Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №8**

**дисциплины «Объектно-ориентированное программирование»**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Евдаков Евгений Владимирович  3 курс, группа ИВТ-б-о-22-1,  09.03.01 «Информатика и вычислительная техника», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Проверил:  Воронкин Р. А., доцент департамента цифровых, робототехнических систем и электроники института перспективной инженерии  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2024 г.

**Тема:** обработка событий и рисование в Tkinter.

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

**Ход работы:**

**Задание 1.** Создал общедоступный репозиторий на GitHub, в котором использована лицензий MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами. Клонировал свой репозиторий на свой компьютер. Организовал свой репозиторий в соответствие с моделью ветвления git-flow, появилась новая ветка develop в которой буду выполнять дальнейшие задачи.

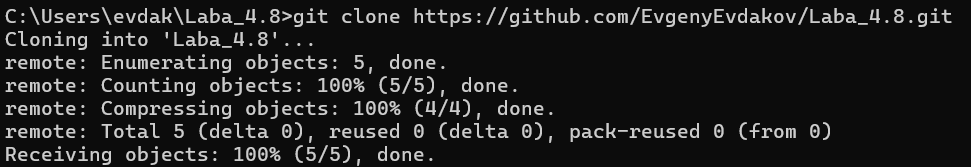


Рисунок 1. Клонирование репозитория

**Задание 2.** Создал файлы poetry.lock и pyproject.toml, также установил необходимые пакеты isort, black, flake8, mypy, pre-commit, pytest.

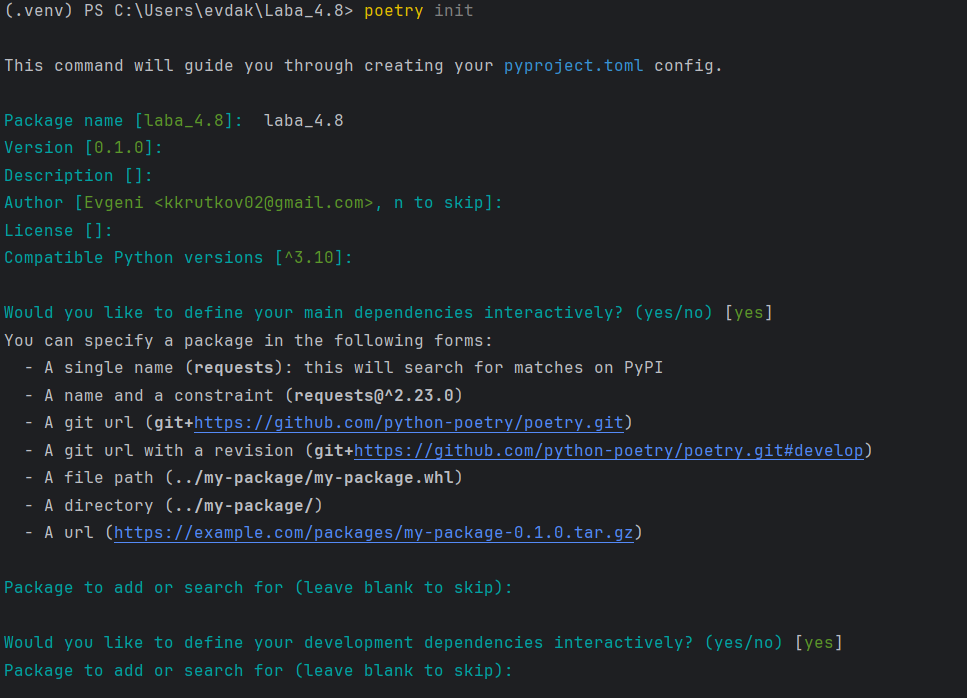


Рисунок 2. Создание виртуального окружения

**Задание 3.** Создал проект PyCharm в папке репозитория.Приступил к работе с примером.Добавил новый файл primer1.py.

**Условие примера:** изучить виджет Listbox.

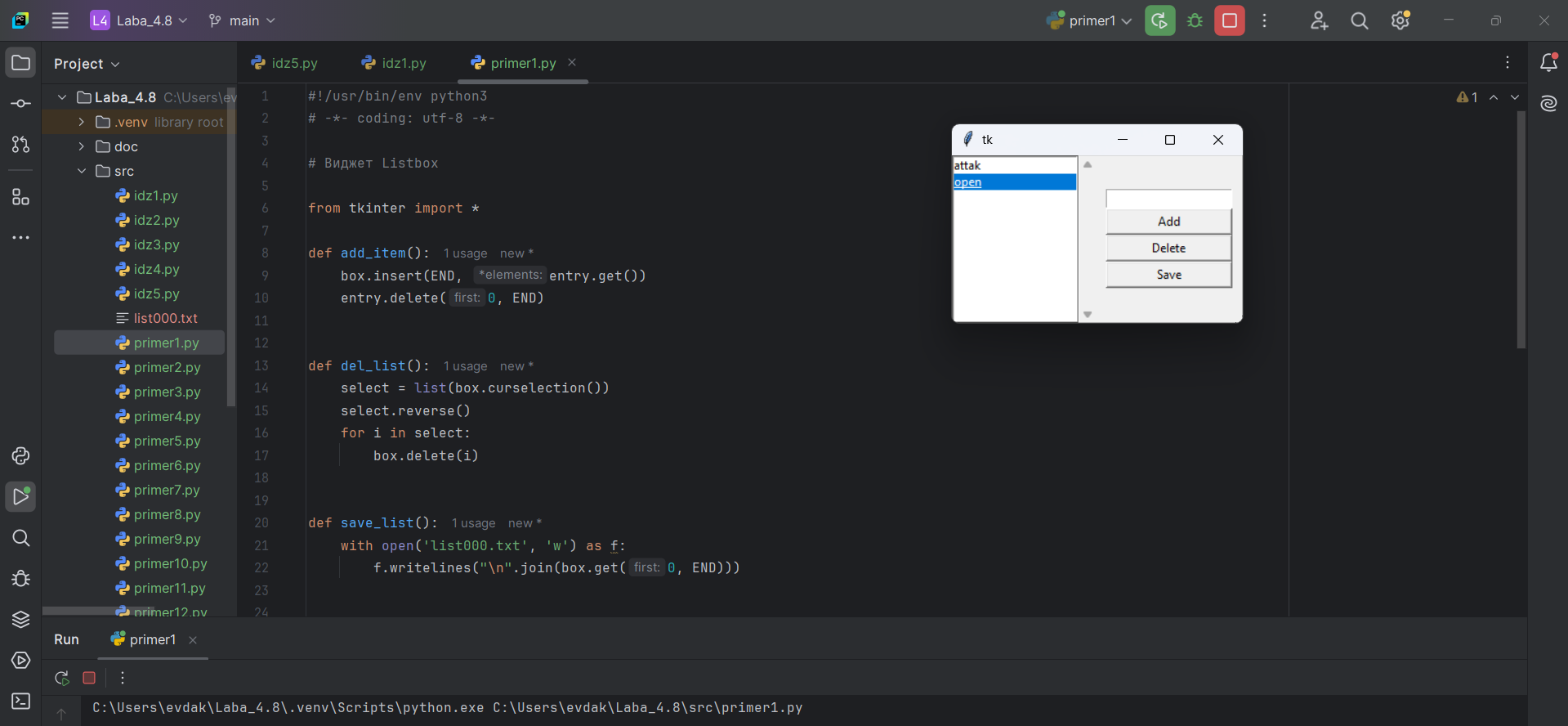


Рисунок 3. Выполнение первого примера

Добавил новый файл primer2.py.

**Условие примера:** изучить метод bind.

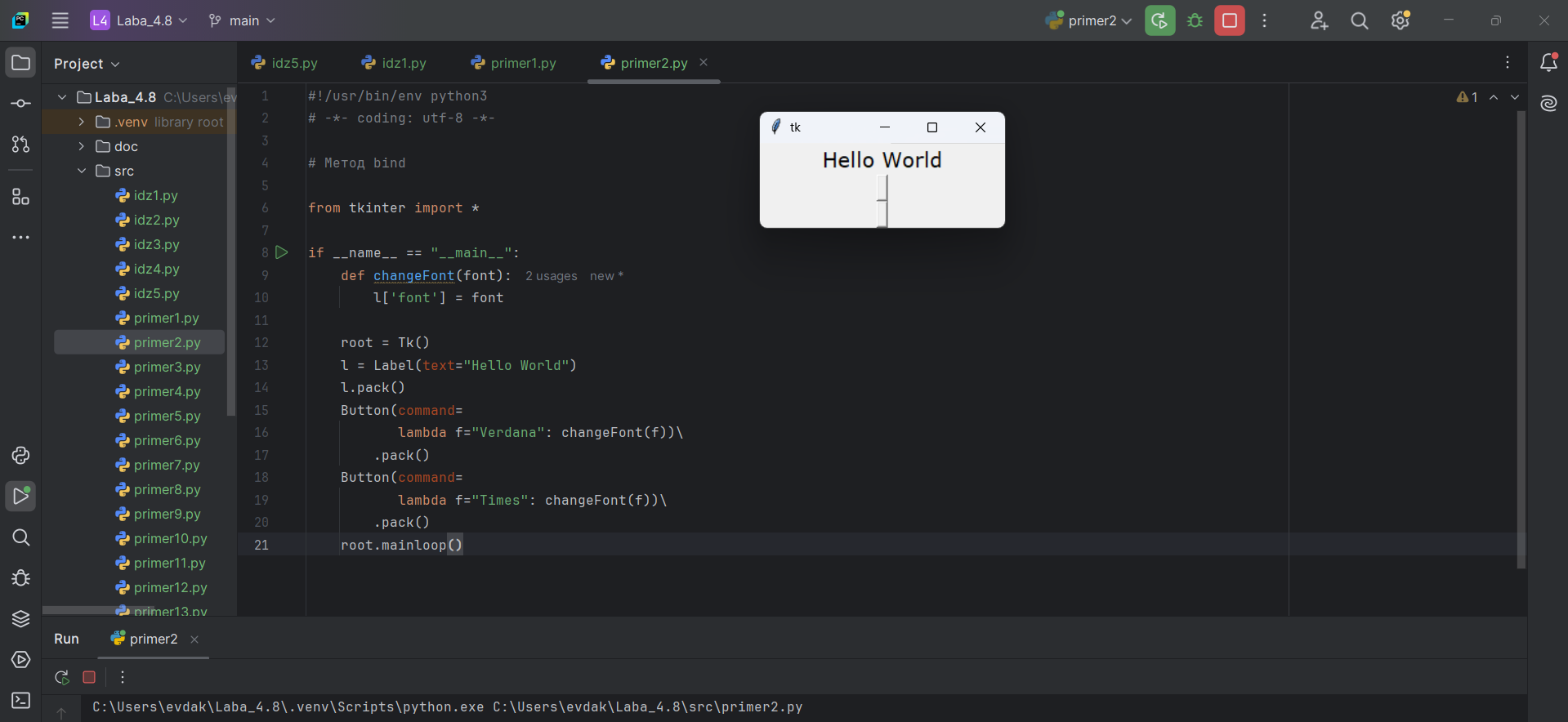


Рисунок 4. Выполнение второго примера

Добавил новый файл primer3.py.

**Условие примера:** изучение событий.

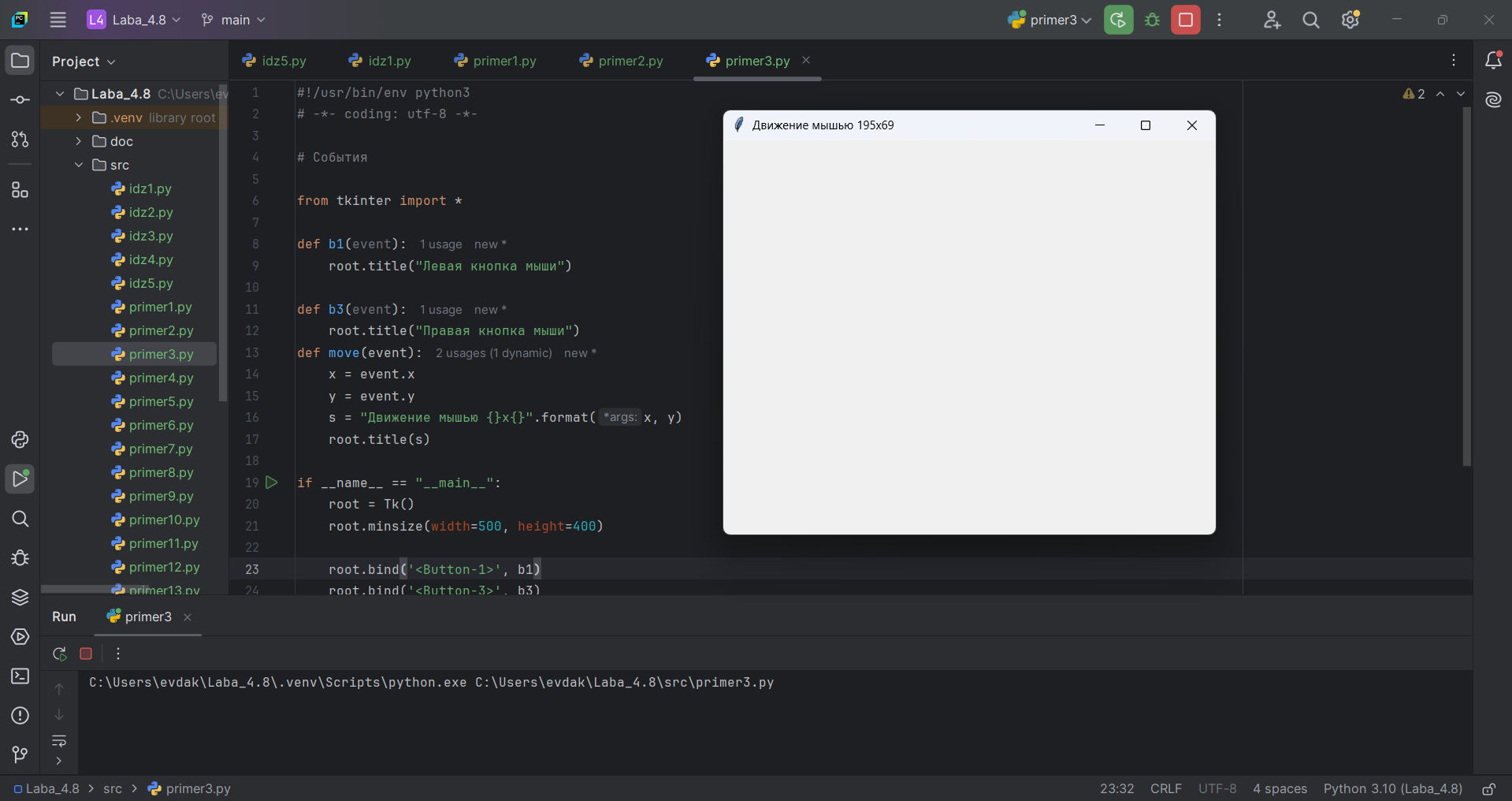


Рисунок 5. Выполнение третьего примера

Добавил новый файл primer4.py.

**Условие примера:** изучение событий.

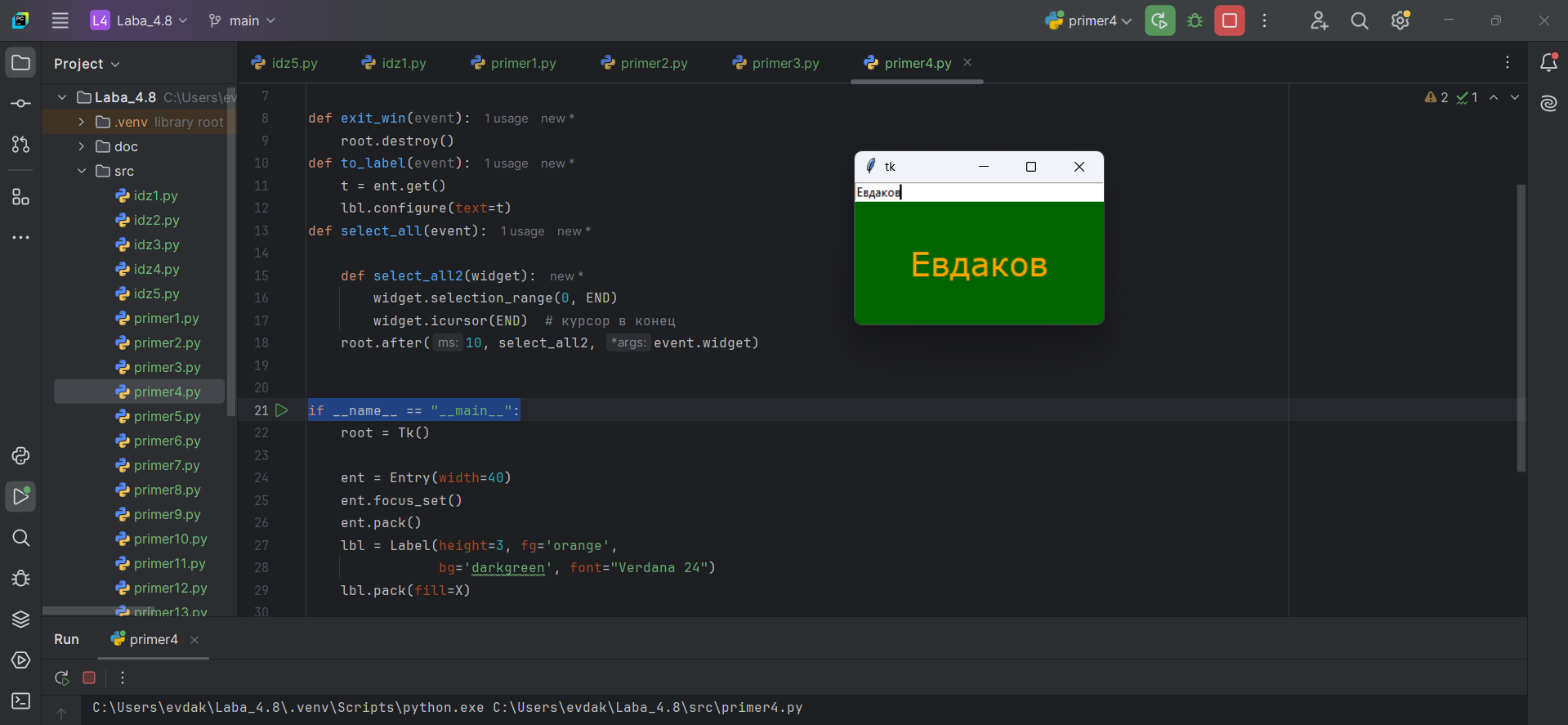


Рисунок 6. Выполнение четвертого примера

Добавил новый файл primer5.py.

**Условие примера:** в программе создается холст. На нем с помощью метода create\_line рисуются отрезки.

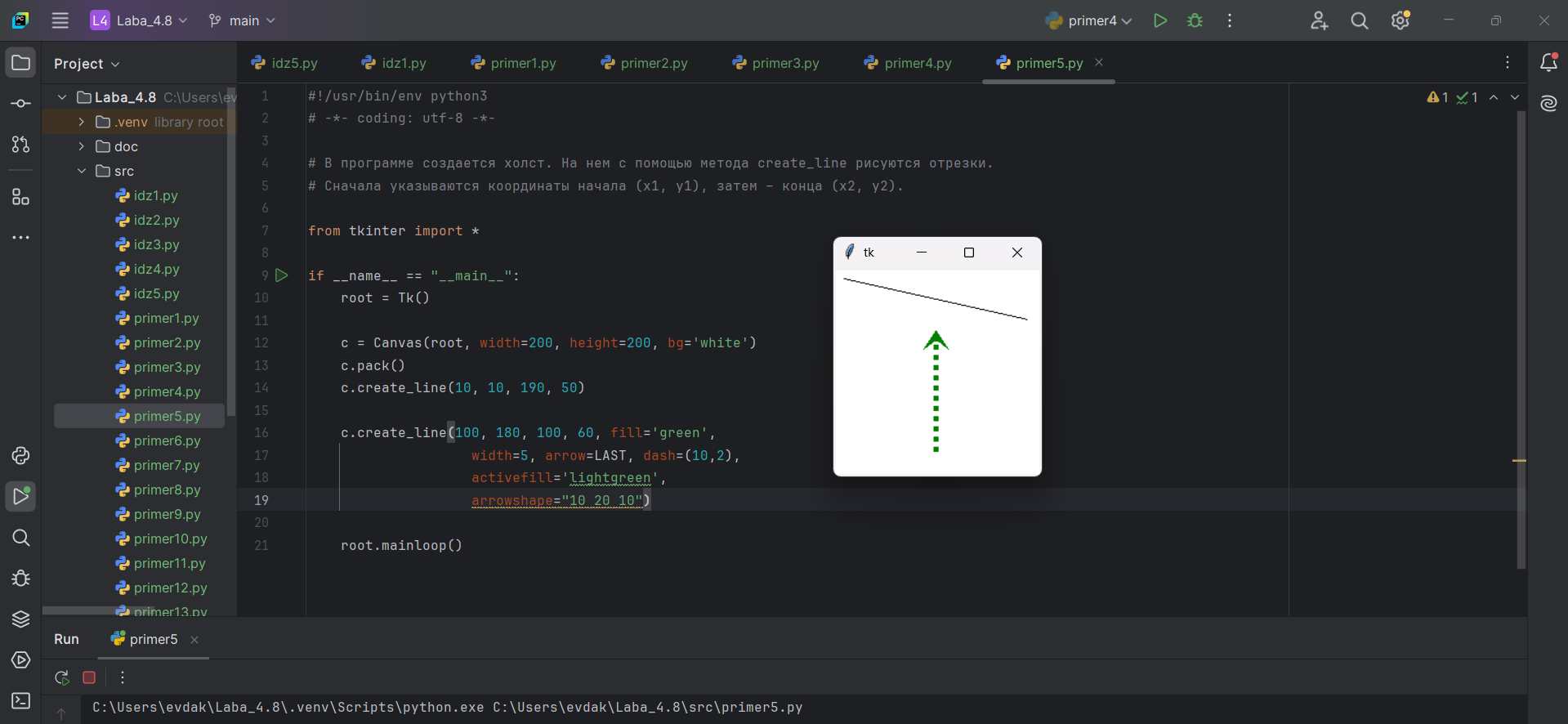


Рисунок 7. Выполнение пятого примера

Добавил новый файл primer6.py.

**Условие примера:** создание прямоугольников методом create\_rectangle.

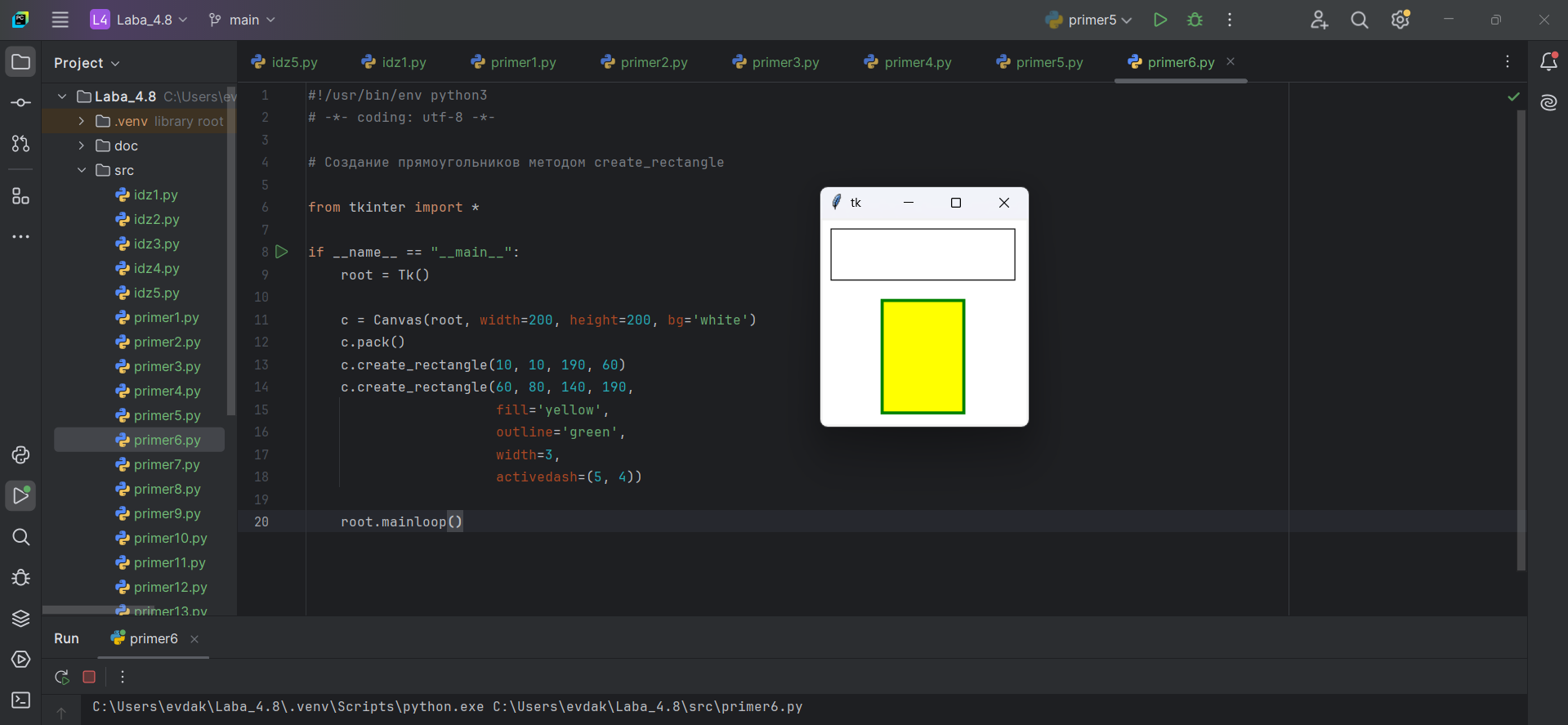


Рисунок 8. Выполнение шестого примера

Добавил новый файл primer7.py.

**Условие примера:** методом create\_polygon рисуется произвольный многоугольник путем задания координат каждой его точки.

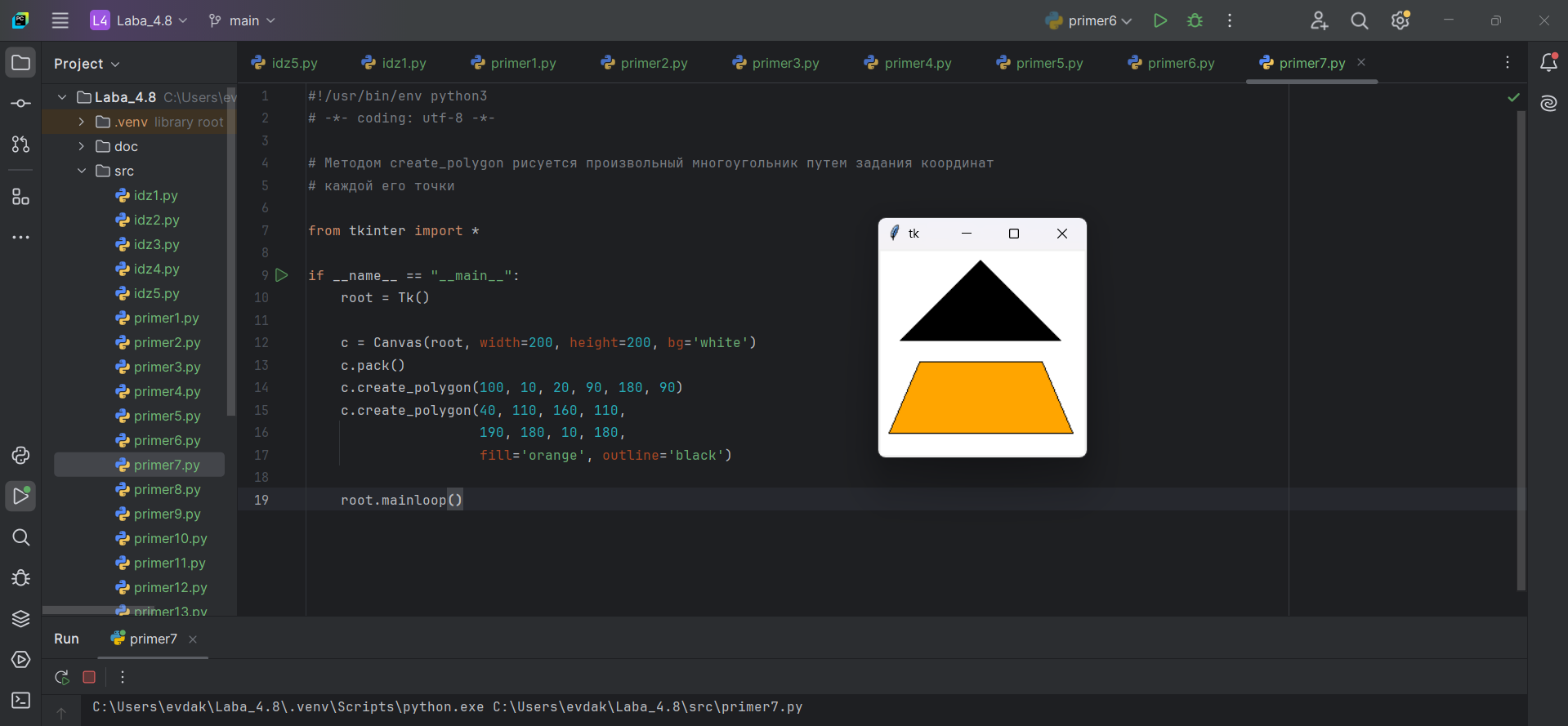


Рисунок 9. Выполнение седьмого примера

Добавил новый файл primer8.py.

**Условие примера:** метод create\_oval создает эллипсы. При этом задаются координаты гипотетического прямоугольника, описывающего эллипс.

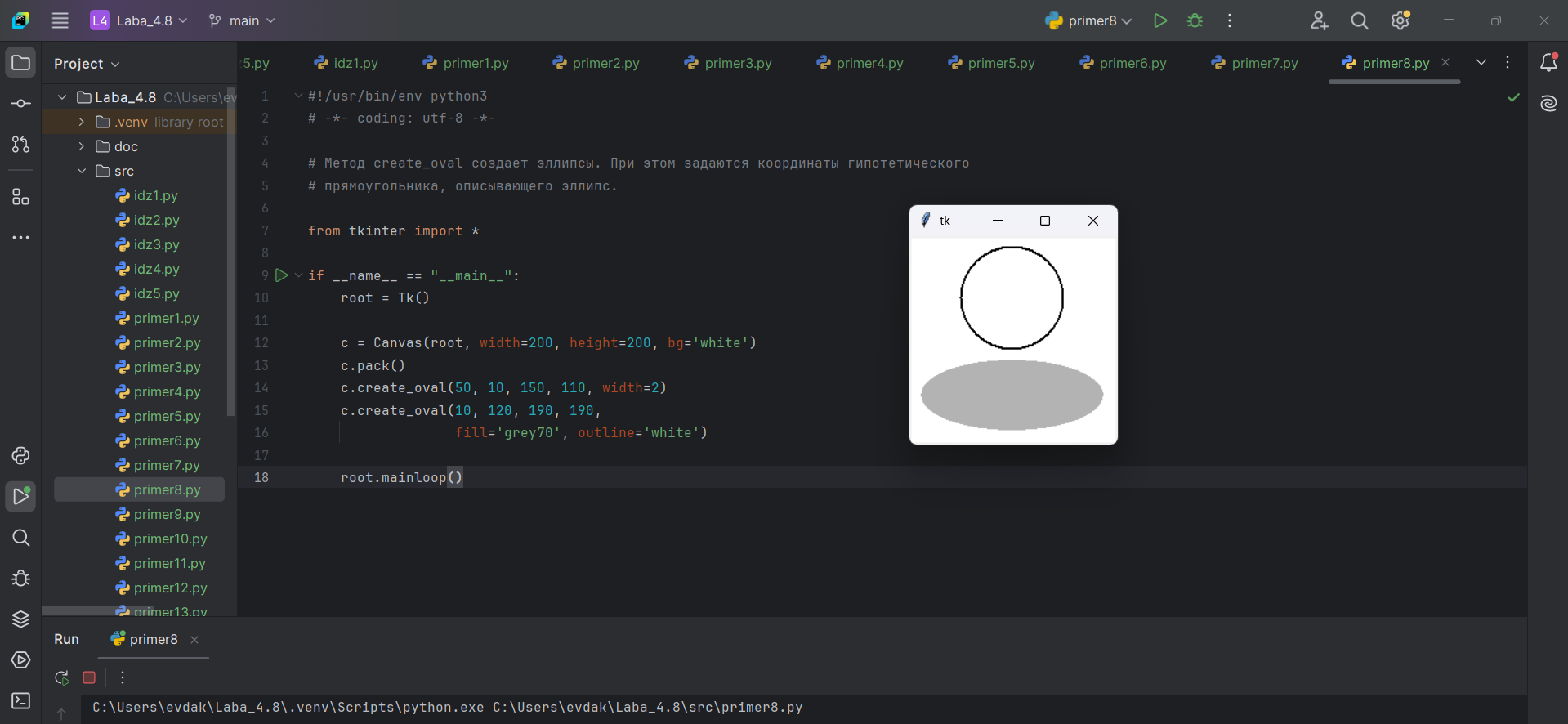


Рисунок 10. Выполнение восьмого примера

Добавил новый файл primer9.py.

**Условие примера:** опции start присваивается градус начала фигуры, extent определяет угол поворота.

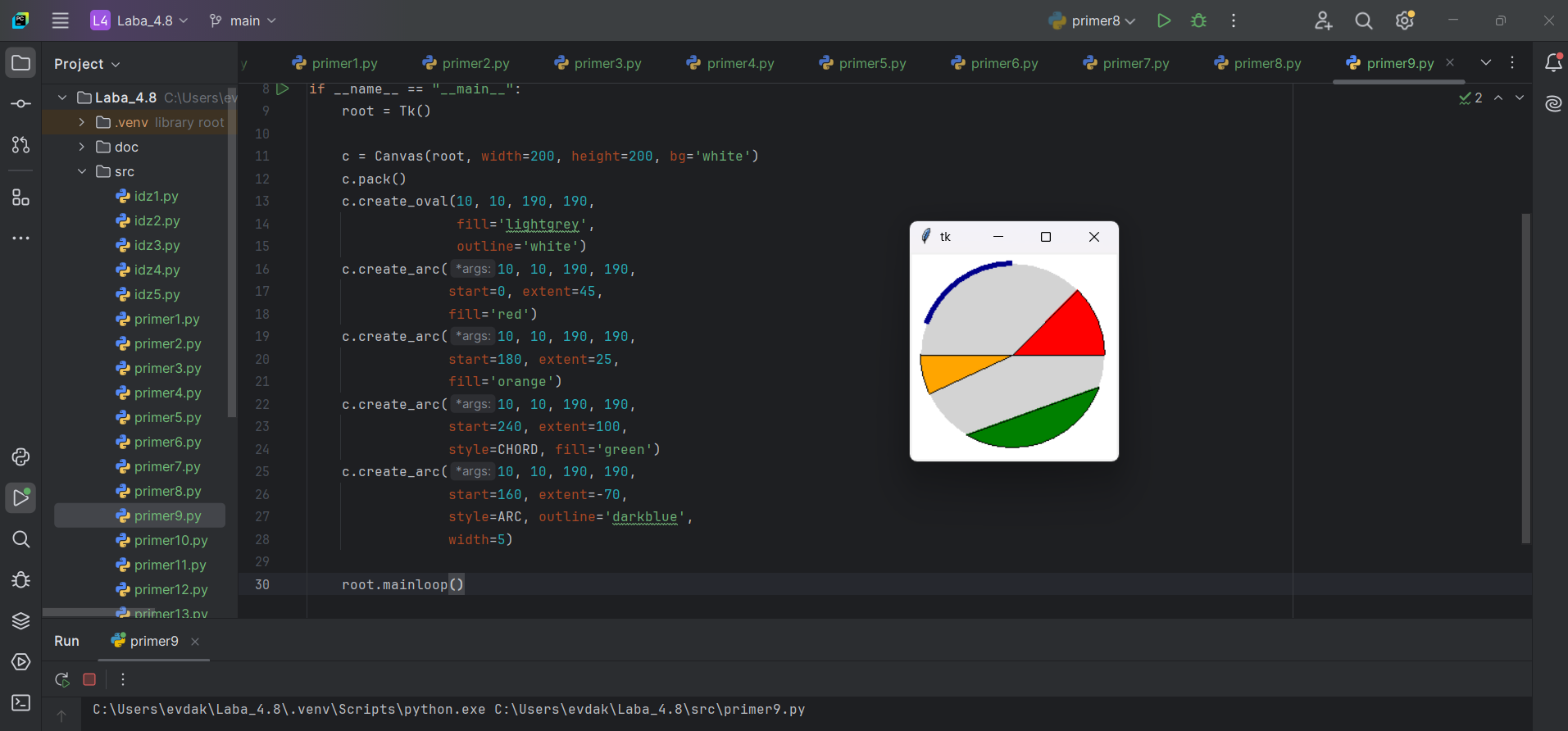


Рисунок 11. Выполнение девятого примера

Добавил новый файл primer10.py.

**Условие примера:** на холсте можно разместить текст. Делается это с помощью метода create\_text

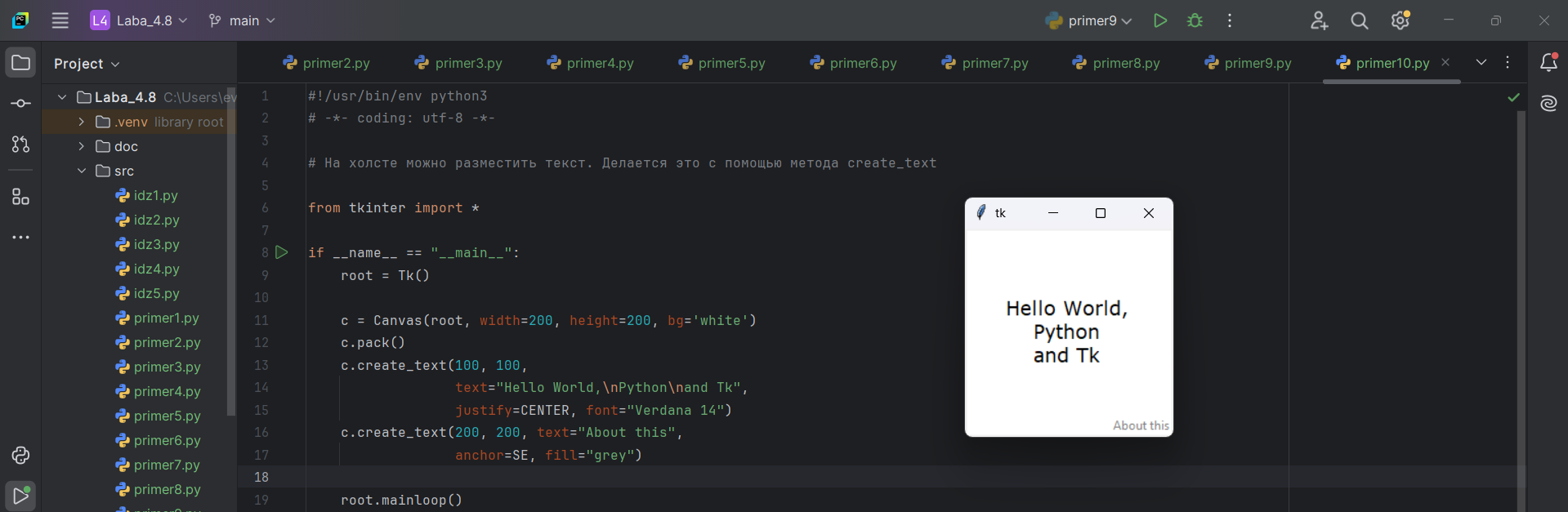


Рисунок 12. Выполнение десятого примера

Добавил новый файл primer11.py.

**Условие примера:** изучение идентификаторов.

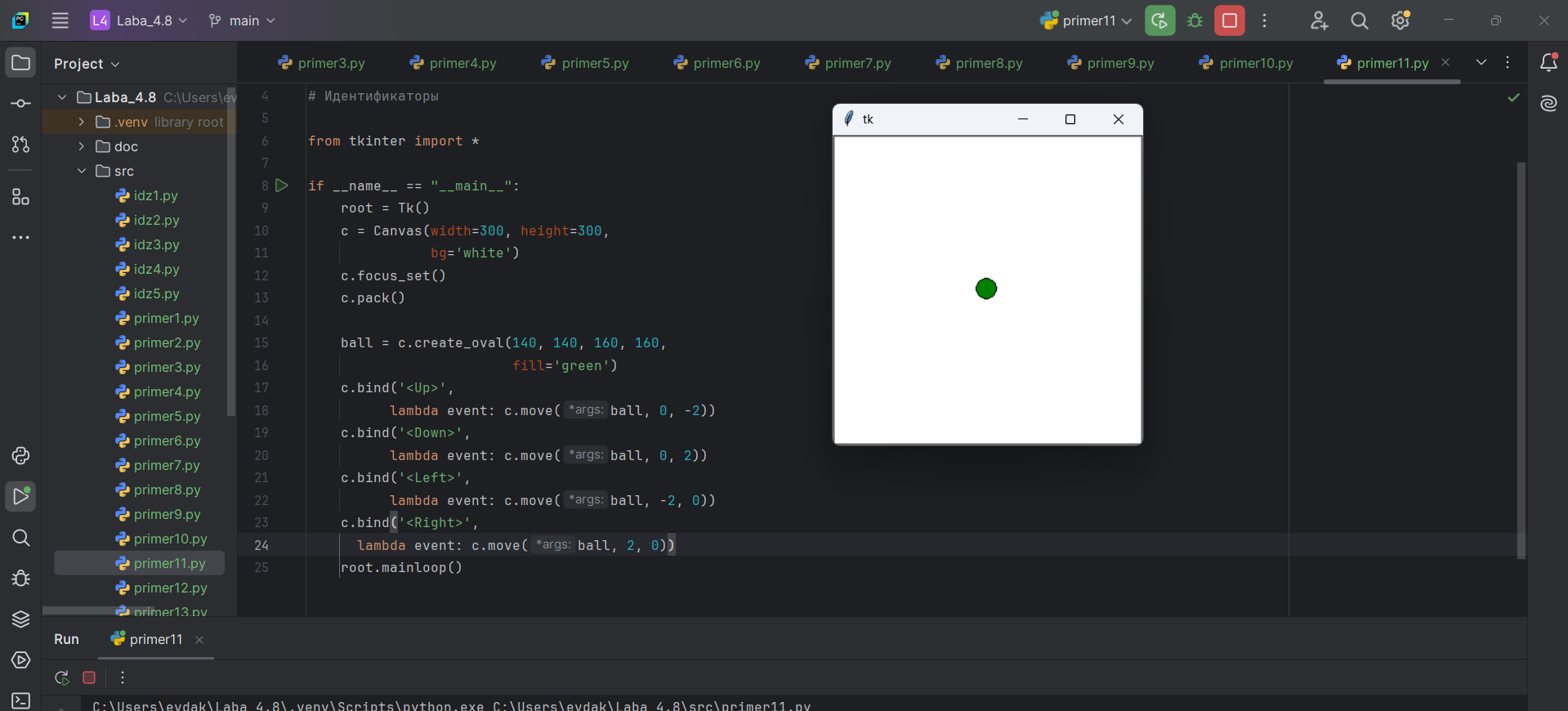


Рисунок 13. Выполнение одиннадцатого примера

Добавил новый файл primer12.py.

**Условие примера:** изучение идентификаторов.

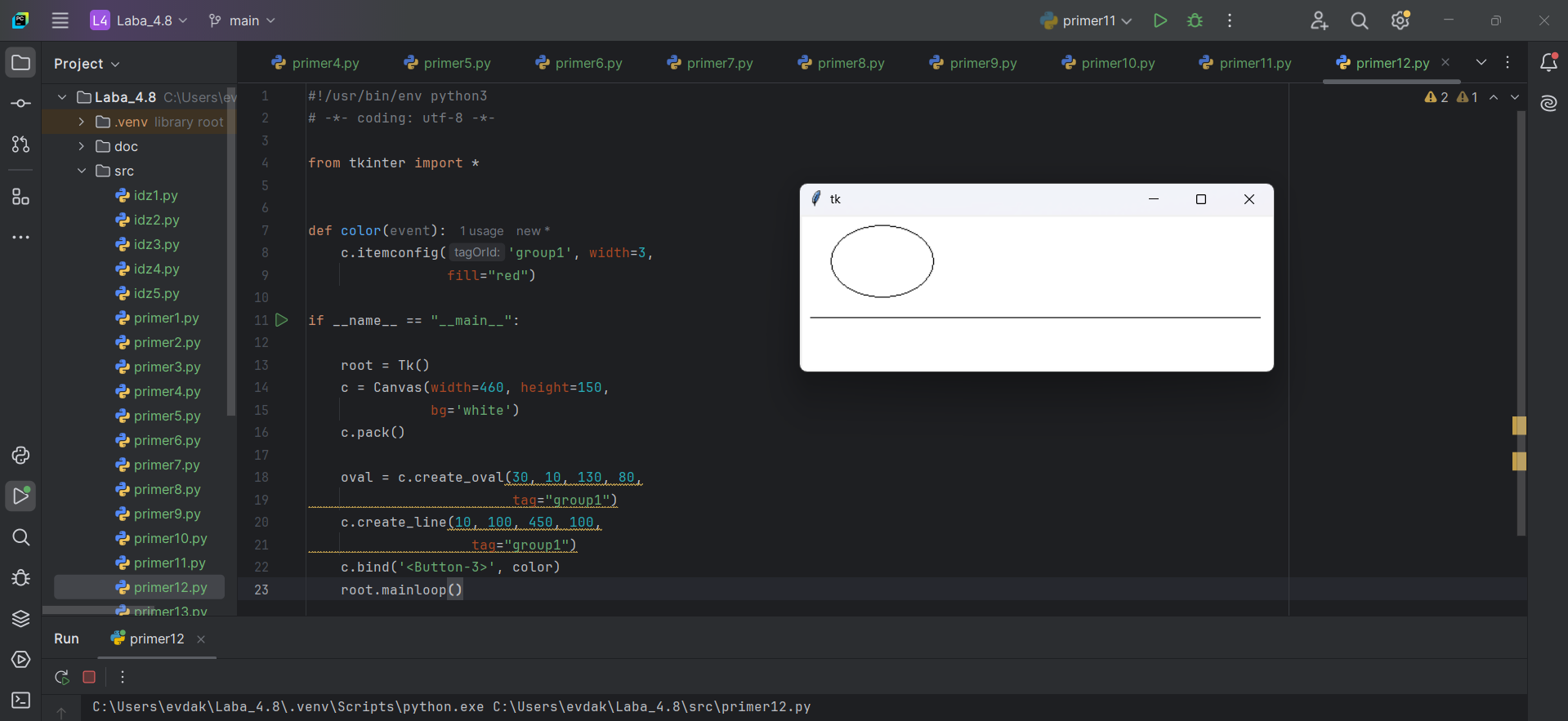


Рисунок 14. Выполнение двенадцатого примера

Добавил новый файл primer13.py.

**Условие примера:** изучение идентификаторов.

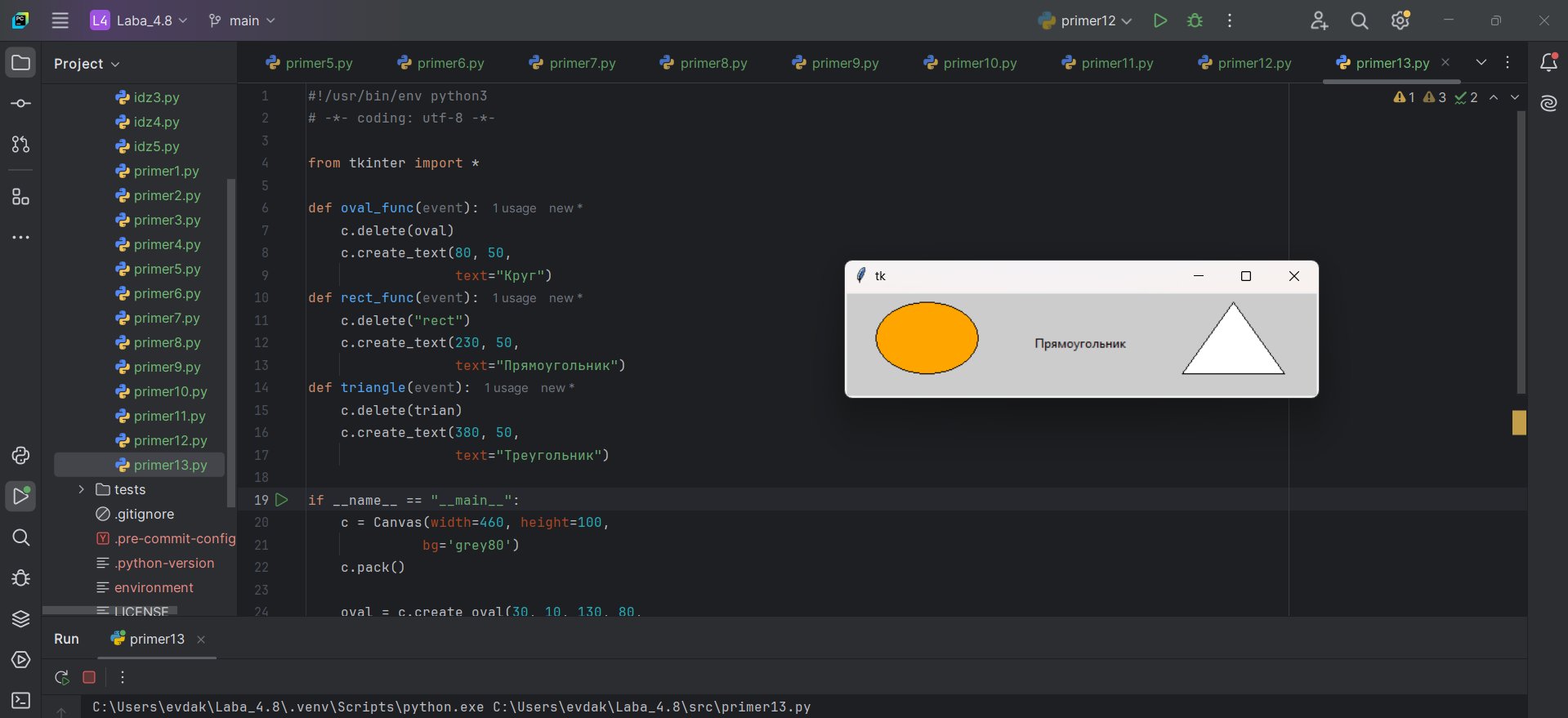


Рисунок 15. Выполнение тринадцатого примера

**Задание 4.**

**Индивидуальное задание**

**Вариант 6**

Создал новый файл под названием idz1.py.

**Условие задания:** необходимо написать 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). Необходимо предусмотреть возможность множественного выбора элементов списка и их перемещения.

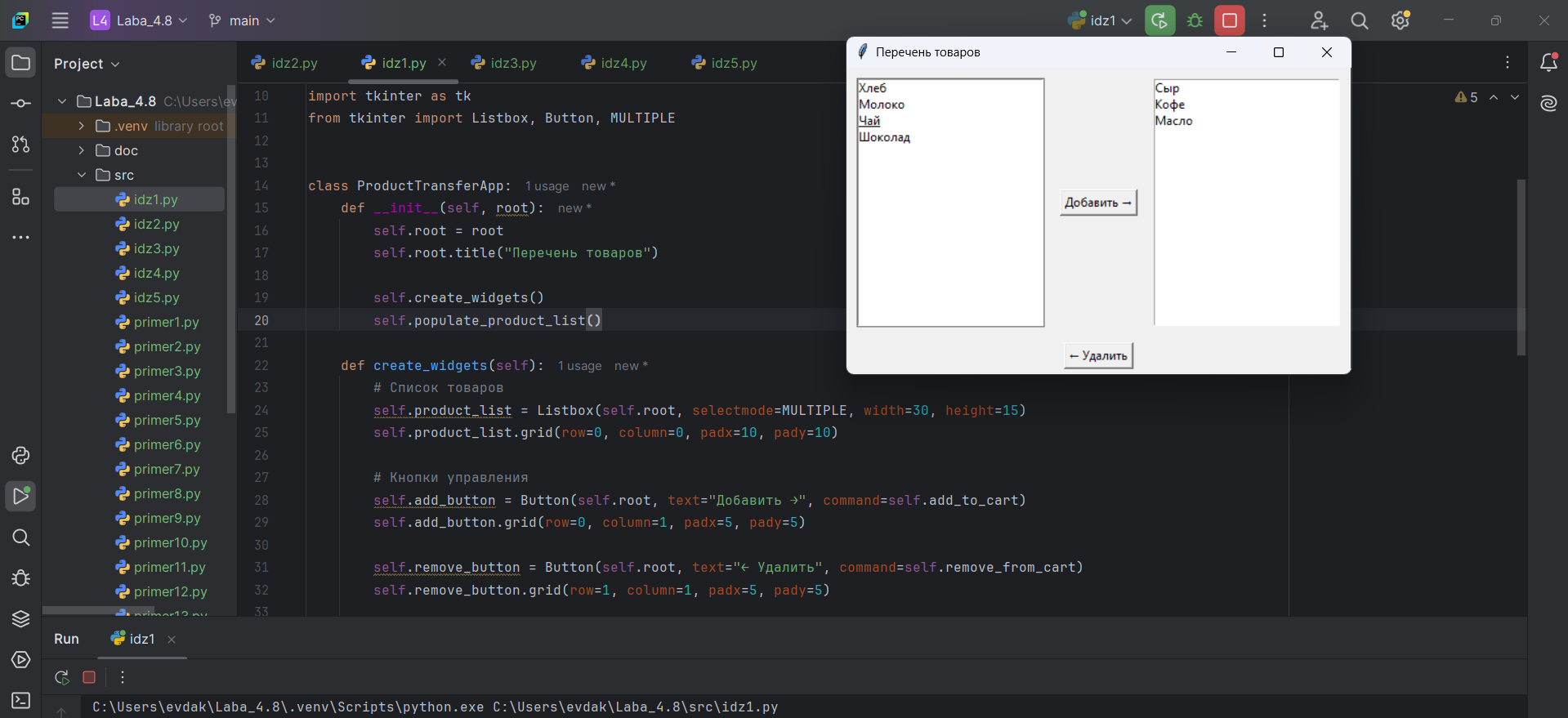
****

Рисунок 16. Код первого индивидуального задания

Создал новый файл под названием idz2.py.

**Условие задания:** необходимо написать 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.

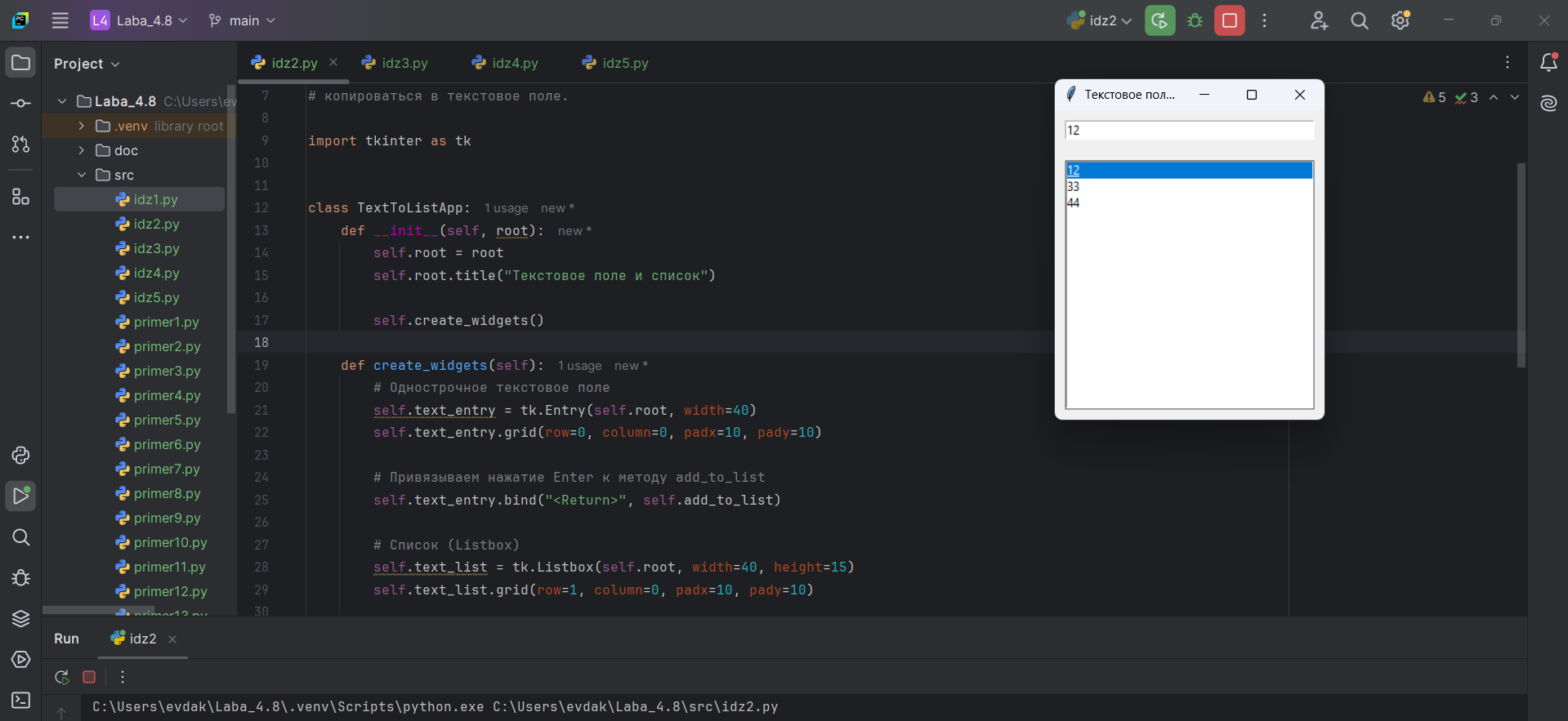


Рисунок 17. Код второго индивидуального задания

Создал новый файл под названием idz3.py.

**Условие задания:** необходимо написать 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>.

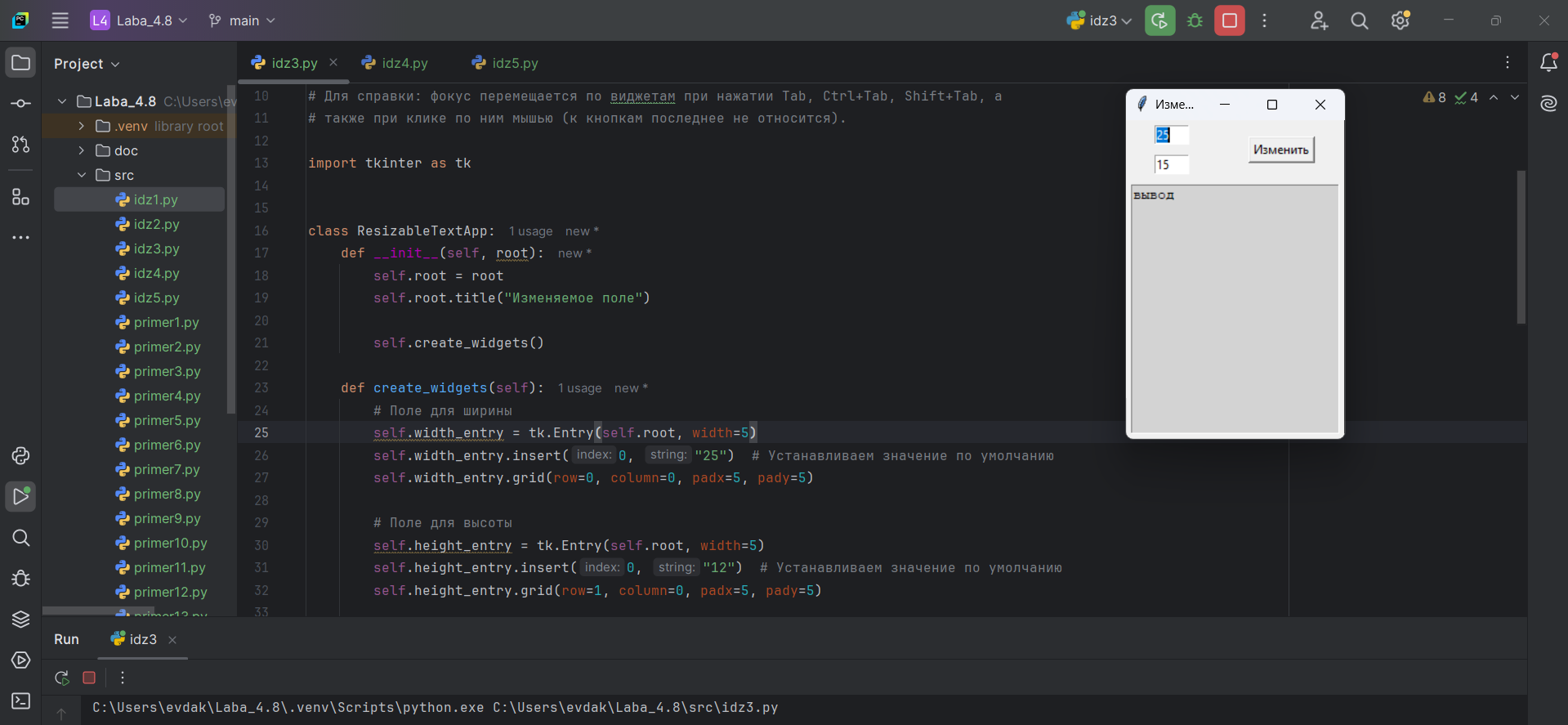


Рисунок 18. Код третьего индивидуального задания

Создал новый файл под названием idz4.py.

**Условие задания:** необходимо создать на холсте изображение. Для создания травы используется цикл.

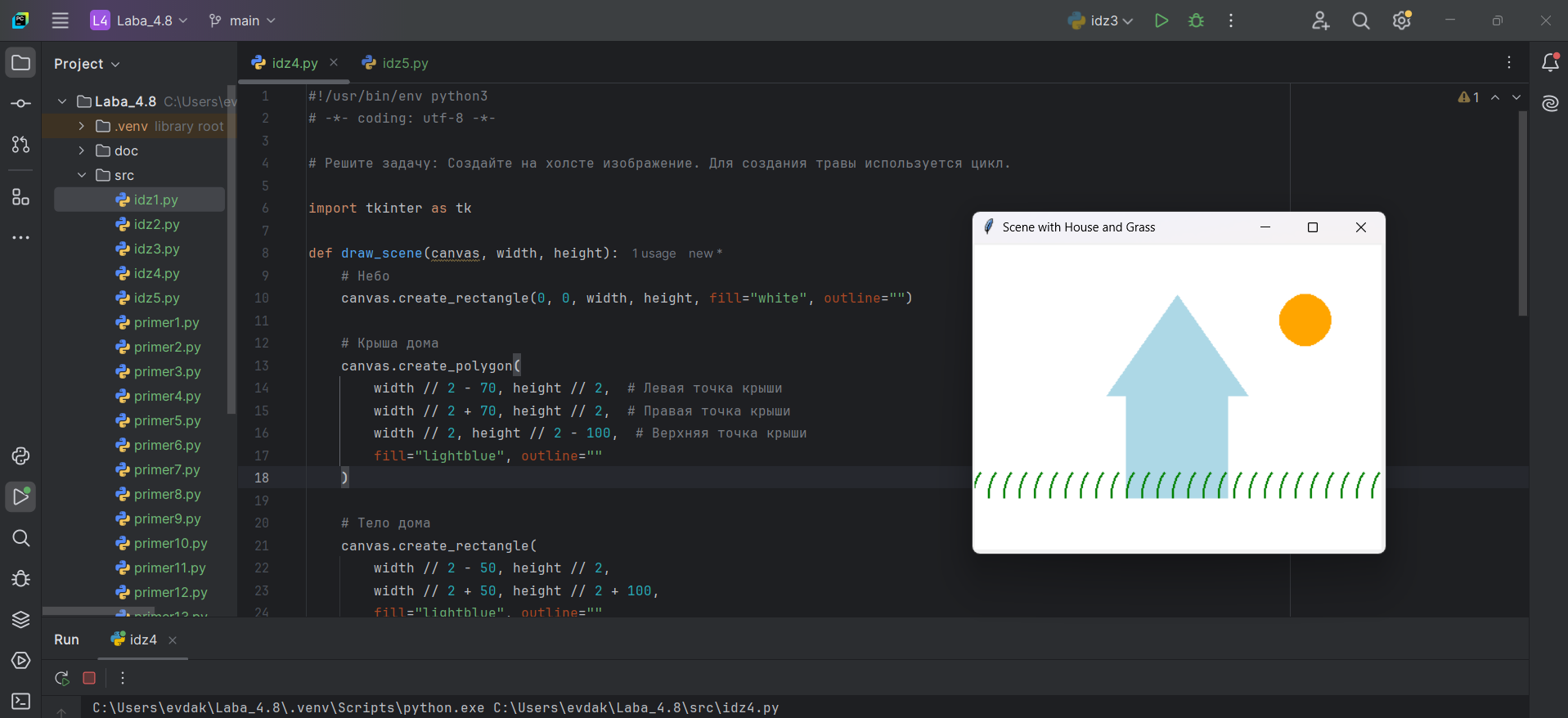


Рисунок 19. Код четвертого индивидуального задания

Создал новый файл под названием idz5.py.

**Условие задания:** необходимо запрограммировать 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).

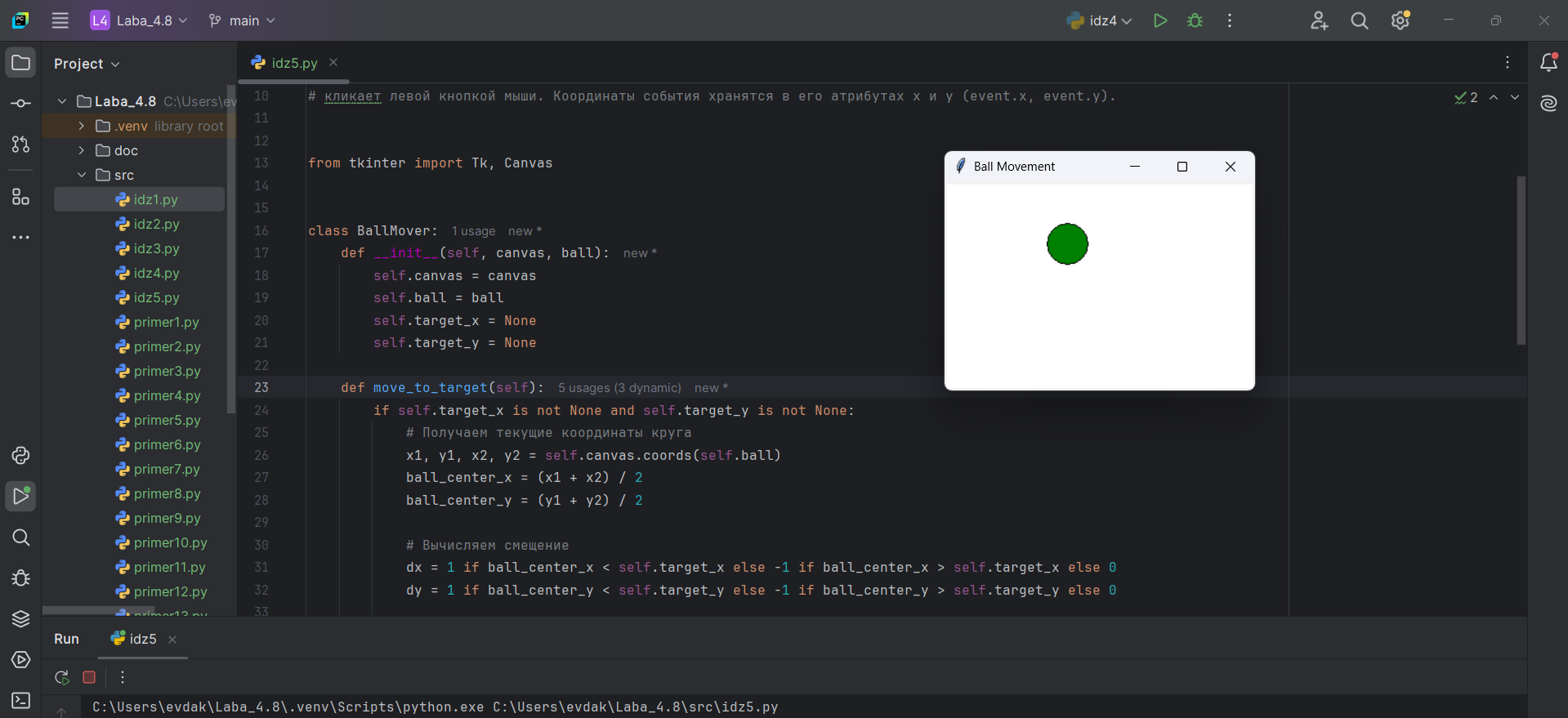


Рисунок 20. Код пятого индивидуального задания

**Задание 5.**

После выполнения работы на ветке develop, слил ее с веткой main и отправил изменения на удаленный сервер.

****

Рисунок 21. Готовый репозиторий

Ссылка: https://github.com/EvgenyEvdakov/Laba\_4.8

**Ответы на контрольные вопросы:**

1. **Каково назначение виджета ListBox?**

ListBox в Tkinter предназначен для отображения списка элементов, из которого пользователь может выбирать один или несколько элементов. Этот виджет часто используется для выбора из заранее заданного набора значений.

1. **Каким образом осуществляется связывание событие или действие с виджетом Tkinter?**

Связывание событий с виджетом осуществляется с помощью метода bind. Синтаксис: widget.bind(event, handler). Event — строка, обозначающая событие (например, нажатие клавиши или щелчок мыши). Handler — функция, которая будет вызвана при наступлении события.

1. **Какие существуют типы событий в Tkinter? Приведите примеры.**

Основные типы событий:

1. События мыши:

<Button-1> — левый клик мыши.

<Button-2> — средний клик мыши.

<Button-3> — правый клик мыши.

<Double-Button-1> — двойной левый клик.

1. События клавиатуры:

<Key> — любое нажатие клавиши.

<KeyPress-a> — нажатие клавиши "a".

<Return> — нажатие клавиши Enter.

1. Системные события:

<Configure> — изменение размера окна.

<Destroy> — уничтожение виджета.

1. **Как обрабатываются события в Tkinter?**

Обработка событий в Tkinter организована через циклический механизм событий. Событие добавляется в очередь, и привязанный обработчик выполняется при возникновении события. Для обработки событий:

1. Создается функция-обработчик (callback).
2. Функция связывается с конкретным виджетом и событием через bind или встроенные методы (например, command для кнопки).
3. Когда событие происходит, вызов функции передается в mainloop.
4. **Как обрабатываются события мыши в Tkinter?**

События мыши обрабатываются через привязку (например, <Button-1> для левого клика).

1. **Каким образом можно отображать графические примитивы в Tkinter?**

Графические примитивы отображаются с использованием виджета Canvas. На холсте можно рисовать линии, круги, прямоугольники и другие фигуры. Методы для создания примитивов:

create\_line — для линий.

create\_oval — для эллипсов и окружностей.

create\_rectangle — для прямоугольников.

create\_polygon — для произвольных многоугольников.

create\_text — для текста.

1. **Перечислите основные методы для отображения графических примитивов в Tkinter.**

Линия: create\_line(x1, y1, x2, y2, ...).

Прямоугольник: create\_rectangle(x1, y1, x2, y2, ...).

Овал/круг: create\_oval(x1, y1, x2, y2, ...).

Многоугольник: create\_polygon(coordinates, ...).

Текст: create\_text(x, y, text=...).

Дуга: create\_arc(x1, y1, x2, y2, ...).

1. **Каким образом можно обратиться к ранее созданным фигурам на холсте?**

Каждая фигура, созданная на холсте, получает уникальный идентификатор. Этот идентификатор можно использовать для обращения к фигуре с помощью методов:

coords(item\_id) — получить или изменить координаты фигуры.

itemconfig(item\_id, options) — изменить свойства фигуры (например, цвет).

delete(item\_id) — удалить фигуру.

1. **Каково назначение тэгов в Tkinter?**

Тэги (tags) используются для группировки и управления несколькими объектами на холсте одновременно. Фигуре можно присвоить один или несколько тэгов, чтобы обращаться к ним как к единой группе.

**Вывод:** приобрел навыки улучшения графического интерфейса пользователя GUI с помощью обработки событий и рисования, реализованных в пакете Tkinter языка программирования Python версии 3.x.