ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

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

Руководство пользователя

**Группа:** БИВ216

**Студент:**

Пахомова Алина Евгеньевна

Алешина Алиса Дмитриевна

Агафонова Екатерина Александровна

**Руководитель:**

Полякова Марина Васильевна

МОСКВА 2022

Требование к характеристикам компьютера и операционной системе

Условия по характеристикам персонального компьютера, выполнение которых необходимо для работы с генератором кода:

* ОС Windows 7 или выше.

Программа написана на Python 3.9 (реализация CPython в составе дистрибутива Anaconda). Среда разработки: Spider.

Характеристики и архитектура программы

Используемые методы

Использовались следующие библиотеки:

## Tkinter (для графического интерфейса приложения);

## tkinter.ttk (для графического интерфейса приложения);

## pandas (для обработки и анализа данных);

## numpy (для обработки и анализа данных);

## matplotlib.pyplot (для построения графиков);

## matplotlib.figure (для построения графиков);

## matplotlib.backends.backend\_tkagg (для построения графиков).

Структура программы

Информационно–аналитическое приложение размещается в стандартной структуре каталогов:

Work <- основной каталог.

Data — содержит базу данных.

Graphics — содержит копии графических отчетов.

Library — содержит библиотеку стандартных (универсальных) функций,т разработанных бригадой, которые могут использоваться для создания других приложений.

Notes — содержит документацию, в нем размещается Руководства пользователя и разработчика.

Output — содержит копии текстовых отчетов.

Scripts — содержит специализированный модуль и файл с определением параметров настройки приложения.

Листинг основного скрипта

"""

@author:

Алина Пахомова

Екатерина Агафонова

Алиса Алёшина

"""

import tkinter as tk

from tkinter.ttk import ttk

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.figure import Figure

from matplotlib.backends.backend\_tkagg import (FigureCanvasTkAgg,

NavigationToolbar2Tk)

movies=pd.read\_excel("movies.xlsx")

rating=pd.read\_excel("ratings.xlsx")

all\_movie = rating.merge(movies)

rating\_mean = pd.pivot\_table(rating, values='rating', index=['movieId'],

aggfunc={'rating': np.mean})

films\_with\_mean\_rating = rating\_mean.merge(movies, left\_on='movieId',

right\_on='movieId')

LARGEFONT = ("Verdana", 35)

"""

@author: Алина Пахомова

Основа интерфейса приложения, переклчение между страничками

"""

class tkinterApp:

"""

@author: Алина Пахомова

Создается окно в котором мы будем работать и выстраиваются все параметры

"""

root = tk.Tk()

root.title('HSE.Films')

root.geometry('1050x750+100+0')

root.config(bg = '#121413')

root.resizable(False, True)

# photo = tk.PhotoImage(file = "C:/Users/user/Desktop/питон/photo.jpg")

# root.iconphoto(False, photo)

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

container = tk.Frame(tkinterApp.root)

container.pack()

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

container.grid\_columnconfigure(0, weight = 1, minsize = 1050)

self.frames = {}

for F in (StartPage, what\_watch, selection, statistics):

frame = F(container, self)

self.frames[F] = frame

frame.grid(row = 0, column = 0, sticky ="nsew")

self.show\_frame(StartPage)

tkinterApp.root.mainloop()

def show\_frame(self, cont):

print(cont)

frame = self.frames[cont]

frame.tkraise()

class StartPage(tk.Frame):

"""

@author: Алина Пахомова

Данное поле открывается когда программна начинает работать

"""

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent, bg = '#121413')

label = tk.Label(self, text ="Startpage", font = LARGEFONT, bg =

'#121413', fg = '#ffffff')

label.grid(row = 2, column = 4, padx = 10, pady = 10)

main\_menu = ['Что посмотреть', 'Подборки', 'Статистика']

tk.Button(self, text = main\_menu[0],

font = ('Raleway', 20, 'bold'),

bg = '#ffffff', fg = '#121413',

activeforeground = '#a784e3',

command = lambda: controller.show\_frame(what\_watch

)).grid(row = 0, column = 0, stick = 'snwe')

tk.Button(self, text = main\_menu[1],

font = ('Raleway', 20, 'bold'),

bg = '#ffffff', fg = '#121413',

activeforeground = '#a784e3',

command = lambda: controller.show\_frame(selection

)).grid(row = 0, column = 1, stick = 'snwe')

tk.Button(self, text = main\_menu[2],

font = ('Raleway', 20, 'bold'),

bg = '#ffffff', fg = '#121413',

activeforeground = '#a784e3',

command = lambda: controller.show\_frame(statistics

)).grid(row = 0, column = 2, stick = 'snwe')

self.grid\_columnconfigure(0, minsize = 350)

self.grid\_columnconfigure(1, minsize = 350)

self.grid\_columnconfigure(2, minsize = 350)

self.grid\_rowconfigure(0, minsize = 100)

class what\_watch(tk.Frame):

"""

@author: Алина Пахомова

Фрейм открывается при нажатии кнопки "Что посмотреть"

Здесь реализуется подбор фильмов для конкретного пользователя

"""

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent, bg='#121413')

main\_menu = ['Что посмотреть', 'Подборки', 'Статистика']

main\_command = [what\_watch, selection, statistics]

tk.Button(self, text=main\_menu[1],

font=('Raleway', 20, 'bold'),

bg='#ffffff', fg='#121413',

activeforeground='#a784e3',

command=lambda: controller.show\_frame(main\_command[1]

)).grid(row=0, column=1, stick='snwe')

tk.Button(self, text=main\_menu[2],

font=('Raleway', 20, 'bold'),

bg='#ffffff', fg='#121413',

activeforeground='#a784e3',

command=lambda: controller.show\_frame(main\_command[2]

)).grid(row=0, column=2, stick='snwe')

tk.Button(self, text=main\_menu[0],

font=('Raleway', 30, 'bold'),

bg='#ffffff', fg='#a784e3',

command=lambda: controller.show\_frame(main\_command[0]

)).grid(row=0, column=0, stick='snwe')

lbl\_1 = tk.Label(self, text='Выберите жанр',

font=('Raleway', 20),

fg='#ffffff', bg='#121413',

width=20, height=3)

lbl\_1.grid(row=2, column=1)

self.grid\_columnconfigure(0, minsize=350)

self.grid\_columnconfigure(1, minsize=350)

self.grid\_columnconfigure(2, minsize=350)

self.grid\_rowconfigure(0, minsize=100)

self.yea\_destroy = []

self.genre\_for\_you = ''

self.films\_for\_you = []

self.decade\_for\_you = ''

self.table\_films = pd.DataFrame()

self.dic\_user = {}

self.film\_destroy = []

genres = ['Action', 'Adventure', 'Animation', 'Children', 'Comedy',

'Crime', 'Documentary', 'Film-Noir', 'Drama', 'Fantasy',

'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi',

'Thriller', 'War', 'Western']

years = ('2015-2020', '2010-2015', '2005-2010', '2000-2005',

'1995-2000', '1990-1995', '1985-1990', '1980-1985',

'1975-1980', '1970-1975', '1965-1970', '1960-1965',

'1955-1960', '1950-1955', '1945-1950', '1940-1945',

'1935-1940', '1930-1935', '1925-1930')

i = 0

k = 4

p = 0

list\_gen = []

for j in range(19):

list\_gen.append(tk.IntVar())

list\_yea = []

for j in range(20):

list\_yea.append(tk.IntVar())

list\_film = []

for j in range(5000):

list\_film.append(tk.IntVar())

def podborki(year, genre):

year = year.split("-")

f1 = movies[movies["genre 1"] == genre]

f2 = movies[movies["genre 2"] == genre]

f3 = movies[movies["genre 3"] == genre]

f4 = movies[movies["genre 4"] == genre]

f5 = movies[movies["genre 5"] == genre]

f6 = movies[movies["genre 6"] == genre]

f7 = movies[movies["genre 7"] == genre]

f8 = movies[movies["genre 8"] == genre]

f9 = movies[movies["genre 9"] == genre]

f10 = movies[movies["genre 10"] == genre]

genreall = pd.concat([f1, f2, f3, f4, f5, f6, f7, f8, f9, f10],

sort=False, axis=0)

new\_movies = genreall[genreall["year"] > int(year[0])]

new1 = new\_movies[new\_movies["year"] <= int(year[1])]

new1 = pd.concat([new1['title']])

b = new1

b = list(b)

return b

def click4():

'''

@author: Алиса

функция суммирует все райтинги по данному жанру и фильмам которые понравились

пользователю, затем выдает список фильльмов, предложенных пользователю,

по определенному критерию

'''

for mid in self.table\_films:

d = rating[rating['movieId'] == mid]

for u in d['userId']:

d\_user = u

r = d[d['userId'] == d\_user]

d\_rate = r['rating'].sum()

self.dic\_user[d\_user]=self.dic\_user.get(d\_user, 0)+d\_rate

maxim = 0

for key in self.dic\_user:

maxim = max(maxim, self.dic\_user[key])

if self.dic\_user[key] == maxim:

main\_user = key

for fi in self.film\_destroy:

fi.destroy()

genre = self.genre\_for\_you

year = self.decade\_for\_you

user = main\_user

year = year.split("-")

f1 = all\_movie[all\_movie["genre 1"] == genre]

f2 = all\_movie[all\_movie["genre 2"] == genre]

f3 = all\_movie[all\_movie["genre 3"] == genre]

f4 = all\_movie[all\_movie["genre 4"] == genre]

f5 = all\_movie[all\_movie["genre 5"] == genre]

f6 = all\_movie[all\_movie["genre 6"] == genre]

f7 = all\_movie[all\_movie["genre 7"] == genre]

f8 = all\_movie[all\_movie["genre 8"] == genre]

f9 = all\_movie[all\_movie["genre 9"] == genre]

f10 = all\_movie[all\_movie["genre 10"] == genre]

new1 = pd.concat([f1, f2, f3, f4, f5, f6, f7, f8, f9, f10],

sort=False, axis=0)

fim = new1[new1['userId'] == main\_user]

fim2 = fim[fim['rating'] >= 5]

fim2 = pd.concat([fim2['title']])

# fim2.reset\_index(drop=True, inplace=True)

b = fim2

if b.empty:

lbl = tk.Label(self, text='Ничего не найдено',

font=('Raleway', 14),

fg='#ffffff', bg='#121413',

width=20, height=3)

else:

# lbl = tk.Label(self, text=b, font=('Raleway', 14, 'italic'),

# bg='#ffffff', fg='#a784e3')

# lbl.grid(column=0, row=4, columnspan = 2, stick = "w")

sb = tk.Scrollbar(self)

sb.grid(row=4, column=2, rowspan = 3, stick = 'nse')

mylist = tk.Listbox(self, yscrollcommand = sb.set, font=

('Raleway', 14, 'italic'), bg='#ffffff', fg='#a784e3')

for line in b:

mylist.insert(tk.END, str(line))

mylist.grid(column=0, row=4, columnspan=2, stick = "we",

padx = 15)

sb.config( command = mylist.yview )

lbl\_2 = tk.Label(self, text='Фильмы для Вас',

font=('Raleway', 20),

fg='#ffffff', bg='#121413',

width=20, height=3)

lbl\_2.grid(row=2, column=1)

def click3():

'''

@author: Алиса

функция обрабатывает кнопки на которые нажал пользователь по фильмам

'''

list\_of\_films = []

my\_films = []

for i in range(len(list\_film)):

list\_of\_films.append(list\_film[i].get())

endbut = tk.Button(self, text='Завершить выбор',

font=('Raleway', 14, 'bold'),

bg='#a784e3', fg='#121413',

activeforeground='#a784e3',

command=click4)

############################

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

###############################################

for m in range(len(list\_of\_films) - 1):

if list\_of\_films[m] == 1:

my\_films.append(self.films\_for\_you[m - 3])

# print(my\_films)

table\_films\_filter = movies[movies['title'].isin(my\_films)]

self.table\_films = table\_films\_filter['movieId']

# print(self.table\_films)

return self.table\_films

fil = []

for m in my\_films:

fil.append(movies[movies['title'] == m]['title'])

def click2():

'''

@author: Алиса

функция генерирует список фильмов которые предлагаются пользователю на выбор

'''

list\_of\_values\_years = []

for i in range(20):

list\_of\_values\_years.append(list\_yea[i].get())

for y in range(len(list\_of\_values\_years) - 1):

if list\_of\_values\_years[y] == 1:

self.decade\_for\_you = years[y - 1]

for ye in self.yea\_destroy:

ye.destroy()

lbl\_2 = tk.Label(self, text='Выберите фильмы ',

font=('Raleway', 20),

fg='#ffffff', bg='#121413',

width=20, height=3)

lbl\_2.grid(row=2, column=1)

self.films\_for\_you = podborki(self.decade\_for\_you,

self.genre\_for\_you)

i = 3

########################################

for fil\_f\_u in self.films\_for\_you:

fil = tk.Checkbutton(self, text=fil\_f\_u,

font=('Raleway', 12, 'bold'),

bg='#a784e3', fg='#121413',

activeforeground='#a784e3', indicatoron=0,

variable=list\_film[i],command=click3)

self.film\_destroy.append(fil)

fil.grid(column=0, row=i, columnspan=2, stick='w', padx=15)

i += 1

################################################

def click1():

'''

@author: Алиса

функция обрабатывает год и жанр которые выбрал пользователь

'''

list\_of\_values = []

for i in range(19):

list\_of\_values.append(list\_gen[i].get())

for g in range(len(list\_of\_values) - 1):

if list\_of\_values[g] == 1:

self.genre\_for\_you = genres[g - 1]

p = 0

j = 0

i = 0

k = 4

for des in gen\_dectroy:

des.destroy()

########################################################Добавила

lbl\_2 = tk.Label(self, text='Выберите год',

font=('Raleway', 20),

fg='#ffffff', bg='#121413',

width=20, height=3)

lbl\_2.grid(row=2, column=1)

##################################################################

for year in years:

p += 1

yea = tk.Checkbutton(self, text=year,

font=('Raleway', 14, 'bold'),

bg='#a784e3', fg='#121413',

activeforeground='#a784e3', indicatoron=0,

variable=list\_yea[p], command=click2)

self.yea\_destroy.append(yea)

yea.grid(column=i, row=k)

i += 1

j += 1

if j % 3 == 0:

k += 1

i = 0

gen\_dectroy = []

for genre in genres:

p += 1

gen = tk.Checkbutton(self, text=genre,

font=('Raleway', 14, 'bold'),

bg='#a784e3', fg='#121413',

activeforeground='#a784e3', indicatoron=0,

variable=list\_gen[p], command=click1)

gen\_dectroy.append(gen)

gen.grid(column=i, row=k)

i += 1

j += 1

if j % 3 == 0:

k += 1

i = 0

class selection(tk.Frame):

"""

@author: Алина Пахомова

Фрейм открывается при нажатии кнопки "Подборки"

Здесь реализуется возможность составить фильмы по определенным жанрам и году с учетом рейтинга

"""

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent, bg='#121413')

main\_menu = ['Что посмотреть', 'Подборки', 'Статистика']

main\_command = [what\_watch, selection, statistics]

tk.Button(self, text=main\_menu[0],

font=('Raleway', 20, 'bold'),

bg='#ffffff', fg='#121413',

activeforeground='#a784e3',

command=lambda: controller.show\_frame(main\_command

[0])).grid(row=0, column=0, stick='snwe')

tk.Button(self, text=main\_menu[2],

font=('Raleway', 20, 'bold'),

bg='#ffffff', fg='#121413',

activeforeground='#a784e3',

command=lambda: controller.show\_frame(main\_command

[2])).grid(row=0, column=2, stick='snwe')

tk.Button(self, text=main\_menu[1],

font=('Raleway', 30, 'bold'),

bg='#ffffff', fg='#a784e3',

command=lambda: controller.show\_frame(main\_command

[1])).grid(row=0, column=1, stick='snwe')

self.grid\_columnconfigure(0, minsize=350)

self.grid\_columnconfigure(1, minsize=350)

self.grid\_columnconfigure(2, minsize=350)

self.grid\_rowconfigure(0, minsize=100)

combostyle = ttk.Style()

combostyle.theme\_create('combostyle', parent='alt',

settings={'TCombobox':

{'configure':

{'selectbackground': '#a784e3',

'fieldbackground': '#a784e3',

'background': '#ffffff'}}})

combostyle.theme\_use('combostyle')

genres = ('Жанр', 'Adventure', 'Animation', 'Children', 'Comedy',

'Crime', 'Documentary','Drama', 'Fantasy', 'Sci-Fi',

'Thriller', 'War', 'Western')

years = ("Год", '2015-2020', '2010-2015', '2005-2010', '2000-2005',

'1995-2000', '1990-1995', '1985-1990', '1980-1985',

'1975-1980', '1970-1975', '1965-1970', '1960-1965',

'1955-1960', '1950-1955', '1945-1950', '1940-1945',

'1935-1940', '1930-1935', '1925-1930')

sorts = ('Сортировать по...', 'От лучшего к худшему',

'От худшего к лучшему')

genre = ttk.Combobox(self, values=genres,

font=("Raleway Thin", 18, "bold"))

year = ttk.Combobox(self, values=years, font=

("Raleway Thin", 18, "bold"))

sort = ttk.Combobox(self, values=sorts, font=

("Raleway Thin", 18, "bold"))

yearlist = list(years)

genre.current(0)

year.current(0)

sort.current(0)

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

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

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

self.grid\_rowconfigure(1, minsize=100)

def podborki(year, genre, sort):

"""

@author: Катя

Функция принимает значения годов выпуска фильмов, их жанр и тип

сортировки, выбранные пользователем, и возвращает таблицу DataFrame

с названием, годом и средним рейтингом подходящих фильмов

"""

year = year.split("-")

f1 = movies[movies["genre 1"] == genre]

f2 = movies[movies["genre 2"] == genre]

f3 = movies[movies["genre 3"] == genre]

f4 = movies[movies["genre 4"] == genre]

f5 = movies[movies["genre 5"] == genre]

f6 = movies[movies["genre 6"] == genre]

f7 = movies[movies["genre 7"] == genre]

f8 = movies[movies["genre 8"] == genre]

f9 = movies[movies["genre 9"] == genre]

f10 = movies[movies["genre 10"] == genre]

genreall = pd.concat(

[f1, f2, f3, f4, f5, f6, f7, f8, f9, f10], sort=False, axis=0)

rating\_mean = pd.pivot\_table(rating, values='rating', index=[

'movieId'], aggfunc={'rating':

np.mean})

genreall = rating\_mean.merge(

genreall, left\_on='movieId', right\_on='movieId')

new\_movies = genreall[genreall["year"] > int(year[0])]

new1 = new\_movies[new\_movies["year"] <= int(year[1])]

new1 = pd.concat([new1['title'], new1['year'], new1['rating']],

axis=1)

if sort == 'От лучшего к худшему':

new2 = new1.sort\_values(by='rating', ascending=False)

new2.reset\_index(drop=True, inplace=True)

b = new2

if sort == 'От худшего к лучшему':

new3 = new1.sort\_values(by='rating')

new3.reset\_index(drop=True, inplace=True)

b = new3

return b

def clck():

"""

@author: Катя

Функция получает данные, выбранные пользователем, вызывает функцию

преобразования этих данных в таблицу и выводит на экран год,

рейтинг и название подходящих фильмов

"""

res = year.get()

res2 = genre.get()

res3 = sort.get()

b = podborki(res, res2, res3)

if b.empty:

lbl.config(text="Ничего не найдено", font=('Raleway', 14,

'italic'), bg='#ffffff', fg='#121413', activeforeground='#a784e3')

else:

sb = tk.Scrollbar(self)

sb.grid(row=8, column=2, rowspan = 3, stick = 'nse')

mylist = tk.Listbox(self, yscrollcommand = sb.set, font=

('Raleway', 14, 'italic'), bg='#ffffff', fg='#a784e3')

for line in range(len(b['title'])):

mylist.insert(tk.END, str(b['year'][line])+ " " +

str(round(float(b['rating'][line]), 1)) +

" " + str(b['title'][line]))

mylist.grid(column=0, row=8, columnspan=3, stick = "we",

padx = 15)

sb.config( command = mylist.yview )

lbl = tk.Label(self, text="Ищем...", font=('Raleway', 14, 'italic'))

lbl.grid(column=0, row=8)

bth = tk.Button(self, text='Вывод результатов', font=('Raleway', 14,

'italic'), bg='#ffffff', fg='#121413',

activeforeground='#a784e3', command=clck)

bth.grid(column=2, row=10)

class statistics(tk.Frame):

"""

@author: Алина Пахомова

Фрейм открывается при нажатии кнопки "Статистика"

Здесь реализуется возможность составлять графики

"""

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent, bg='#121413')

main\_menu = ['Что посмотреть', 'Подборки', 'Статистика']

main\_command = [what\_watch, selection, statistics]

tk.Button(self, text=main\_menu[0],

font=('Raleway', 20, 'bold'),

bg='#ffffff', fg='#121413',

activeforeground='#a784e3',

command=lambda: controller.show\_frame(main\_command[0]

)).grid(row=0, column=0, stick='snwe')

tk.Button(self, text=main\_menu[1],

font=('Raleway', 20, 'bold'),

bg='#ffffff', fg='#121413',

command=lambda: controller.show\_frame(main\_command[1]

)).grid(row=0, column=1, stick='snwe')

tk.Button(self, text=main\_menu[2],

font=('Raleway', 30, 'bold'),

bg='#ffffff', fg='#a784e3',

activeforeground='#a784e3',

command=lambda: controller.show\_frame(main\_command[2]

)).grid(row=0, column=2, stick='snwe')

self.grid\_columnconfigure(0, minsize=350)

self.grid\_columnconfigure(1, minsize=350)

self.grid\_columnconfigure(2, minsize=350)

self.grid\_rowconfigure(0, minsize=100)

combostyle2 = ttk.Style()

combostyle2.theme\_use('combostyle')

xcrits = ('Критерии по x', 'Жанр', '2014-2022', '2005-2013',

'1996-2004', '1987-1995', '1978-1986',

'1969-1977', '1960-1968', '1951-1959', '1942-1950')

ycrits = ("Критерии по y","Средний рейтинг", "Количество оценок")

xcrit = ttk.Combobox(self, values=xcrits, font=(

"Raleway Thin", 18, "bold"))

ycrit=ttk.Combobox(self, values=ycrits, font=(

"Raleway Thin", 18, "bold"))

xcrit.current(0)

ycrit.current(0)

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

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

def select(xcrit, ycrit):

"""

@author: Катя

Функция принимает критерии по x(год или жанры) и критерии по y

(средний рейтинг или количество оценок), выбранные пользователем,

и возвращает таблицу DataFrame с нужными данными

"""

# if rate=="Количество фильмов":

# z=films\_with\_mean\_rating["rating"]

# if rate =="Среднее число фильмов":

# z=films\_with\_mean\_rating['year']

# plt.boxplot(y, notch=True, showmeans=True) #год, в который

# было снято среднее колво фильмов

if ycrit=="Количество оценок":

if xcrit!='Жанр' and xcrit!='Критерии по y':

xcrit=xcrit.split("-")

print(xcrit)

all\_rating = pd.pivot\_table(all\_movie, index=['year'],

values=['rating'], aggfunc={'rating': len})

all\_rating["year"] = all\_rating.index

ind = np.array(range(1, 108))

all\_rating = all\_rating.set\_index(ind)

x = all\_rating[all\_rating["year"] > int(xcrit[0])]

x = x[x["year"] <= int(xcrit[1])]

x = x["year"]

y = all\_rating["rating"][x.index[0]-1:x.index[-1]]

z = pd.concat([x, y], axis=1)

if xcrit=='Жанр':

#z=all\_movie[all\_movie['userId']==1]

v=pd.pivot\_table(all\_movie, index=['genre 1'], values=

['rating'], aggfunc={'rating':len})

v["genre 1"]=v.index

d=np.array(range(1,21))

u=v.set\_index(d)

u=u.drop(labels = [1],axis = 0)

u=u.drop(labels = [2],axis = 0)

x=u['genre 1']

y=u['rating']

z = pd.concat([x, y], axis=1)

# plt.figure(figsize=(9.5,6))

if ycrit=='Средний рейтинг':

if xcrit!='Жанр' and xcrit!='Критерии по y':

xcrit=xcrit.split("-")

print(xcrit)

mean\_rating = pd.pivot\_table(all\_movie, index=['year'],

values=['rating'], aggfunc={'rating': np.mean})

mean\_rating["year"] = mean\_rating.index

ind = np.array(range(1, 108))

mean\_rating = mean\_rating.set\_index(ind)

x = mean\_rating[mean\_rating["year"] > int(xcrit[0])]

x = x[x["year"] <= int(xcrit[1])]

x = x["year"]

y = mean\_rating["rating"][x.index[0]-1:x.index[-1]]

z = pd.concat([x, y], axis=1)

if xcrit=='Жанр':

#z=all\_movie[all\_movie['userId']==1]

v=pd.pivot\_table(all\_movie, index=['genre 1'], values=

['rating'], aggfunc={'rating':np.mean})

v["genre 1"]=v.index

d=np.array(range(1,21))

u=v.set\_index(d)

u=u.drop(labels = [1],axis = 0)

u=u.drop(labels = [2],axis = 0)

x=u['genre 1']

y=u['rating']

z = pd.concat([x, y], axis=1)

# plt.figure(figsize=(9.5,6))

# plt.scatter(x,y, marker="^", color="#DC143C")

return z

def clck2():

"""

@author: Катя

Функция получает данные, выбранные пользователем, вызывает функцию

преобразования этих данных в таблицу и выводит на экран график

зависимости этих данных друг от друга

"""

global canvas1

canvas1=None

if canvas1:

canvas1.get\_tk\_widget().destroy()

res = xcrit.get()

res2 = ycrit.get()

data = select(res, res2)

b=int(data.index[0])

if data.columns[0]=="genre 1":

plt.rcParams['xtick.labelsize'] = '6'

# тоже меняет размер шрифта жанров

# for label in ax.get\_xticklabels():

# label.set\_fontsize(6)

fig = Figure(figsize=(10, 4), dpi=100)

ax = fig.add\_subplot(111)

if data["rating"][b]>6:

ax.set\_title

('График зависимости количества оценок от жанра')

ax.set\_ylabel('Количество оценок')

else:

ax.set\_title

('График зависимости среднего рейтинга от жанра')

ax.set\_ylabel('Средний рейтинг')

ax.set\_xlabel('Жанр')

ax.scatter(data["genre 1"], data["rating"], color='#a784e3')

if data.columns[0]=="year":

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

ax = fig.add\_subplot(111)

if data["rating"][b]>6:

ax.set\_title

('График зависимости количества оценок от года')

ax.set\_ylabel('Количество оценок')

else:

ax.set\_title

('График зависимости среднего рейтинга от года')

ax.set\_ylabel('Средний рейтинг')

ax.set\_xlabel('Год выпуска')

ax.bar(data["year"], data["rating"], color='#a784e3')

canvas1 = FigureCanvasTkAgg(fig, master=self)

canvas1.draw()

canvas1.get\_tk\_widget().grid(column=0, row=4, columnspan=3)

self.after(200, None)

def close():

"""

@author: Катя

Функция удаляет текущий график с экрана. Обратите внимание, что

очищение экрана необходимо проводить после каждого нового вывода

графика!

"""

global canvas1

if canvas1:

canvas1.get\_tk\_widget().destroy()

canvas1 = None

bth = tk.Button(self, text='Вывод графика', font=('Raleway', 14,

'italic'),bg='#ffffff', fg='#121413',

activeforeground='#a784e3', command=clck2, padx=5, pady=5)

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

bth = tk.Button(self, text='Сброс графика', font=('Raleway', 14,

'italic'),bg='#ffffff', fg='#121413',

activeforeground='#a784e3', command=close, padx=5, pady=5)

bth.grid(column=2, row=2)

# Driver Code

win = tkinterApp()