x=110, y=460)  btn = Button(window1, text=**"Сохранить"**, command=clicked, width=50)  btn.place(x=120, y=500)  window1.geometry(**'650x600'**)  window1.mainloop()   btn\_graph = Button(window, text=**"Графики"**, command=graphs, width=50)  btn\_pt1 = Button(window, text=**"Скачать сводную таблицу 1"**, command=save\_pt1, width=50)  btn\_pt2 = Button(window, text=**"Скачать сводную таблицу 2"**, command=save\_pt2, width=50)  btn\_pt3 = Button(window, text=**"Скачать сводную таблицу 3"**, command=save\_pt3, width=50)  btn1 = Button(window, text=**"Добавить строку"**, command=add, width=50)  btn2 = Button(window, text=**"Удалить строку"**, command=delete, width=50)  btn3 = Button(window, text=**"Скачать таблицу"**, command=save, width=50)  lbl = Label(window, text=**"Для удаления строки выделите ее и нажмите кнопку 'Удалить строку'"**)  btn1.place(x=500, y=500)  btn2.place(x=500, y=550)  btn3.place(x=500, y=600)  btn\_graph.place(x=500, y=450)  btn\_pt1.place(x=500, y=300)  btn\_pt2.place(x=500, y=350)  btn\_pt3.place(x=500, y=400)  lbl.place(x=500, y=260)   tree.pack()  window.geometry(**'1400x700'**)  window.mainloop()   **def** graphs():  *"""  Функция создает окно с графиками  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* window = Tk()  window.title(**"Graphics"**)  lbl = Label(window, text=**"Графики"**, font=(**"Arial Bold"**, 15))  lbl.place(x=260, y=5)  lbl1 = Label(window, text=**'Зависимость успешных запусков от неуспешных'**)  lbl1.place(x=150, y=40)  btn1 = Button(window, text=**'Построить'**, command=show\_gr1, width=30)  btn1.place(x=50, y=65)  btn2 = Button(window, text=**'Сохранить'**, command=save\_gr1, width=30)  btn2.place(x=300, y=65)  lbl2 = Label(window, text=**'Зависимость успешных запусков от года'**)  lbl2.place(x=165, y=120)  btn3 = Button(window, text=**'Построить'**, command=show\_gr2, width=30)  btn3.place(x=50, y=145)  btn4 = Button(window, text=**'Сохранить'**, command=save\_gr2, width=30)  btn4.place(x=300, y=145)  lbl3 = Label(window, text=**'Зависимость стоимости запуска от года (диаграмма рассеивания)'**)  lbl3.place(x=120, y=200)  btn5 = Button(window, text=**'Построить'**, command=show\_gr3, width=30)  btn5.place(x=50, y=225)  btn6 = Button(window, text=**'Сохранить'**, command=save\_gr3, width=30)  btn6.place(x=300, y=225)  lbl4 = Label(window, text=**'Зависимость количества запусков от года'**)  lbl4.place(x=165, y=280)  btn7 = Button(window, text=**'Построить'**, command=show\_gr4, width=30)  btn7.place(x=50, y=305)  btn8 = Button(window, text=**'Сохранить'**, command=save\_gr4, width=30)  btn8.place(x=300, y=305)  lbl5 = Label(window, text=**'Зависимость стоимости от года'**)  lbl5.place(x=190, y=360)  btn9 = Button(window, text=**'Построить'**, command=show\_gr5, width=30)  btn9.place(x=50, y=385)  btn10 = Button(window, text=**'Сохранить'**, command=save\_gr5, width=30)  btn10.place(x=300, y=385)  lbl6 = Label(window, text=**'Зависимость успешных запусков от неуспешных (столбчатая диаграмма)'**)  lbl6.place(x=100, y=440)  btn11 = Button(window, text=**'Построить'**, command=show\_gr6, width=30)  btn11.place(x=50, y=465)  btn12 = Button(window, text=**'Сохранить'**, command=save\_gr6, width=30)  btn12.place(x=300, y=465)  window.geometry(**'600x600'**)  window.mainloop()   **def** save\_pt1():  *"""  Функция сохраняет сводную таблицу 1  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* space\_missions = reading(path\_to\_csv)  to\_3nf(space\_missions)  space\_missions.info()  pt1 = pt\_country\_status\_mission(space\_missions)  save\_table(pt1, **'Сводная таблица 1'**)  messagebox.showinfo(**'Сделано!'**, **'Вы можете открыть сводную таблицу в формате xlsx '  'в папке Output в файле под названием Сводная таблица 1'**)   **def** save\_pt2():  *"""  Функция сохраняет сводную таблицу 2  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* space\_missions = reading(path\_to\_csv)  to\_3nf(space\_missions)  space\_missions.info()  pt2 = pt\_2(space\_missions)  save\_table(pt2, **'Сводная таблица 2'**)  messagebox.showinfo(**'Сделано!'**, **'Вы можете открыть сводную таблицу в формате xlsx '  'в папке Output в файле под названием Сводная таблица 2'**)   **def** save\_pt3():  *"""  Функция сохраняет сводную таблицу 3  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* space\_missions = reading(path\_to\_csv)  to\_3nf(space\_missions)  space\_missions.info()  pt3 = pt\_3(space\_missions)  save\_table(pt3, **'Сводная таблица 3'**)  messagebox.showinfo(**'Сделано!'**, **'Вы можете открыть сводную таблицу в формате xlsx '  'в папке Output в файле под названием Сводная таблица 3'**)   **def** save\_gr1():  *"""  Функция сохраняет график зависимости успешных запусков от неуспешных  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* outcomes\_for\_russia(**'True'**)  messagebox.showinfo(**'Скачано!'**, **'Вы можете открыть график в папке Graphics '  'в файле под названием Outcomes for Russia.png'**)   **def** show\_gr1():  *"""  Функция выводит график зависимости успешных запусков от неуспешных  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* outcomes\_for\_russia(**'False'**)   **def** save\_gr2():  *"""  Функция сохраняет график зависимости успешных запусков от года  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* year\_success(**'True'**)  messagebox.showinfo(**'Скачано!'**, **'Вы можете открыть график в папке Graphics '  'в файле под названием Year and success.png'**)   **def** show\_gr2():  *"""  Функция выводит график зависимости успешных запусков от года  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* year\_success(**'False'**)   **def** save\_gr3():  *"""  Функция сохраняет график зависимости средней стоимости запуска от года  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* price\_year(**'True'**)  messagebox.showinfo(**'Скачано!'**, **'Вы можете открыть график в папке Graphics '  'в файле под названием Price and year.png'**)   **def** show\_gr3():  *"""  Функция выводит график зависимости средней стоимости запуска от года  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* price\_year(**'False'**)   **def** save\_gr4():  *"""  Функция сохраняет график зависимости количества запусков от года  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* number\_year(**'True'**)  messagebox.showinfo(**'Скачано!'**, **'Вы можете открыть график в папке Graphics '  'в файле под названием Launches per year.png'**)   **def** show\_gr4():  *"""  Функция выводит график зависимости количества запусков от года  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* number\_year(**'False'**)   **def** save\_gr5():  *"""  Функция скачивает график зависимости цены запусков от года  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* max\_min\_price(**'True'**)  messagebox.showinfo(**'Скачано!'**, **'Вы можете открыть график в папке Graphics '  'в файле под названием Max and min price.png'**)   **def** show\_gr5():  *"""  Функция выводит график зависимости цены запусков от года  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* max\_min\_price(**'False'**)  **def** save\_gr6():  *"""  Функция скачивает график количества успешных/неуспешных запусков  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* success\_failure(**'True'**)  messagebox.showinfo(**'Скачано!'**, **'Вы можете открыть график в папке Graphics '  'в файле под названием Outcomes.png'**)   **def** show\_gr6():  *"""  Функция выводит график количества успешных/неуспешных запусков  Входные данные:нет  Выходные данные:нет  Автор: Маркова Э.  """* success\_failure(**'False'**) |
| main.py  (главная программа) | **import** pandas **as** pd **import** matplotlib.pyplot **as** plt **import** sys  sys.path.append(**'../'**) **from** Library.lib **import** \*  **def** pt\_country\_status\_mission(df):  *"""  Функция создает сводную таблицу 1  Входные данные: датафрейм  Выходные данные: датафрейм  Автор: Тарасенко И.  """* pt1 = pd.pivot\_table(df,  index=[**'Country'**],  values=[**'Date'**],  columns=[**'Status Mission'**],  aggfunc=[len],  fill\_value=0)  **return** pt1   **def** pt\_2(df):  *"""  Функция создает сводную таблицу 2  Входные данные: датафрейм  Выходные данные: датафрейм  Автор: Тарасенко И.  """* pt2 = pd.pivot\_table(df,  index=[**'Year'**, **'Status Mission'**],  values=[**'Date'**],  columns=[**'Country'**],  aggfunc=[len],  fill\_value=0)  **return** pt2   **def** pt\_3(df):  *"""  Функция создает сводную таблицу 3  Входные данные: датафрейм  Выходные данные: датафрейм  Автор: Тарасенко И.  """* pt3 = pd.pivot\_table(df,  index=[**'Year'**],  values=[**'Date'**],  columns=[**' Rocket'**],  aggfunc=[len],  fill\_value=0)  **return** pt3   **def** outcomes\_for\_russia(save):  *"""  Функция создает круговой график, показывающий соотношение  успешных запусков к неуспешным для России, на основе сводной таблицы  Входные данные: булева переменная  Выходные данные: нет  Автор: Маркова Э.  """* labels = **'Failure'**, **'Partial failure'**, **'Success'** sizes = [62, 30, 1303]  explode = (0.5, 0.5, 0.5)  fig1, ax1 = plt.subplots()  ax1.pie(sizes, explode=explode, labels=labels, autopct=**'%1.1f%%'**,  shadow=**False**, radius=20, startangle=90)  ax1.legend(title=**'Outcomes for Russia:'**)  ax1.axis(**'equal'**)  **if** save == **'True'**:  fig1.savefig(path\_graph + **'/Outcomes for Russia.png'**)  **if** save == **'False'**:  plt.show()   **def** year\_success(save):  *"""  Функция строит категоризированную диаграмму зависимости успешных  запусков от года  Входные данные: булева переменная  Выходные данные: нет  Автор: Маркова Э.  """* year = [1962, 1963, 1964, 1965, 1966, 1967, 1968,  1969, 1970, 1971, 1972, 1973, 1974, 1975]  number = [4, 7, 7, 13, 33, 37, 40, 52, 54, 56, 60, 63, 64, 71]   plt.bar(year, number, align=**'center'**)  plt.xlabel(**'Year'**)  plt.ylabel(**'Number'**)  plt.title(**'Year/successful missions'**)  **if** save == **'True'**:  plt.savefig(path\_graph + **'/Year and success.png'**)  **if** save == **'False'**:  plt.show()   **def** price\_year(save):  *"""  Функция строит категоризированную диаграмму рассеивания  зависимости средней стоимости от года (за 10-e года 21 века)  Входные данные: булева переменная  Выходные данные: нет  Автор Маркова Э.  """* x = [2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2011, 2012, 2012,  2012,  2012, 2012, 2012, 2012, 2012, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2014, 2014, 2014, 2014, 2014,  2014, 2014, 2014,  2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2017, 2017,  2017, 2017, 2017,  2017, 2017, 2017, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2018, 2019, 2019, 2019, 2019, 2019, 2019, 2019,  2019]  y = [90, 47, 450, 350, 21, 200, 140, 120, 80, 450, 40, 25, 21, 164, 145, 113, 90, 42, 350, 31, 21, 164, 140, 113,  90, 57, 350, 31, 21, 200, 140, 120,  80, 47, 42, 350, 31, 21, 200, 164, 90, 5, 42, 37, 200, 153, 120, 65, 62, 5, 47, 350, 200, 145, 115, 90, 8, 65,  49, 31, 200, 145, 123, 65,  62, 5, 47, 350, 200, 145, 113, 90, 8, 62, 5, 42, 350, 200, 113, 153]  fig3, ax3 = plt.subplots()  ax3.scatter(x, y)  plt.title(**'Year/average price'**)  **if** save == **'True'**:  fig3.savefig(path\_graph + **'/Price and year.png'**)  **if** save == **'False'**:  plt.show()   **def** number\_year(save):  *"""  Функция строит столбчатую диаграмму зависимости количества запусков от года  Входные данные: булева переменная  Выходные данные: нет  Автор Маркова Э.  """* df = pd.read\_csv(path\_to\_csv)  df[**'Year'**] = df[**'Datum'**].apply(**lambda** x: str(x).split(**', '**)[-1])  df[**'Launch Date\_year'**] = df[**'Datum'**].apply(**lambda** x: int(str(x).split()[3]))  fig4 = plt.figure(figsize=(80, 20))  df[**'Launch Date\_year'**].value\_counts().plot(kind=**'bar'**)  plt.xticks(rotation=90)  plt.title(**'Number of launches per year'**)  **if** save == **'True'**:  fig4.savefig(path\_graph + **'/Launches per year.png'**)  **if** save == **'False'**:  fig4.show()   **def** max\_min\_price(save):  *"""  Функция строит категоризированную диаграмму Бокса-Вискера для средней цены запусков  (за 10-е года 21 века)  Автор: Маркова Э.  """* dict = {**'2010'**: [21, 152, 450], **'2011'**: [21, 450, 105], **'2012'**: [350, 21, 122], **'2013'**: [350, 21, 90],  **'2014'**: [350, 21, 103],  **'2015'**: [5.3, 200, 92], **'2016'**: [5.3, 350, 76], **'2017'**: [7.5, 200, 69], **'2018'**: [5.3, 350, 65],  **'2019'**: [5.3, 350, 59]}  fig, ax = plt.subplots()  ax.boxplot(dict.values())  ax.set\_xticklabels(dict.keys())  ax.set\_title(**'Year/price'**)  **if** save == **'True'**:  fig.savefig(path\_graph + **'/Max and min price.png'**)  **if** save == **'False'**:  fig.show()  **def** success\_failure(save):  *"""  Функция создает столбчатую диаграмму, показывающую количество  успешных/неуспешных запусков, на основе сводной таблицы  Входные данные: булева переменная  Выходные данные: нет  Автор: Маркова Э.  """* x =[**'Failure'**, **'Partial failure'**, **'Prelaunch failure'**, **'Success'**]  y = [62, 30, 0, 1303]  plt.bar(x, y)  plt.xlabel(**'Outcome'**)  plt.ylabel(**'Number'**)  plt.title(**'Outcomes for Russia'**)  **if** save == **'True'**:  plt.savefig(path\_graph + **'/Outcomes.png'**)  **if** save == **'False'**:  plt.show() |
| run.py  (исполняемый скрипт) | **import** sys  sys.path.append(**'../'**)  **from** Scripts.main **import** \* **from** Scripts.config **import** \* **from** Scripts.gui **import** \*  space\_missions = reading(path\_to\_csv) to\_3nf(space\_missions) space\_missions.info() main\_window() |
| lib.py  (файл со стандартными библиотеками) | **import** pandas **as** pd **import** matplotlib.pyplot **as** plt **import** sys  sys.path.append(**'../'**) **from** Scripts.config **import** \*   **def** reading(pwd):  *"""  Чтение базы данных  Входные данные: строка  Выходные данные: датафрейм  Автор: Тарасенко И.  """* **return** pd.read\_csv(pwd)   **def** export\_to\_csv(df, title):  *"""  Входные данные: датафрейм, строка  Выходные данные: нет  Автор: Тарасенко И.  """* df.to\_csv(path\_or\_buf=(path + **'/'** + title + **'.csv'**))  **def** save\_table(table, title):  *"""  Функция сохраняет датафрейм в .xslx  Входные данные: датафрейм, строка  Выходные данные: нет  Автор: Маркова Э.  """* table.to\_excel(path\_out + **'/'** + title + **'.xlsx'**)  **def** to\_3nf(df):  *"""  Приведение сводной таблицы к третьей нормальной форме  Входные данные: датафрейм  Выходные данные: нет  Автор: Тарасенко И.  """* df[**'Year'**] = df[**'Datum'**].apply(**lambda** x: int(str(x).split()[3])) *# выцепляю интовый год* df[**'Month'**] = df[**'Datum'**].apply(**lambda** x: str(x).split()[1]) *# выцепляю месяц* df[**'Day of Week'**] = df[**'Datum'**].apply(**lambda** x: str(x).split()[0]) *# выцепляю день недели* df[**'Date'**] = df[**'Datum'**].apply(**lambda** x: int(str(x).split()[2].replace(**','**, **''**))) *# число* df[**'Time in Min'**] = df[**'Datum'**].apply(**lambda** x: int(int(str(x).split()[4].split(**':'**)[0]) \* 60 +  int(str(x).split()[4].split(**':'**)[1]))  **if** (len(str(x).split()) > 4) **else None**) *# Время в минутах* df.drop(**'Datum'**, axis=1, inplace=**True**)   df[**'Country'**] = df[**'Location'**].apply(**lambda** x: str(x).split(**', '**)[-1]) *# заполняю страны как есть* df.drop(**'Location'**, axis=1, inplace=**True**)   df.drop(df.columns[[0, 1]], axis=1, inplace=**True**) *# Выпиливаю повторяющуюся нумерацию* |