

## Лабораторная работа 4.8


Тема: 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.

Ссылка на GitHub: [https://github.com/RomanGorchakov/Py3\\_9](https://github.com/RomanGorchakov/Py3_9)

### Порядок выполнения работы

1. Создаём новый общедоступный репозиторий, в котором будет использована лицензия MIT и язык программирования Python.

 Start writing code ...

**Start a new repository for RomanGorchakov**

A repository contains all of your project's files, revision history, and collaborator discussion.

**Repository name \***

Py3\_9

✓ Py3\_9 is available.

☒ **Public**  
Anyone on the internet can see this repository

☐ **Private**  
You choose who can see and commit to this repository

Create a new repository

**Introduce yourself with a profile README**

Share information about yourself by creating a profile README, which appears at the top of your profile page.

RomanGorchakov / README.md

Create

```
1 - 🙋 Hi, I'm @RomanGorchakov
2 - 👀 I'm interested in ...
3 - 🌱 I'm currently learning ...
4 - ❤️ I'm looking to collaborate on ...
5 - 💻 How to reach me ...
6 - 😊 Pronouns: ...
7 - ⚡ Fun fact: ...
8
```

Add a license to your project

Apache License 2.0	A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.			<p>To adopt MIT License, enter your details. You'll have a chance to review before committing a <i>LICENSE</i> file to a new branch or the root of your project.</p> <p>Year ⓘ 2024</p> <p>Full name ⓘ RomanGorchakov</p> <p><b>Review and submit</b></p>
GNU General Public License v3.0				
MIT License	<b>Permissions</b> ✓ Commercial use ✓ Modification ✓ Distribution ✓ Private use	<b>Limitations</b> ✗ Liability ✗ Warranty	<b>Conditions</b> ⓘ License and copyright notice	
BSD 2-Clause "Simplified" License				
BSD 3-Clause "New" or "Revised" License				
Boost Software License 1.0				
Creative Commons Zero v1.0 Universal				
Eclipse Public License 2.0				
GNU Affero General Public License v3.0				
GNU General Public License v2.0				

This is not legal advice. [Learn more about repository licenses.](#)

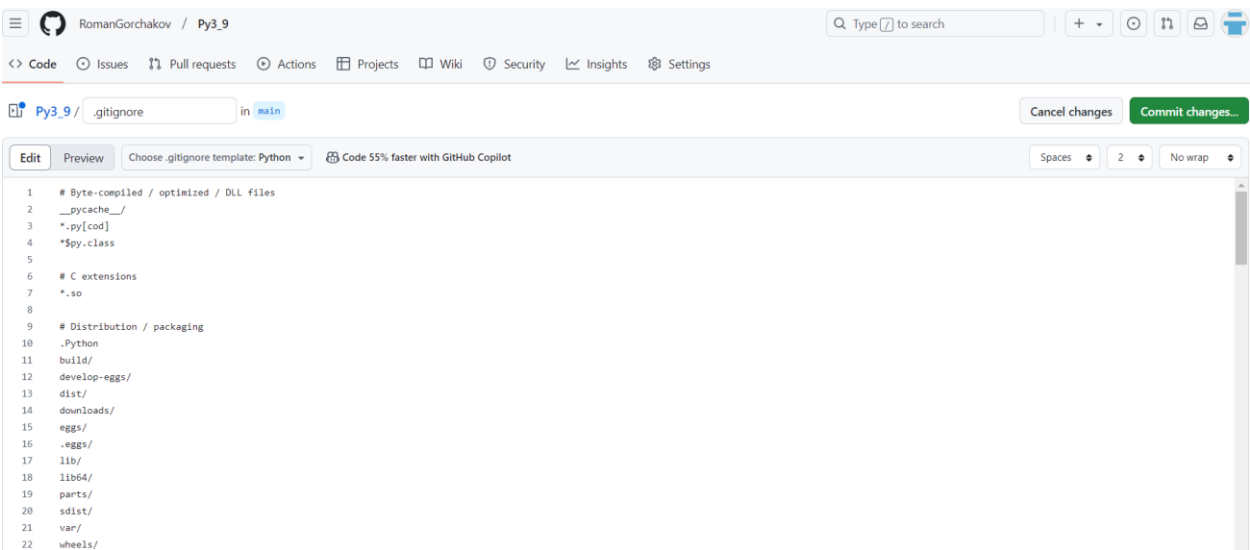
---

MIT License

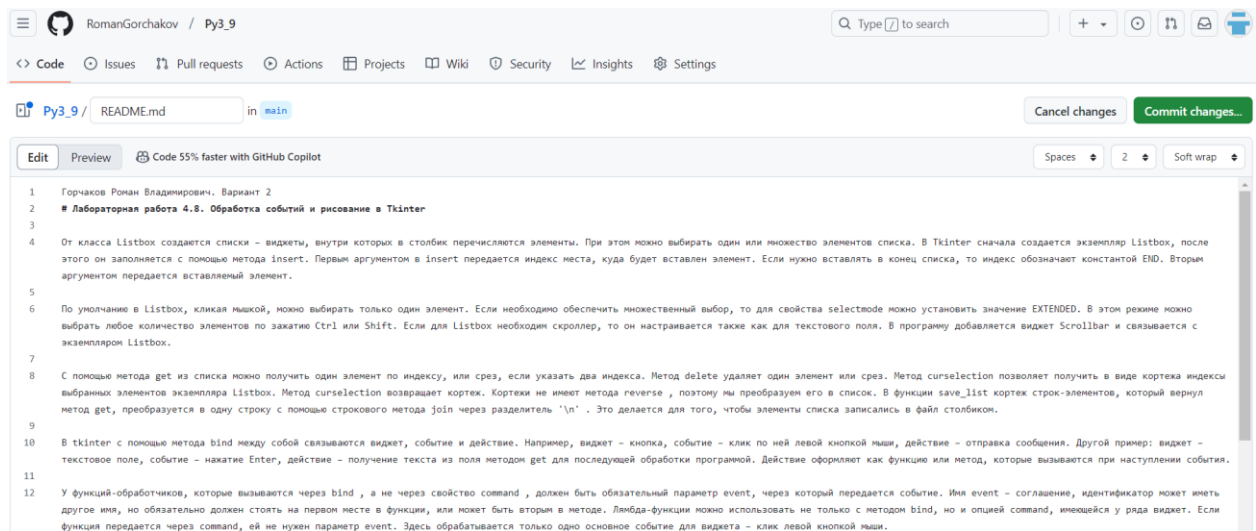
Copyright (c) Year Full name

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

2. Теперь необходимо дополнить файл `.gitignore` с необходимыми правилами для языка программирования Python. Для этого переходим по ссылке «<https://github.com/github/gitignore>» и скачиваем оттуда файл «Python.gitignore».



3. Теперь создаём файл «README.md», где вносим ФИО и теоретический конспект лекции. Сохраняем набранный текст через кнопку «Commit changes».



4. В окне «Codespace» выбираем опцию «Create codespace on main». Откроется терминал, куда мы введём команду «git clone», чтобы клонировать свой репозиторий. После этого организуем репозиторий в соответствии с моделью ветвления Git-flow. Для этого введём в терминал команды: «git checkout –b develop» для создания ветки разработки; «git branch feature\_branch» для создания ветки функций; «git branch release/1.0.0» для создания ветки релиза; «git checkout main» и «git branch hotfix» для создания веток hotfix. Устанавливаем библиотеки isort, black и flake8 и создаём файлы .pre-commit-config.yaml и environment.yml.

```
• @RomanGorchakov →/workspaces/Py3_9 (main) $ git checkout -b develop
Switched to a new branch 'develop'
• @RomanGorchakov →/workspaces/Py3_9 (develop) $ git branch feature_branch
• @RomanGorchakov →/workspaces/Py3_9 (develop) $ git branch release/1.0.0
• @RomanGorchakov →/workspaces/Py3_9 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
• @RomanGorchakov →/workspaces/Py3_9 (main) $ git branch hotfix
• @RomanGorchakov →/workspaces/Py3_9 (main) $ git checkout develop
Switched to branch 'develop'
○ @RomanGorchakov →/workspaces/Py3_9 (develop) $
```

```

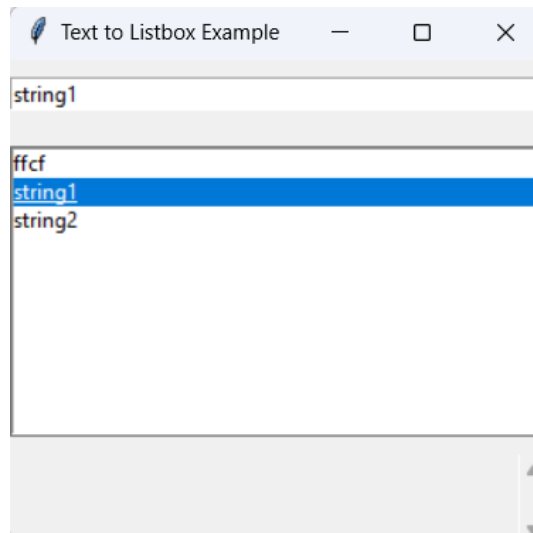
Installing collected packages: isort
Successfully installed isort-5.13.2
• @RomanGorchakov →/workspaces/Py3_9 (develop) $ pip install black
Collecting black
  Downloading black-24.10.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl.metadata (79 kB)
Collecting click>=8.0.0 (from black)
  Downloading click-8.1.8-py3-none-any.whl.metadata (2.3 kB)
Collecting mypy_extensions>=0.4.3 (from black)
  Downloading mypy_extensions-1.0.0-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: packaging>=22.0 in /home/codespace/.local/lib/python3.12/site-packages (from black) (24.2)
Collecting pathspec>=0.9.0 (from black)
  Downloading pathspec-0.12.1-py3-none-any.whl.metadata (21 kB)
Requirement already satisfied: platformdirs>=2 in /home/codespace/.local/lib/python3.12/site-packages (from black) (4.3.6)
Downloading black-24.10.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (1.8 MB)
1.8/1.8 MB 41.6 MB/s eta 0:00:00
Downloading click-8.1.8-py3-none-any.whl (98 kB)
Downloading mypy_extensions-1.0.0-py3-none-any.whl (4.7 kB)
Downloading pathspec-0.12.1-py3-none-any.whl (31 kB)
Installing collected packages: pathspec, mypy_extensions, click, black
Successfully installed black-24.10.0 click-8.1.8 mypy_extensions-1.0.0 pathspec-0.12.1
• @RomanGorchakov →/workspaces/Py3_9 (develop) $ pip install flake8
Collecting flake8
  Downloading flake8-7.1.1-py2.py3-none-any.whl.metadata (3.8 kB)
Collecting mccabe<0.8.0,>=0.7.0 (from flake8)
  Downloading mccabe-0.7.0-py2.py3-none-any.whl.metadata (5.0 kB)
Collecting pycodestyle<2.13.0,>=2.12.0 (from flake8)
  Downloading pycodestyle-2.12.1-py2.py3-none-any.whl.metadata (4.5 kB)
Collecting pyflakes<3.3.0,>=3.2.0 (from flake8)
  Downloading pyflakes-3.2.0-py2.py3-none-any.whl.metadata (3.5 kB)
Downloading flake8-7.1.1-py2.py3-none-any.whl (57 kB)
Downloading mccabe-0.7.0-py2.py3-none-any.whl (7.3 kB)
Downloading pycodestyle-2.12.1-py2.py3-none-any.whl (31 kB)
Downloading pyflakes-3.2.0-py2.py3-none-any.whl (62 kB)
Installing collected packages: pyflakes, pycodestyle, mccabe, flake8
Successfully installed flake8-7.1.1 mccabe-0.7.0 pycodestyle-2.12.1 pyflakes-3.2.0

```

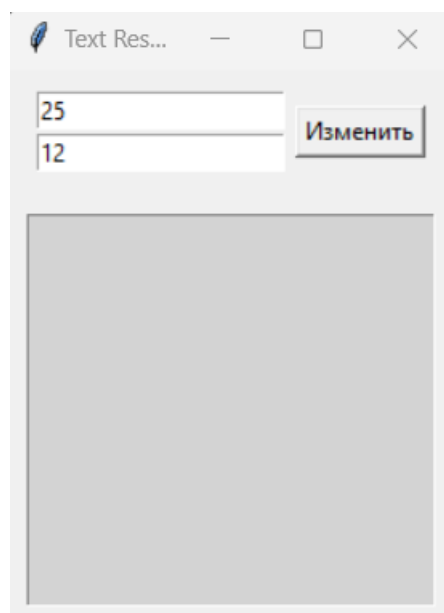
5. Создаём файл «task1.py», в котором нужно написать программу, состоящую из двух списков Listbox. В первом будет, например, перечень товаров, заданный программно. Второй изначально пуст, пусть это будет перечень покупок. При клике на одну кнопку товар должен переходить из одного списка в другой. При клике на вторую кнопку – возвращаться (человек передумал покупать).



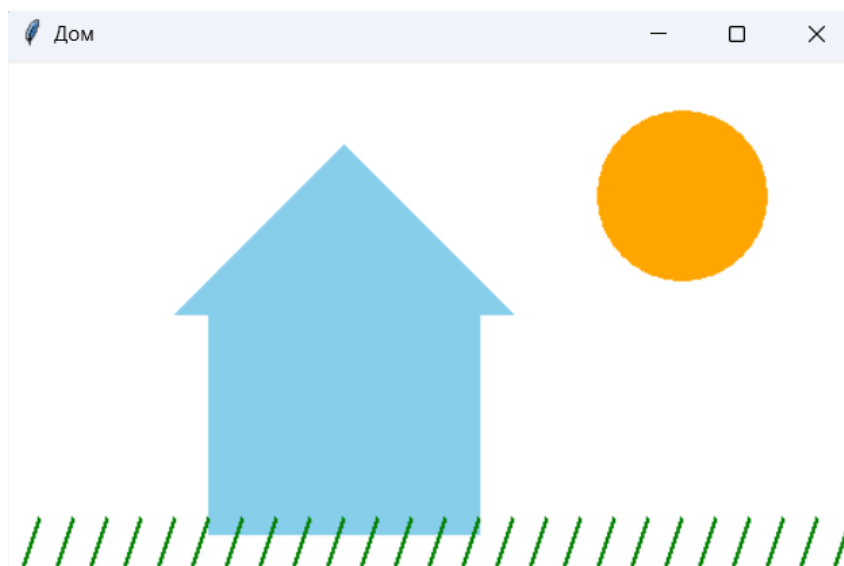
6. Создаём файл «task2.py», в котором нужно написать программу. Нажатие Enter в однострочном текстовом поле приводит к перемещению текста из него в список (экземпляр Listbox). При двойном клике по элементу-строке списка, она должна копироваться в текстовое поле.



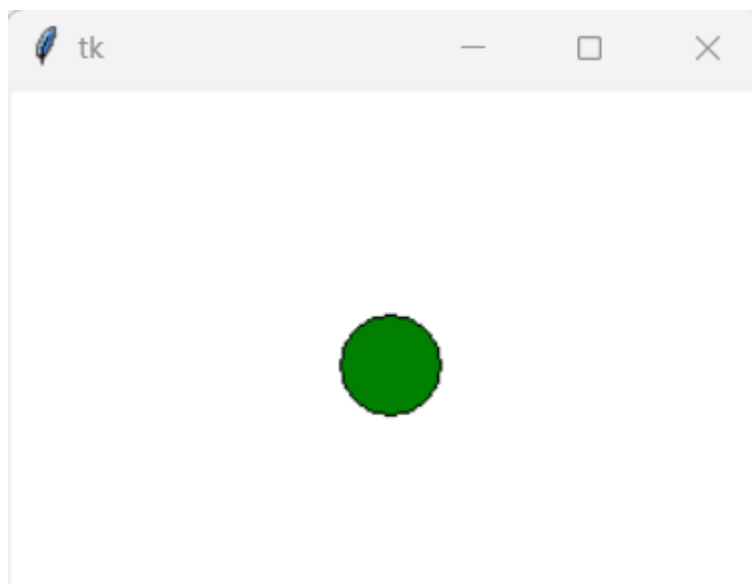
7. Создаём файл «task3.py», в котором нужно написать программу. Размеры многострочного текстового поля определяются значениями, введенными в однострочные текстовые поля. Изменение размера происходит при нажатии мышью на кнопку, а также при нажатии клавиши Enter. Цвет фона экземпляра Text светлосерый (lightgrey), когда поле не в фокусе, и белый, когда имеет фокус.



8. Создаём файл «task4.py», в котором нужно создать на холсте изображение, показанное в примере.



9. Создаём файл «task5.py», в котором нужно запрограммировать постепенное движение фигуры в ту точку холста, где пользователь кликает левой кнопкой мыши.



10. Выполняем коммит файлов в репозиторий Git в ветку разработки, сливаем её с веткой main и отправляем изменения на сервер GitHub.

```
pyproject.toml
setup.cfg
uv.lock
.../\320\227\320\260\320\264\320\260\320\275\320\270\320\265 1/task1.py"
.../\320\227\320\260\320\264\320\260\320\275\320\270\320\265 2/task2.py"
.../\320\227\320\260\320\264\320\260\320\275\320\270\320\265 3/task3.py"
.../\320\227\320\260\320\264\320\260\320\275\320\270\320\265 4/task4.py"
.../\320\227\320\260\320\264\320\260\320\275\320\270\320\265 5/task5.py"
.../\320\2404.6_\320\223\320\276\321\200\321\207\320\260\320\272\320\276\320\262\320\240\320\222.pdf"
13 files changed, 836 insertions(+)
create mode 100644 .pre-commit-config.yaml
create mode 100644 .python-version
create mode 100644 edu.pyoop.code-workspace
create mode 100644 environment.yml
create mode 100644 pyproject.toml
create mode 100644 setup.cfg
create mode 100644 uv.lock
create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217\320\227\320\260\320\264\320\260\320\275\320\270\320\265 1/task1.py"
create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217\320\227\320\260\320\264\320\260\320\275\320\270\320\265 2/task2.py"
create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217\320\227\320\260\320\264\320\260\320\275\320\270\320\265 3/task3.py"
create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217\320\227\320\260\320\264\320\260\320\275\320\270\320\265 4/task4.py"
create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217\320\227\320\260\320\264\320\260\320\275\320\270\320\265 5/task5.py"
create mode 100644 "\320\236\321\202\321\207\321\221\321\202\320\233\320\2404.6_\320\223\320\276\321\200\321\207\320\272\320\276\320\262\320\240\320\222.pdf"
• @RomanGorchakov →/workspaces/Py3_9 (main) $ git push -u
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 2 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (22/22), 982.51 KiB | 16.38 MiB/s, done.
Total 22 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/RomanGorchakov/Py3_9
e9ba630..8063013 main -> main
branch 'main' set up to track 'origin/main'.
• @RomanGorchakov →/workspaces/Py3_9 (main) $
```

Py3\_9Public

PinUnwatch1Fork0Star0

main1 Branch0 Tags

Go to fileAdd fileCode

About

RomanGorchakovCanvas8063013 · now5 Commits

Задания	Canvas	now
Отчёт	Canvas	now
.gitignore	Create .gitignore	yesterday
.pre-commit-config.yaml	Canvas	now
.python-version	Canvas	now
LICENSE	Create LICENSE	yesterday
README.md	Create README.md	yesterday
edu.pyoop.code-workspace	Canvas	now
environment.yml	Canvas	now
pyproject.toml	Canvas	now
setup.cfg	Canvas	now
uv.lock	Canvas	now

READMEMIT license

Горчаков Роман Владимирович. Вариант 2

Лабораторная работа 4.8. Обработка событий и рисование в Tkinter

No description, website, or topics provided.

ReadmeMIT licenseActivity0 stars1 watching0 forks

ReleasesNo releases publishedCreate a new release

PackagesNo packages publishedPublish your first package

Suggested workflowsBased on your tech stack

Publish Python PackageConfigure

PylintConfigure

DjangoConfigure

## Контрольные вопросы

### 1. Каково назначение виджета ListBox?

От класса `Listbox` создаются списки – это виджеты, внутри которых в столбик перечисляются элементы. При этом можно выбирать один или множество элементов списка

2. Каким образом осуществляется связывание события или действия с виджетом Tkinter?

В `tkinter` с помощью метода `bind` между собой связываются виджет, событие и действие. Например, виджет – кнопка, событие – клик по ней левой кнопкой мыши, действие – отправка сообщения. Другой пример: виджет – текстовое поле, событие – нажатие `Enter`, действие – получение текста из поля методом `get` для последующей обработки программой. Действие оформляют как функцию или метод, которые вызываются при наступлении события.

3. Какие существуют типы событий в Tkinter? Приведите примеры.

Можно выделить три основных типа событий: производимые мышью (щелчки кнопками мыши, ввод и вывод курсора за пределы виджета, движения мышью), нажатиями клавиш на клавиатуре (нажатия клавиш, нажатия комбинация клавиш, отжатия клавиш), а также события, возникающие в результате изменения виджетов (ввод данных, изменение размеров окна, прокрутка).

4. Как обрабатываются события в Tkinter?

При вызове метода `bind` событие передается в качестве первого аргумента. Название события заключается в кавычки, а также в угловые скобки `<и>`. События описывается с помощью зарезервированных последовательностей ключевых слов.

5. Как обрабатываются события мыши в Tkinter?

В функции `move` извлекаются значения атрибутов `x` и `y` объекта `event`, в которых хранятся координаты местоположения курсора мыши в пределах виджета, по отношению к которому было сгенерировано событие. В данном случае виджетом является главное окно, а событием – `<Motion>`, т. е. перемещение мыши.

6. Каким образом можно отображать графические примитивы в Tkinter?

В `tkinter` от класса `Canvas` создаются объекты-холсты, на которых можно "рисовать", размещая различные фигуры и объекты. Делается это с помощью вызовов соответствующих методов.



При создании экземпляра Canvas необходимо указать его ширину и высоту. При размещении геометрических примитивов и других объектов указываются их координаты на холсте. Точкой отсчета является верхний левый угол.

7. Перечислите основные методы для отображения графических примитивов в Tkinter.

- `create_line()` – рисует линию;
- `create_rectangle()` – рисует прямоугольник;
- `create_oval()` – рисует овал;
- `create_polygon()` – рисует многоугольник;
- `create_text()` – добавляет текст.

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

Обратиться к ранее созданным фигурам на холсте можно с помощью идентификаторов и тегов.

9. Каково назначение тегов в Tkinter?

В отличие от идентификаторов, которые являются уникальными для каждого объекта, один и тот же тег может присваиваться разным объектам. Дальнейшее обращение к такому тегу позволит изменить все объекты, в которых он был указан.