Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ
№220дисциплины «Основы программной инженерии»

Выполнил:
Джараян Арег Александрович
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил Воронкин Роман Александрович
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Лабораторная работа 4.8 Обработка событий и рисование в Tkinter.

Цель работы: приобретение навыков улучшения графического интерфейса пользователя GUI с помощью обработки событий и рисования, реализованных в пакете Tkinter языка программирования Python версии 3.x.

1. Создание нового репозитория с лицензией MIT.

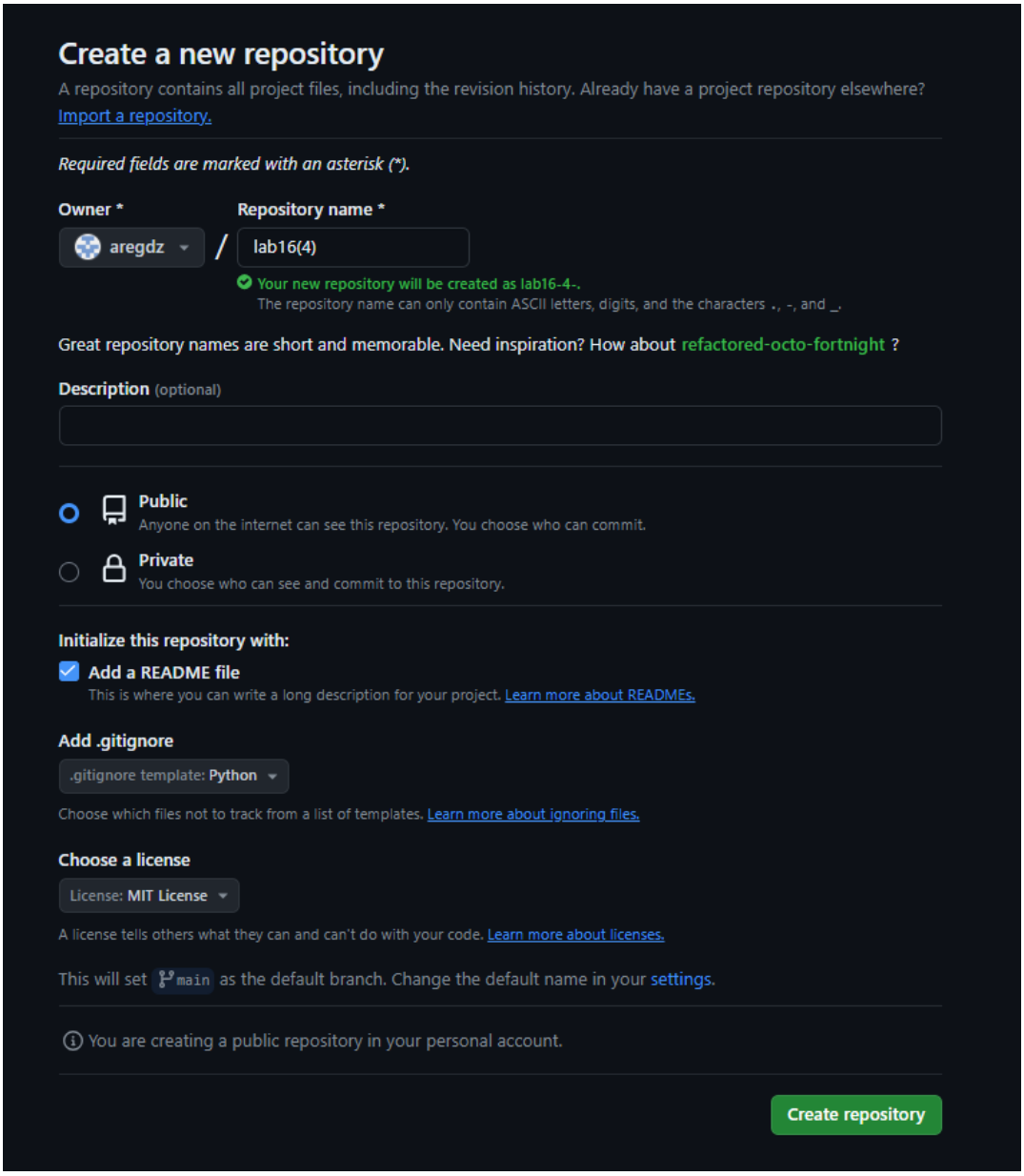

Рисунок 1 – создание репозитория

2. Клонировал репозиторий на рабочий ПК.

```
aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/16(4)
$ git clone https://github.com/aregdz/lab16-4-.git
Cloning into 'lab16-4-'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/16(4)
$
```

Рисунок 2 – клонирование репозитория

3. Дополнил файл .gitignore необходимыми инструкциями.

```
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
```

Рисунок 4 – Файл .gitignore

```
aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/16(4)/lab16-4- (main
)
$ git checkout -b develop
Switched to a new branch 'develop'

aregd@DESKTOP-5KV9QA9 MINGW64 /d/Рабочий стол/4 семестр/опи/16(4)/lab16-4- (deve
lop)
$
```

Рисунок 4 – организация ветки

```
(venv) PS D:\Рабочий стол\4 сем
Package         Version
--------------- -------
black           24.4.0
cfgv            3.4.0
click           8.1.7
colorama        0.4.6
distlib         0.3.8
pyflakes        3.2.0
PyYAML          6.0.1
setuptools      69.5.1
virtualenv      20.25.2
```

Рисунок 5 – создание виртуального окружения

4.Отработал примеры лабораторной работы.

5. . Решите задачу: напишите программу, состоящую из двух списков Listbox . В первом будет, например, перечень товаров, заданный программно. Второй изначально пуст, пусть это будет перечень покупок. При клике на одну кнопку товар должен переходить из одного списка в другой. При клике на вторую кнопку – возвращаться (человек передумал покупать). Предусмотрите возможность множественного выбора элементов списка и их перемещения.

```
from tkinter import *

def move_items(from_list, to_list):
```

```
    selected_items = list(from_list.curselection())
    selected_items.reverse()
    for index in selected_items:
        to_list.insert(END, from_list.get(index))
        from_list.delete(index)

def move_to_shopping_list():
    move_items(available_items, shopping_list)

def move_back_to_available():
    move_items(shopping_list, available_items)

root = Tk()
root.title("Список товаров и покупок")

available_items = Listbox(root, selectmode=MULTIPLE)
available_items.pack(side=LEFT, padx=5, pady=5)
available_items.insert(END, "Яблоки")
available_items.insert(END, "Бананы")
available_items.insert(END, "Молоко")
available_items.insert(END, "Хлеб")
available_items.insert(END, "Масло")

shopping_list = Listbox(root, selectmode=MULTIPLE)
shopping_list.pack(side=LEFT, padx=5, pady=5)

button_frame = Frame(root)
button_frame.pack(side=LEFT, padx=5, pady=5)
Button(button_frame, text="Добавить в список",
command=move_to_shopping_list).pack(fill=X)
Button(button_frame, text="Удалить из списка",
command=move_back_to_available).pack(fill=X)

root.mainloop()
```

Рисунок 6 – Задание 1

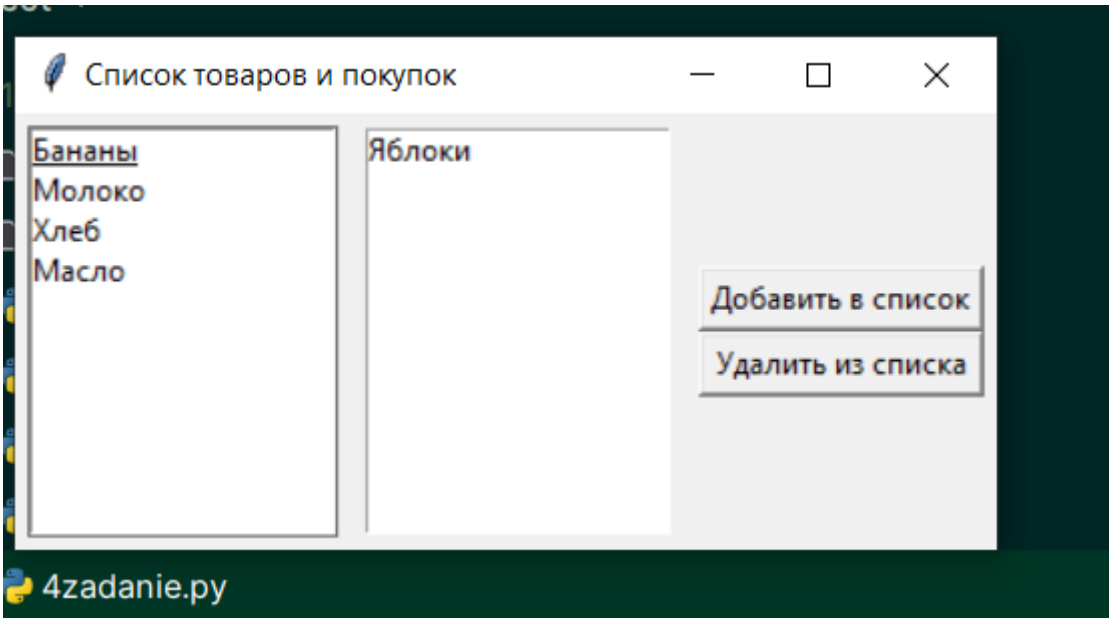

Рисунок 7 – пример выполнения примера 2

6. Решите задачу: напишите программу по следующему описанию. Нажатие Enter в однострочном текстовом поле приводит к перемещению

текста из него в список (экземпляр Listbox). При двойном клике (<Double-Button-1>) по элементу-строке списка, она должна копироваться в текстовое поле.

```
from tkinter import *

def add_item(event=None):
    text = entry.get()
    if text:
        shopping_list.insert(END, text)
        entry.delete(0, END)

def copy_item(event):
    selected_index = shopping_list.curselection()
    if selected_index:
        text_to_copy = shopping_list.get(selected_index)
        entry.delete(0, END)
        entry.insert(END, text_to_copy)

root = Tk()
root.title("Перемещение и копирование элементов")

entry = Entry(root)
entry.pack(pady=5, padx=10, fill=X)
entry.bind("<Return>", add_item)

shopping_list = Listbox(root)
shopping_list.pack(pady=5, padx=10, fill=BOTH, expand=True)
shopping_list.bind("<Double-Button-1>", copy_item)

root.mainloop()
```

Рисунок 10 – код для выполнения задания 2

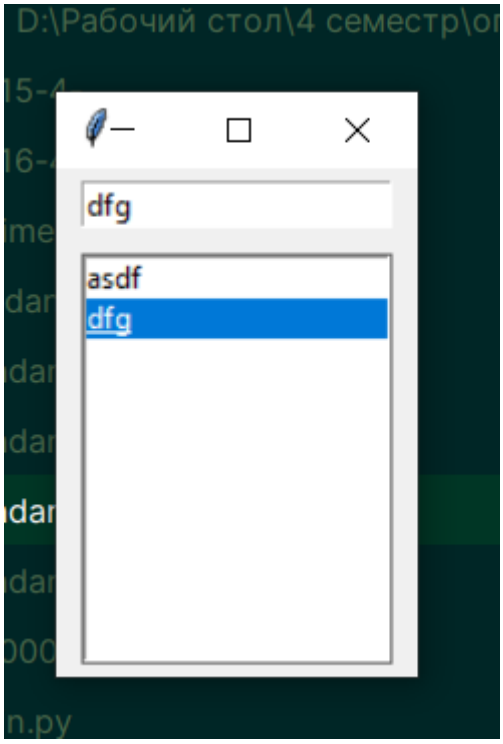

Рисунок 11 – пример выполнения задания 2

7. Решите задачу: напишите программу по описанию. Размеры многострочного текстового поля определяются значениями, введенными в однострочные текстовые поля. Изменение размера происходит при нажатии мышью на кнопку, а также при нажатии клавиши Enter. Цвет фона экземпляра Text светлосерый (lightgrey), когда поле не в фокусе, и белый, когда имеет фокус. Событие получения фокуса обозначается как <FocusIn> , потери – как <FocusOut> . Для справки: фокус перемещается по виджетам при нажатии Tab, Ctrl+Tab, Shift+Tab, а также при клике по ним мышью (к кнопкам последнее не относится).

```
from tkinter import *

def change_size(event=None):
    try:
        width = int(width_entry.get())
        height = int(height_entry.get())
        text_area.config(width=width, height=height)
    except ValueError:
        pass

def change_background_color(event):
    text_area.config(bg="white")

def change_background_color_out(event):
    text_area.config(bg="lightgrey")

root = Tk()
root.title("Изменение размера многострочного текстового поля")

Label(root, text="Ширина:").grid(row=0, column=0, padx=5, pady=5)
width_entry = Entry(root)
width_entry.grid(row=0, column=1, padx=5, pady=5)
Label(root, text="Высота:").grid(row=1, column=0, padx=5, pady=5)
height_entry = Entry(root)
height_entry.grid(row=1, column=1, padx=5, pady=5)
Button(root, text="Применить", command=change_size).grid(row=2, columnspan=2, pady=5)

text_area = Text(root, bg="lightgrey")
text_area.grid(row=3, columnspan=2, padx=5, pady=5)

text_area.bind("<FocusIn>", change_background_color)
text_area.bind("<FocusOut>", change_background_color_out)

width_entry.bind("<Return>", change_size)
height_entry.bind("<Return>", change_size)

root.mainloop()
```

Рисунок 12 – код для выполнения задания 3

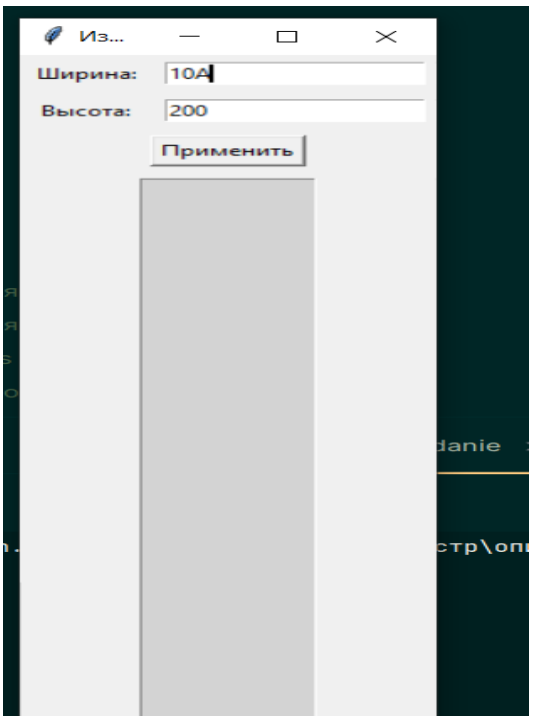

Рисунок 13 – пример выполнения задания 3

8. Решите задачу: Создайте на холсте подобное изображение:

```
from tkinter import *

root = Tk()
root.title("Рисование на холсте")

canvas = Canvas(root, width=400, height=400, bg='lightblue')
canvas.pack()

sun = canvas.create_oval(250, 50, 350, 150, fill='yellow', outline='yellow')

triangle = canvas.create_polygon(50, 200, 250, 200, 150, 50, fill='blue')

rectangle = canvas.create_rectangle(130, 200, 180, 400, fill='blue')

for i in range(0, 400, 20):
    canvas.create_line(i, 400, i, 380, fill='green')


root.mainloop()
```

Рисунок 14 – код для выполнения задания 4

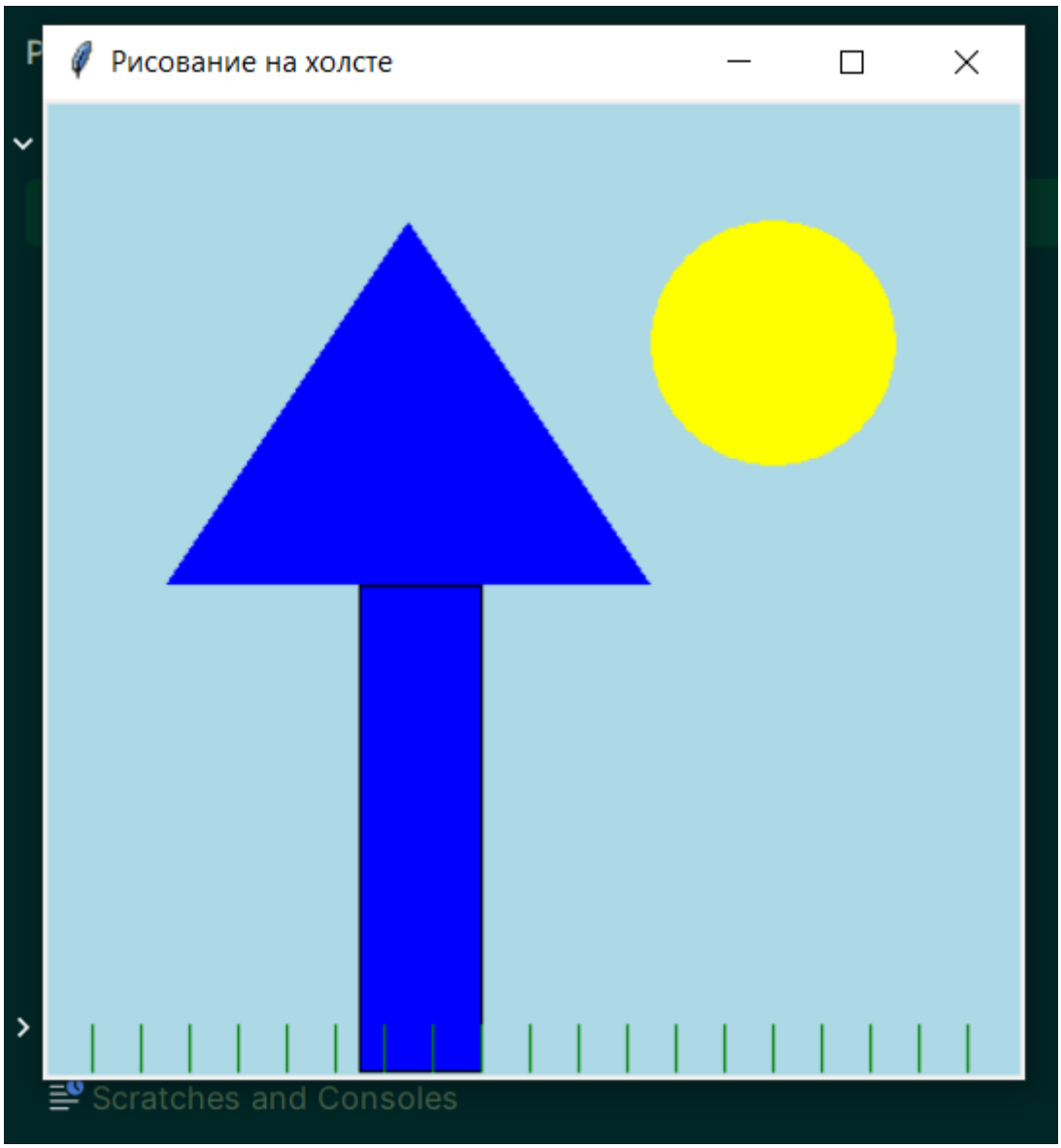

Рисунок 15 – пример выполнения задания 4

9.Решите задачу: в данной программе создается анимация круга, который движется от левой границы холста до правой: Выражение c.coords(ball) возвращает список текущих координат объекта (в данном случае это ball). Третий элемент списка соответствует его второй координате x. Метод after вызывает функцию, переданную вторым аргументом, через количество миллисекунд, указанных первым аргументом. Изучите приведенную программу и самостоятельно запрограммируйте постепенное движение фигуры в ту точку холста, где пользователь кликает левой кнопкой мыши.

Координаты события хранятся в его атрибутах x и y (event.x , event.y).

```
from tkinter import *

def move_to_point(event):
    x1, y1, x2, y2 = c.coords(ball)
    target_x, target_y = event.x, event.y
    dx = target_x - x1
    dy = target_y - y1
    while abs(dx) > 1 or abs(dy) > 1:
        x1 += int(dx * 0.1)
        y1 += int(dy * 0.1)
        x2 += int(dx * 0.1)
        y2 += int(dy * 0.1)
        c.coords(ball, x1, y1, x2, y2)
        c.update()
        dx = target_x - x1
        dy = target_y - y1


root = Tk()
root.title("Постепенное движение круга")


c = Canvas(root, width=400, height=400, bg="white")
c.pack()


ball = c.create_oval(0, 100, 40, 140, fill='green')


c.bind("<Button-1>", move_to_point)


root.mainloop()
```

Рисунок 16 – код для выполнения задания 5

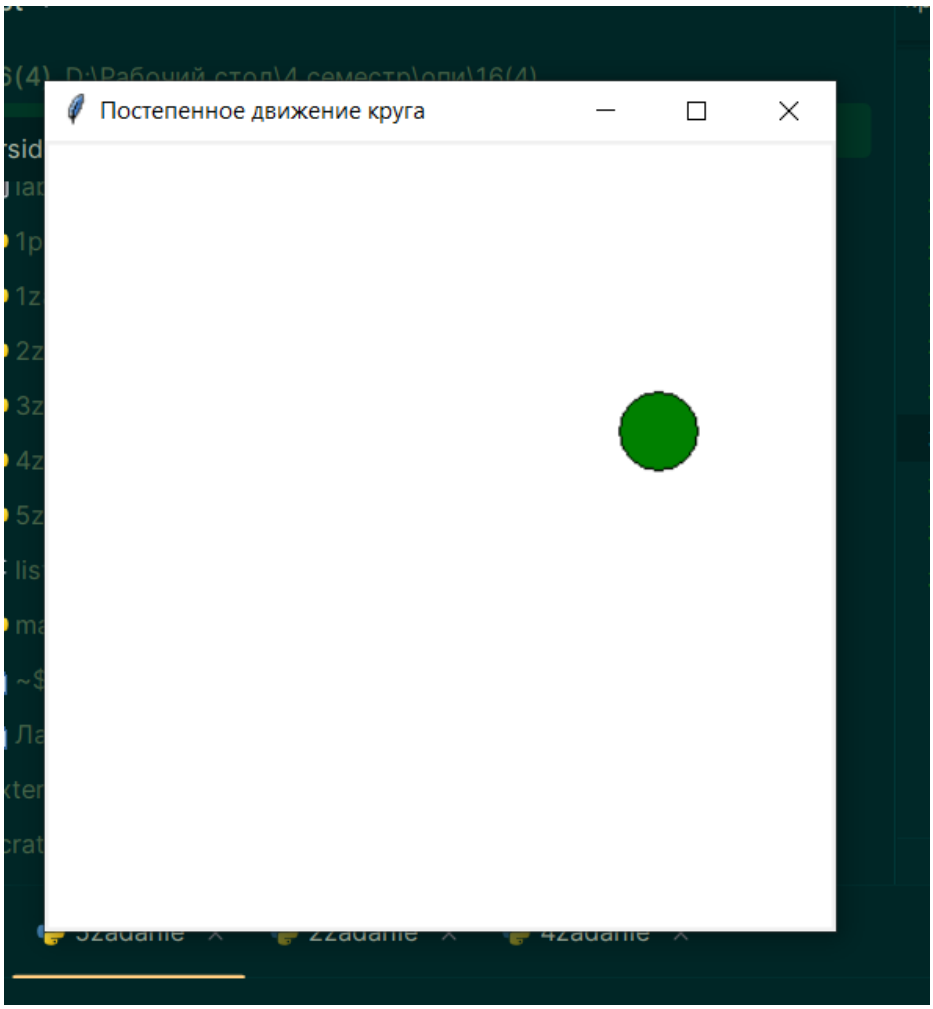

Рисунок 17 – пример выполнения задания 5

Контрольные вопросы:

1.Виджет ListBox в Tkinter предназначен для отображения списка элементов, из которых пользователь может выбирать один или несколько.

2.Связывание событий или действий с виджетом Tkinter происходит с помощью метода bind() или с помощью передачи функции обратного вызова в параметре command (если поддерживается).

3.В Tkinter существует несколько типов событий, включая события клавиатуры (например, <KeyPress>), события мыши (например, <Button-1> для щелчка левой кнопкой мыши), события фокуса (например, <FocusIn>), и другие.

4.События в Tkinter обрабатываются путем привязки функций (обработчиков) к событиям с помощью метода bind() или путем использования метода bind_all() для обработки событий во всем приложении.

5.События мыши в Tkinter обрабатываются путем привязки функций к различным событиям мыши, таким как щелчок, движение мыши и т. д., с помощью метода bind().

6.Графические примитивы в Tkinter можно отображать на холсте (Canvas) с помощью специальных методов, таких как create_line(), create_rectangle(), create_oval() и т. д.

7.Основные методы для отображения графических примитивов в Tkinter включают create_line() для отрисовки линий, create_rectangle() для отображения прямоугольников, create_oval() для отрисовки овалов и кругов, и другие подобные методы, такие как create_polygon() и create_text().

8.Для обращения к ранее созданным фигурам на холсте в Tkinter можно использовать их уникальные идентификаторы, которые возвращаются при создании с помощью методов, таких как create_line(), create_rectangle() и т. д.

9.Тэги в Tkinter предназначены для идентификации групп элементов на холсте. Они позволяют применять операции к нескольким элементам сразу, например, изменять их цвет, перемещать и т. д. Также тэги могут использоваться для определения, какие элементы будут обрабатываться определенными событиями или действиями.