# Лабораторная работа 4.7

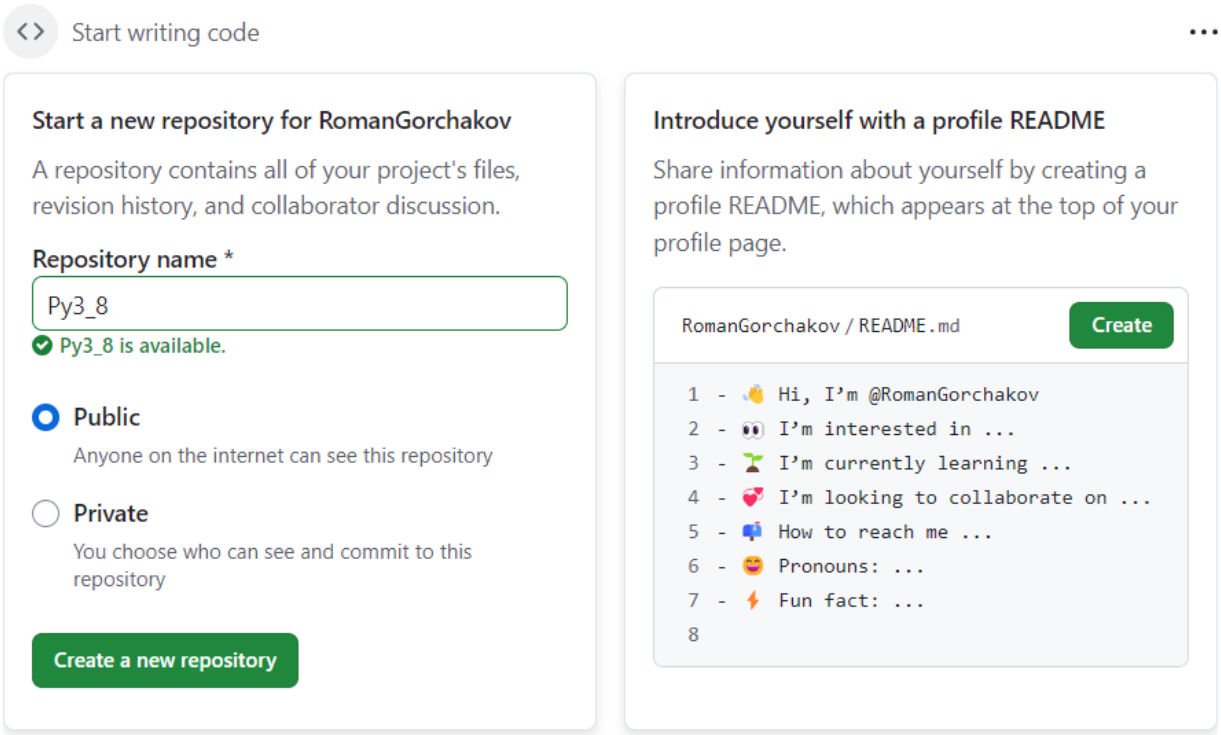
Тема: Основы работы с Tkinter

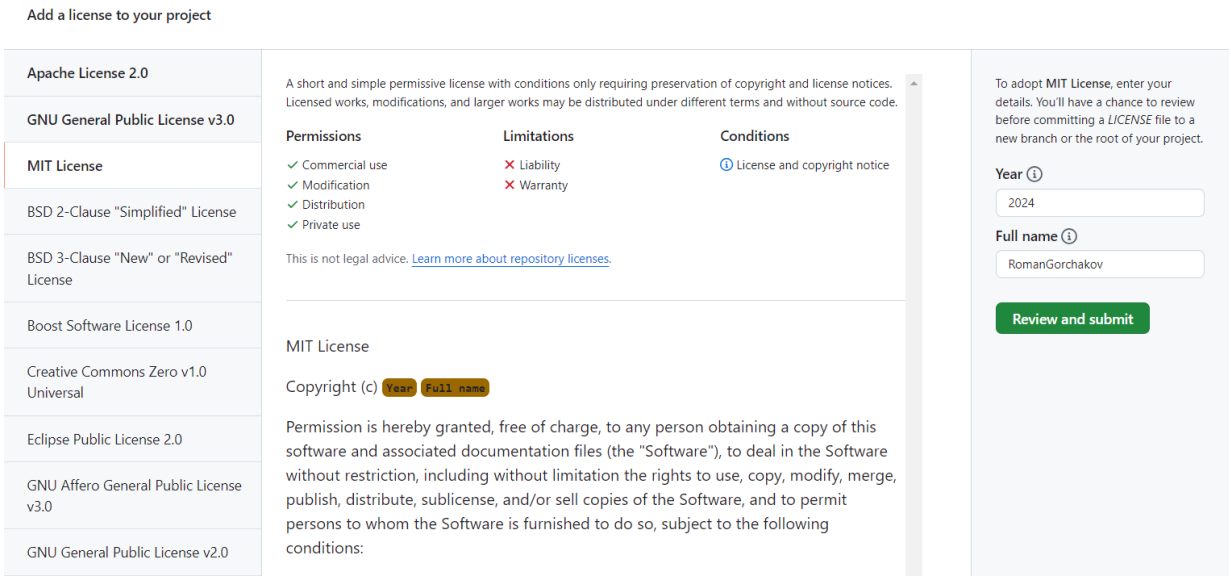
Цель работы: приобретение навыков построения графического интерфейса пользователя GUI с помощью пакета Tkinter языка программирования Python версии 3.x.

Ссылка на GitHub: https://github.com/RomanGorchakov/Py3_8

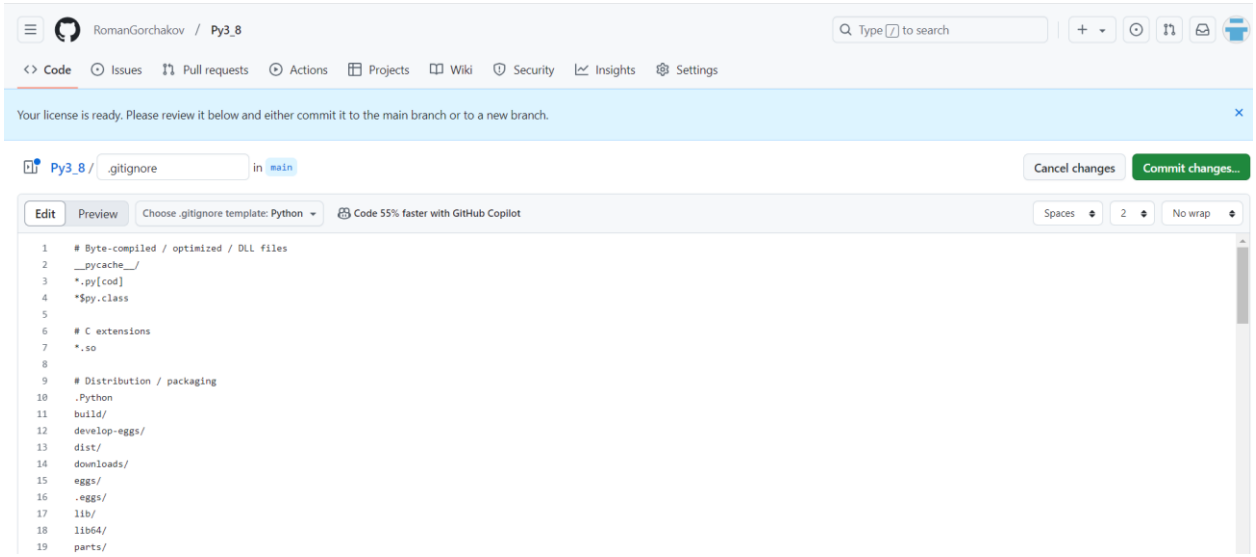
## Порядок выполнения работы

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

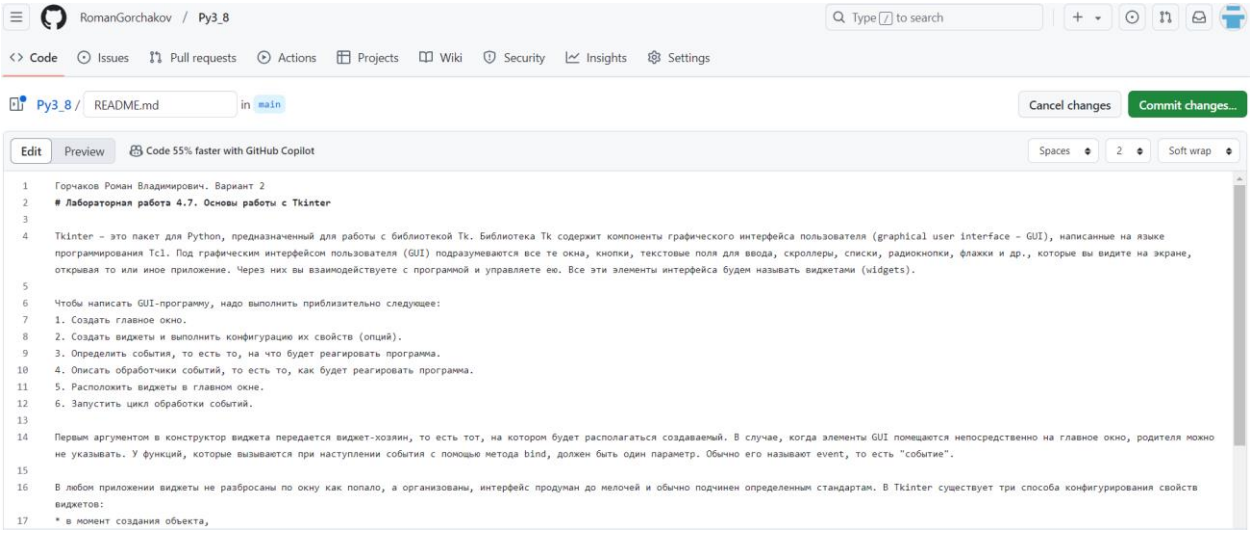





2. Теперь необходимо дополнить файл .gitignore с необходимыми правилами для языка программирования Python. Для этого переходим по ссылке «https://github.com/github/gitignore» и скачиваем оттуда файл «Python.gitignore».




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

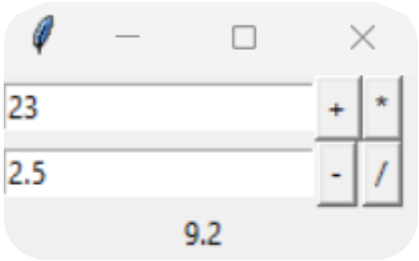



4. В окне «Codespace» выбираем опцию «Create codespace on main». Откроется терминал, куда мы введём команду «git clone», чтобы клонировать свой репозиторий. После этого организуем репозиторий в соответствие с моделью ветвления Git-flow. Для этого введём в терминал команды: «git checkout –b develop» для 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.

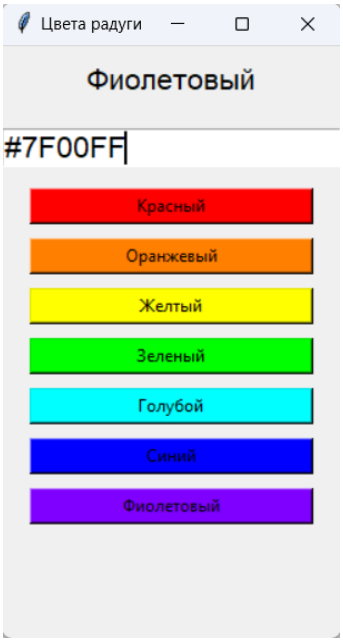
```
@RomanGorchakov →/workspaces/Py3_8 (main) $ git checkout -b develop
Switched to a new branch 'develop'
@RomanGorchakov →/workspaces/Py3_8 (develop) $ git branch feature_branch
@RomanGorchakov →/workspaces/Py3_8 (develop) $ git branch release/1.0.0
@RomanGorchakov →/workspaces/Py3_8 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
@RomanGorchakov →/workspaces/Py3_8 (main) $ git branch hotfix
@RomanGorchakov →/workspaces/Py3_8 (main) $ git checkout develop
Switched to branch 'develop'
@RomanGorchakov →/workspaces/Py3_8 (develop) $
```

```
@RomanGorchakov →/workspaces/Py3_8 (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 68.7 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_8 (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
@RomanGorchakov →/workspaces/Py3_8 (develop) $ pre-commit sample-config > .pre-commit-config.yaml
@RomanGorchakov →/workspaces/Py3_8 (develop) $ conda env export > environment.yml
```

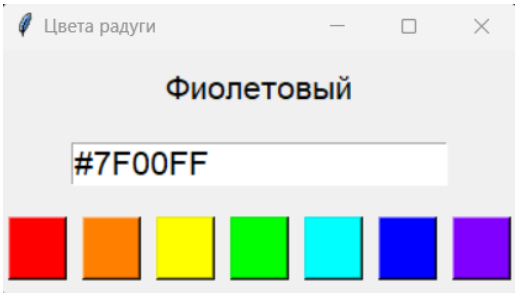
5. Создаём файл «task1.py», в котором нужно написать простейший калькулятор, состоящий из двух текстовых полей, куда пользователь вводит числа, и четырех кнопок «+», «-», «*», «/». Результат вычисления должен отображаться в метке. Если арифметическое действие выполнить невозможно (например, если были введены буквы, а не числа), то в метке должно появляться слово "ошибка".



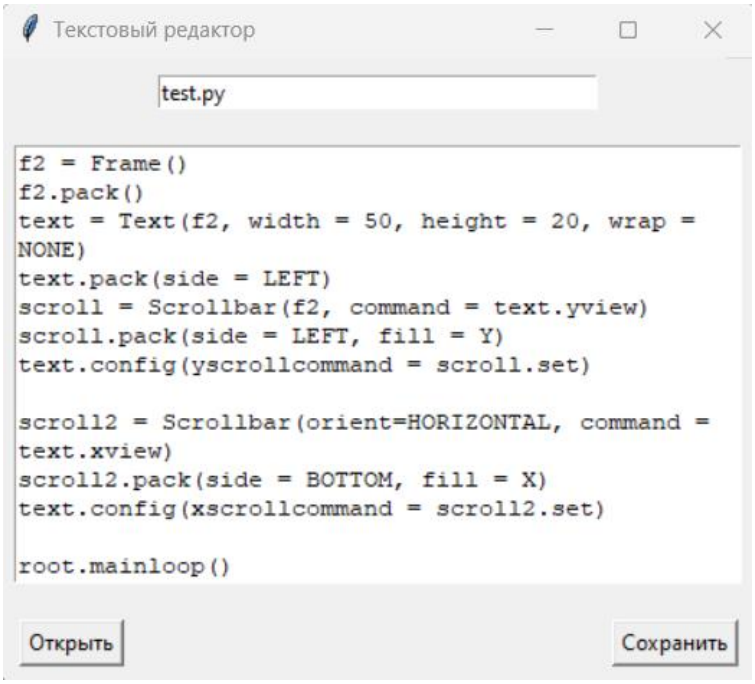

6. Создаём файл «task2.py», в котором нужно написать программу, состоящую из семи кнопок, цвета которых соответствуют цветам радуги. При нажатии на ту или иную кнопку в текстовое поле должен вставляться код цвета, а в метку – название цвета.



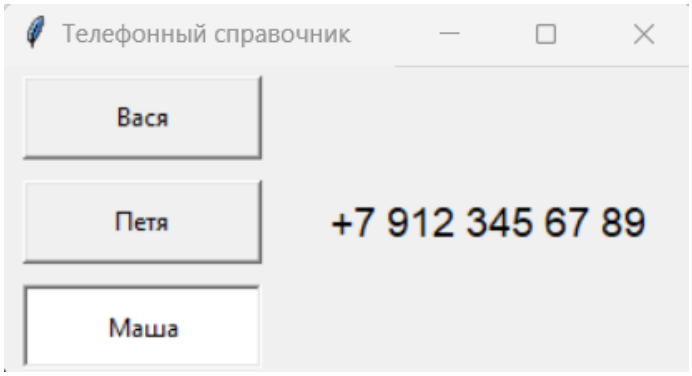

7. Создаём файл «task3.py», перепишем программу из файла «task2.py» так, чтобы интерфейс выглядел примерно так, как в образце.




8. Создаём файл «task4.py», в котором нужно написать программу, состоящую из однострочного и многострочного текстовых полей и двух кнопок "Открыть" и "Сохранить". При клике на первую должен открываться на чтение файл, чье имя указано в поле класса Entry, а содержимое файла должно загружаться в поле типа Text. При клике на вторую кнопку текст, введенный пользователем в экземпляр Text, должен сохраняться в файле под именем, которое пользователь указал в однострочном текстовом поле. Файлы будут читаться и записываться в том же каталоге, что и файл скрипта, если указывать имена файлов без адреса.




9. Создаём файл «task5.py», в котором нужно написать программу, в которой имеется несколько объединенных в группу радиокнопок, индикатор которых выключен (indicatoron=0). Если какая-нибудь кнопка включается, то в метке должна отображаться соответствующая ей информация. Обычных кнопок в окне быть не должно.




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

```
 .../\320\227\320\260\320\264\320\260\320\275\320\270\320\265 1/task1.py"                          |  52 +++++++++
 .../\320\227\320\260\320\264\320\260\320\275\320\270\320\265 2/task2.py"                          |  38 +++++++
 .../\320\227\320\260\320\264\320\260\320\275\320\270\320\265 3/task3.py"                          |  40 +++++++
 .../\320\227\320\260\320\264\320\260\320\275\320\270\320\265 4/task4.py"                          |  48 ++++++++
 .../\320\227\320\260\320\264\320\260\320\275\320\270\320\265 4/test.py"                           |  19 ++++
 .../\320\227\320\260\320\264\320\260\320\275\320\270\320\265 5/task5.py"                          |  48 ++++++++
 ...\320\2404.6_\320\223\320\276\321\200\321\207\320\260\320\272\320\276\320\262\320\240\320\222.pdf" | Bin 0 -> 1047156 bytes
 14 files changed, 845 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 4/test.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\260\320\272\320\276\320\262\320\240\320\222.pdf"
@RomanGorchakov →/workspaces/Py3_8 (main) $ git push -u
Enumerating objects: 24, done.
Counting objects: 100% (24/24), done.
Delta compression using up to 2 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (23/23), 982.85 KiB | 20.91 MiB/s, done.
Total 23 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/RomanGorchakov/Py3_8
   2e6c608..c6014d7  main -> main
branch 'main' set up to track 'origin/main'.
@RomanGorchakov →/workspaces/Py3_8 (main) $
```

Py3_8 Public

Pin | Unwatch 1 | Fork 0 | Star 0

main | 1 Branch | 0 Tags | Go to file t | Add file | Code

RomanGorchakov TKinter — c6014d7 · now — 4 Commits

| | | |
|---|---|---|
| Задания | TKinter | now |
| Отчёт | TKinter | now |
| .gitignore | Create .gitignore | last week |
| .pre-commit-config.yaml | TKinter | now |
| .python-version | TKinter | now |
| LICENSE | Create LICENSE | last week |
| README.md | Create README.md | 48 minutes ago |
| edu.pyoop.code-workspace | TKinter | now |
| environment.yml | TKinter | now |
| pyproject.toml | TKinter | now |
| setup.cfg | TKinter | now |
| uv.lock | TKinter | now |

README | MIT license

Горчаков Роман Владимирович. Вариант 2

# Лабораторная работа 4.7. Основы работы с Tkinter

**About**

*No description, website, or topics provided.*

Readme
MIT license
Activity
0 stars
1 watching
0 forks

**Releases**

No releases published
Create a new release

**Packages**

No packages published
Publish your first package

**Languages**

Python 100.0%

**Suggested workflows**

Based on your tech stack

Publish Python Package — Configure

Publish a Python Package to PyPI on release.

Контрольные вопросы

1. Какие существуют средства в стандартной библиотеке Python для построения графического интерфейса пользователя?

PyQt – это одна из наиболее популярных и мощных библиотек для разработки графического интерфейса на Python. Она предоставляет обёртку над Qt Framework, который является одним из самых мощных и широко используемых фреймворков для создания приложений с GUI.

Kivy – это открытая и свободная библиотека Python для создания мультимедийных приложений и графического интерфейса. Она разработана с учётом кросс-платформенности, что позволяет создавать приложения, работающие на различных операционных системах, включая Windows, macOS, Linux, Android и iOS.

PySide – это библиотека Python, предоставляющая возможности для создания графического интерфейса с использованием фреймворка Qt. Она предоставляет Python-обёртку над Qt, позволяя разработчикам создавать мощные и кросс-платформенные приложения с интерактивным пользовательским интерфейсом. 3

wxPython – это библиотека Python, которая предоставляет возможности для создания кросс-платформенных графических интерфейсов с использованием фреймворка wxWidgets. Она позволяет разработчикам создавать приложения с настраиваемыми виджетами, обработкой событий и взаимодействием с пользователем.

2. Что такое Tkinter?

Tkinter – пакет для Python, предназначенный для работы с библиотекой Tk. Библиотека Tk содержит компоненты графического интерфейса пользователя (graphical user interface – GUI), написанные на языке программирования Tcl.

3. Какие требуется выполнить шаги для построения графического интерфейса с помощью Tkinter?

Чтобы написать GUI-программу, надо выполнить приблизительно следующее:

1. Создать главное окно.

2. Создать виджеты и выполнить конфигурацию их свойств (опций).

3. Определить события, то есть то, на что будет реагировать программа.

4. Описать обработчики событий, то есть то, как будет реагировать программа.

5. Расположить виджеты в главном окне. 6. Запустить цикл обработки событий.

4. Что такое цикл обработки событий?

Цикл обработки событий – это программная конструкция, которая ожидает и отправляет события или сообщения в программе.

5. Каково назначение экземпляра класса Tk при построении графического интерфейса с помощью Tkinter?

Назначение экземпляра класса Tk при построении графического интерфейса с помощью Tkinter – это создание базового окна приложения.

6. Для чего предназначены виджеты Button, Label, Entry и Text?

Button – кнопка.

Label – метка.

Entry – однострочное текстовое поле.

Text – многострочное текстовое поле.

7. Каково назначение метода pack() при построении графического интерфейса пользователя?

Упаковщик (packer) вызывается методом pack, который имеется у всех виджетов-объектов. Если к элементу интерфейса не применить какой-либо из менеджеров геометрии, то он не отобразится в окне. При этом в одном окне (или любом другом родительском виджете) нельзя комбинировать разные менеджеры.

8. Как осуществляется управление размещением виджетов с помощью метода pack()?

У метода pack есть параметр side (сторона), который принимает одно из четырех значенийконстант tkinter – TOP, BOTTOM, LEFT, RIGHT (верх, низ, лево,

право). По умолчанию, когда в pack не указывается side, его значение равняется TOP. Из-за этого виджеты располагаются вертикально.

9. Как осуществляется управление полосами прокрутки в виджете Text?

В tkinter скроллеры производятся от класса Scrollbar . Объект-скроллер связывают с виджетом, которому он требуется. Это не обязательно многострочное текстовое поле.

10. Для чего нужны тэги при работе с виджетом Text?

Особенностью текстового поля библиотеки Tk является возможность форматировать текст в нем, то есть придавать его разным частям разное оформление. Делается это с помощью методов tag_add и tag_config. Первый добавляет тег, при этом надо указать его произвольное имя и отрезок текста, к которому он будет применяться. Метод tag_config настраивает тегу стили оформления.

11. Как осуществляется вставка виджетов в текстовое поле?

В Text можно вставлять другие виджеты помощью метода window_create. Потребность в этом не велика, однако может быть интересна с объектами типа Canvas.

12. Для чего предназначены виджеты Radiobutton и Checkbutton?

В Tkinter от класса Radiobutton создаются радиокнопки, от класса Checkbutton – флажки.

13. Что такое переменные Tkinter и для чего они нужны?

Переменные Tkinter нужны, чтобы снимать сведения о состоянии флажков.

14. Как осуществляется связь переменных Tkinter с виджетами Radiobutton и Checkbutton?

По значению связанной с Checkbutton переменной можно определить, установлен ли флажок или снят, что в свою очередь повлияет на ход выполнения программы. У каждого флажка должна быть своя переменная Tkinter. Иначе при включении одного флажка, другой будет выключаться, так как значение общей tkinter-переменной изменится и не будет равно значению опции onvalue первого флажка.