Tkinter

Основная информация

Tkinter (от англ. Tk interface) — кросс-платформенная событийно-ориентированная графическая библиотека на основе средств Tcl/Tk (широко распространённая в мире GNU/Linux и других UNIX-подобных систем, портирована также и на Microsoft Windows), написанная Стином Лумхольтом и Гвидо ван Россумом. **Входит в стандартную библиотеку Python.**

Tcl — это динамически интерпретируемый язык программирования, как и Python. Хотя его можно использовать самостоятельно как язык программирования общего назначения, чаще всего он встраивается в приложения на языке C как скриптовый движок или интерфейс к инструментарию Tk.

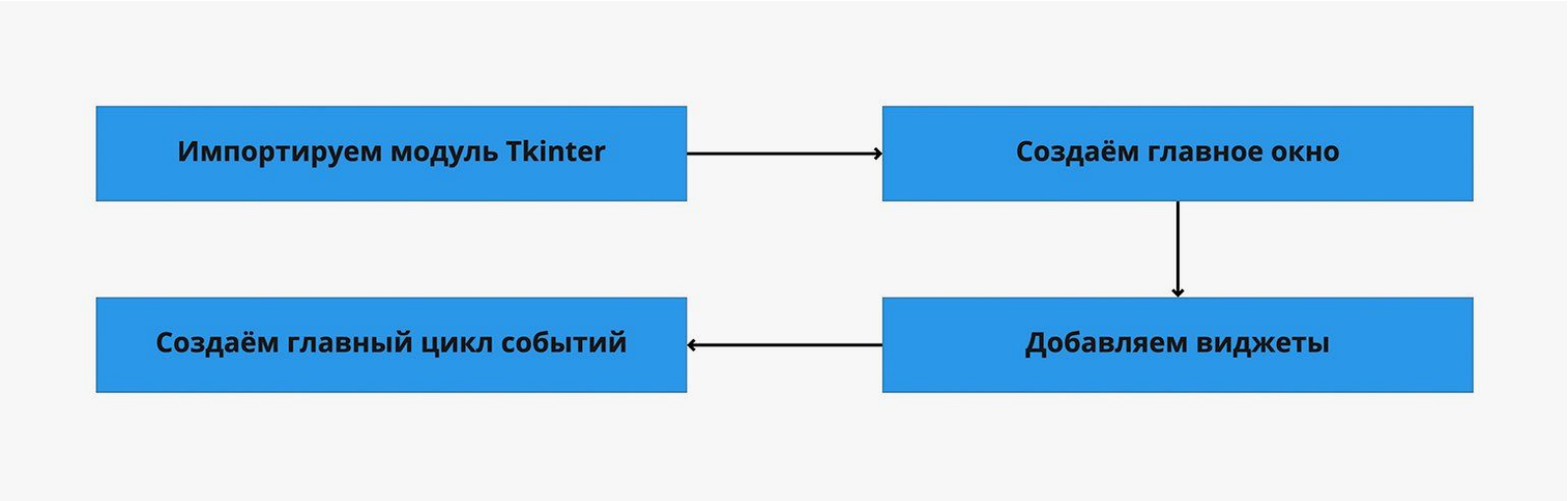
Tk — кроссплатформенная библиотека базовых элементов **графического интерфейса**. Tk представляет разработчику набор Tcl-команд, предназначенных для создания компонентов и выполнения различных действий с ними.

Ttk — это новое семейство виджетов Tk, которые обеспечивают гораздо лучший внешний вид на разных платформах, чем многие классические виджеты Tk.

https://ru.wikipedia.org/wiki/Tk_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)

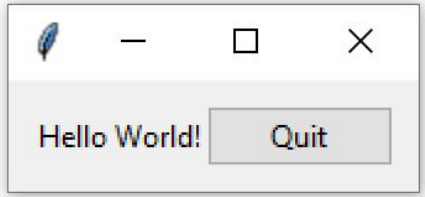
Основные концепции

Схематично работу с Tkinter можно представить в виде четырёх шагов:



Основные концепции

```
from tkinter import *
from tkinter import ttk
root = Tk()
frm = ttk.Frame(root, padding=10)
frm.grid()
ttk.Label(frm, text="Hello World!").grid(column=0, row=0)
ttk.Button(frm, text="Quit", command=root.destroy).grid(column=1, row=0)
root.mainloop()
```



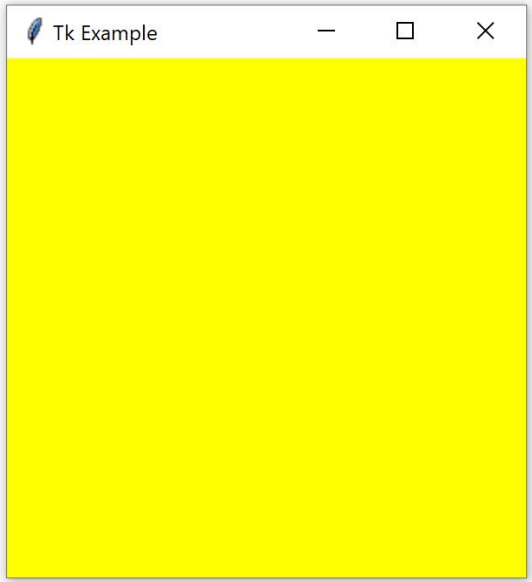
виджеты	Пользовательский интерфейс Tkinter состоит из отдельных *виджетов* . Каждый виджет представлен как объект Python, созданный из таких классов, как `ttk.Frame`, `ttk.Label`, и `ttk.Button`.
иерархия виджетов	Виджеты организованы в *иерархию* . Метка и кнопка содержались в фрейме, который в свою очередь содержался в корневом окне. При создании каждого *дочернего* виджета его *родительский* виджет передается в качестве первого аргумента конструктору виджета.
параметры конфигурации	Виджеты имеют *параметры конфигурации* , которые изменяют их внешний вид и поведение, например, текст, отображаемый на метке или кнопке. Различные классы виджетов будут иметь различные наборы параметров.
управление геометрией (geometry management)	Виджеты не добавляются автоматически в пользовательский интерфейс при их создании. *Менеджер геометрии,* например, `grid` контролирует, где в пользовательском интерфейсе они размещаются.
цикл событий	Tkinter реагирует на изменения в программе только при активном выполнении *цикла событий* . Если ваша программа не выполняет цикл событий, ваш пользовательский интерфейс не обновится.

Как Tkinter взаимодействует с Tcl/Tk

```
ttk::frame .frm -padding 10
grid .frm
grid [ttk::label .frm.lbl -text "Hello World!"] -column 0 -row 0
grid [ttk::button .frm.btn -text "Quit" -command "destroy ."] -column 1 -row 0
```

Основное окно

```
from tkinter import *

root = Tk()  # create a root widget
root.title("Tk Example")
root.configure(background="yellow")
# root['bg'] = "yellow"
root.minsize(200, 200)  # width, height
root.maxsize(500, 500)
root.geometry("300x300+50+50")  # width x height + x + y
root.mainloop()
```



Виджеты

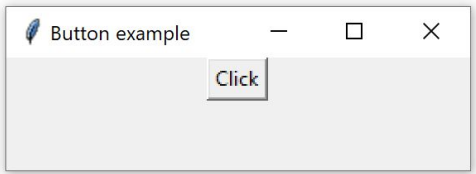
Ключевые объекты в работе с Tkinter — виджеты. Это аналоги тегов из HTML, которые позволяют создавать интерактивные и неинтерактивные элементы, например надписи или кнопки. Всего их 18, но чаще всего используют следующие:

- Button — кнопки;
- Canvas — «холст», на котором рисуют графические фигуры;
- Entry — виджет для создания полей ввода;
- Label — контейнер для размещения текста или изображения;
- Menu — виджет для создания пунктов меню

Виджеты

Одним из наиболее используемых компонентов в графических программах является кнопка. В tkinter кнопки представлены классом **Button**. Основные параметры виджета Button:

- **command**: функция, которая вызывается при нажатии на кнопку
- **compund**: устанавливает расположение картинки и текста относительно друг друга
- **cursor**: курсор указателя мыши при наведении на метку
- **image**: ссылка на изображение, которое отображается на метке
- **pading**: отступы от границ виджета до его текста
- **state**: состояние кнопки
- **text**: устанавливает текст метки
- **textvariable**: устанавливает привязку к элементу StringVar
- **underline**: указывает на номер символа в тексте кнопки, который подчеркивается. По умолчанию значение -1, то есть никакой символ не подчеркивается
- **width**: ширина виджета



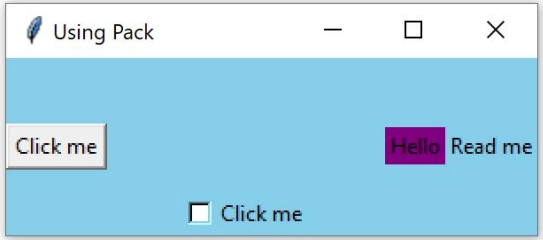
Позиционирование виджетов

В Tkinter есть три типа менеджеров компоновки — **pack**, **place**, и **grid**.

Параметры метода **pack()**:

- **expand**: если равно True, то виджет заполняет все пространство контейнера.
- **fill**: определяет, будет ли виджет растягиваться, чтобы заполнить свободное пространство вокруг. Этот параметр может принимать следующие значения: NONE (по умолчанию, элемент не растягивается), X (элемент растягивается только по горизонтали), Y (элемент растягивается только по вертикали) и BOTH (элемент растягивается по вертикали и горизонтали).
- **anchor**: помещает виджет в определенной части контейнера. Может принимать значения n, e, s, w, ne, nw, se, sw, c, которые являются сокращениями от Noth(север - вверх), South (юг - низ), East (восток - правая сторона), West (запад - левая сторона) и Center (по центру). Например, значение nw указывает на верхний левый угол
- **side**: выравнивает виджет по одной из сторон контейнера. Может принимать значения: TOP (по умолчанию, выравнивается по верхней стороне контейнера), BOTTOM (выравнивание по нижней стороне), LEFT (выравнивание по левой стороне), RIGHT (выравнивание по правой стороне).
- **ipadx**: устанавливает отступ содержимого виджета от его границы по горизонтали.
- **ipady**: устанавливают отступ содержимого виджета от его границы по вертикали.
- **padx**: устанавливает отступ виджета от границ контейнера по горизонтали.
- **pady**: устанавливает отступ виджета от границ контейнера по вертикали.

```
from tkinter import *

root = Tk()
root.title("Using Pack")
root.geometry("300x100")  # set starting size of window
root.config(bg="skyblue")

# Example of how to arrange Button widget using pack
button1 = Button(root, text="Click me")
button1.pack(side="left")

# Example of how to arrange Label widgets using pack
label1 = Label(root, text="Read me", bg="skyblue")
label1.pack(side="right")
label2 = Label(root, text="Hello", bg="purple")
label2.pack(side="right")

def toggled():
    '''display a message to the terminal every time the check button
    is clicked'''
    print("The check button works.")

# Example of how to arrange Checkbutton widget using pack
var = IntVar()  # Variable to check if checkbox is clicked, or not
check = Checkbutton(root, text="Click me", bg="skyblue", command=toggled, variable=var)
check.pack(side="bottom")
root.mainloop()
```



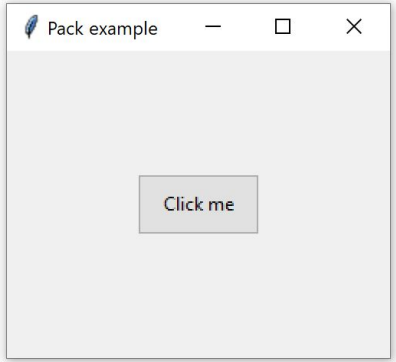
Позиционирование виджетов

Метод **place()** позволяет более точно настроить координаты и размеры виджета. Он принимает следующие параметры:

- **height** и **width**: устанавливают соответственно высоту и ширину элемента в пикселях
- **relheight** и **relwidth**: также задают соответственно высоту и ширину элемента, но в качестве значения используется число float в промежутке между 0.0 и 1.0, которое указывает на долю от высоты и ширины родительского контейнера
- **x** и **y**: устанавливают смещение элемента по горизонтали и вертикали в пикселях соответственно относительно верхнего левого угла контейнера
- **relx** и **rely**: также задают смещение элемента по горизонтали и вертикали, но в качестве значения используется число float в промежутке между 0.0 и 1.0, которое указывает на долю от высоты и ширины родительского контейнера
- **bordermode**: задает формат границы элемента. Может принимать значение `INSIDE` (по умолчанию) и `OUTSIDE`
- **anchor**: устанавливает опции растяжения элемента. Может принимать значения n, e, s, w, ne, nw, se, sw, c, которые являются сокращениями от North(север - вверх), South (юг - низ), East (восток - правая сторона), West (запад - левая сторона) и Center (по центру). Например, значение nw указывает на верхний левый угол

Позиционирование виджетов

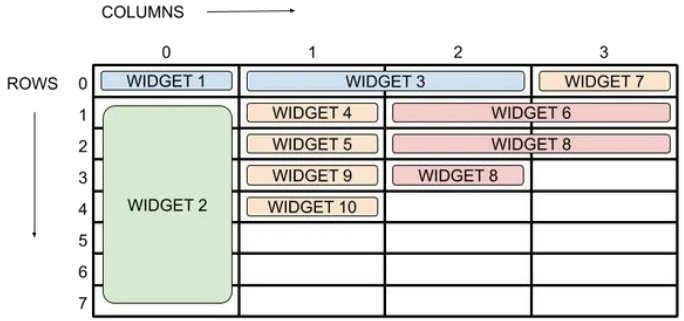
```
from tkinter import *
from tkinter import ttk

root = Tk()
root.title("METANIT.COM")
root.geometry("250x200")

btn = ttk.Button(text="Click me")
btn.place(relx=0.5, rely=0.5, anchor="c", width=80, height=40)

root.mainloop()
```



Позиционирование виджетов

Метод **grid** применяет следующие параметры:

- **column**: номер столбца, отсчет начинается с нуля
- **row**: номер строки, отсчет начинается с нуля
- **columnspan**: сколько столбцов должен занимать элемент
- **rowspan**: сколько строк должен занимать элемент
- **ipadx** и **ipady**: отступы по горизонтали и вертикали соответственно от границ элемента до его содержимого
- **padx** и **pady**: отступы по горизонтали и вертикали соответственно от границ ячейки грида до границ элемента
- **sticky**: выравнивание элемента в ячейке, если ячейка больше элемента. Может принимать значения n, e, s, w, ne, nw, se, sw, которые указывают соответствующее направление выравнивания



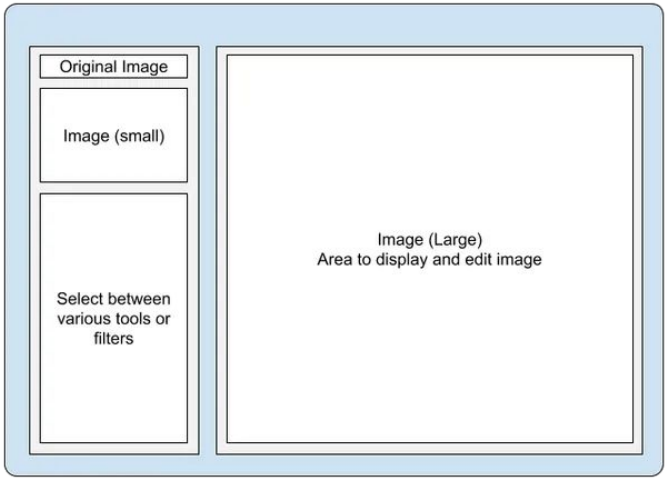
Позиционирование виджетов

```
from tkinter import *
from tkinter import ttk

root = Tk()
root.title("Grid example")
root.geometry("250x200")

for c in range(3): root.columnconfigure(index=c, weight=1)
for r in range(3): root.rowconfigure(index=r, weight=1)

for r in range(3):
    for c in range(3):
        btn = ttk.Button(text=f"({r},{c})")
        btn.grid(row=r, column=c, ipadx=6, ipady=6, padx=4, pady=4, sticky=NSEW)

root.mainloop()
```

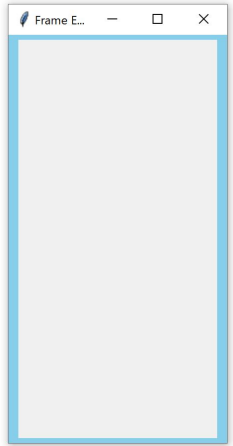
Frame

Компоновка GUI становится проще с помощью **Frame** виджета.

Фреймы — это как коробки или мини-окна в родительском окне — самодостаточные области, каждая из которых может содержать свои сетки, поэтому добавлять к ним виджеты очень просто. При необходимости можно даже вкладывать фреймы друг в друга.




```
from tkinter import *

root = Tk()  # create root window
root.title("Frame example")
root.config(bg="skyblue")

# Create Frame widget
left_frame = Frame(root, width=200, height=400)
left_frame.grid(row=0, column=0, padx=10, pady=5)
root.mainloop()
```



Обработка событий

Для обработки распространенных и наиболее используемых событий Tkinter предоставляет интерфейс команд.

```
def click():
    print("Hello")

btn = ttk.Button(text="Click", command=click)
```

Для обработки других событий (например, для кнопки обработать получение фокуса) Tkinter предоставляет ряд встроенных событий.

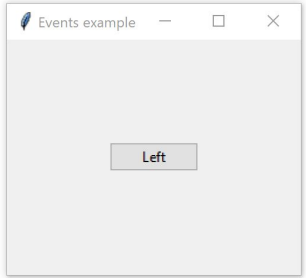
Пример некоторых из них:

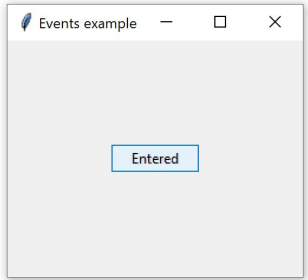
- **Activate**: окно становится активным.
- **Deactivate**: окно становится неактивным.
- **MouseWheel**: прокрутка колеса мыши.
- **KeyPress**: нажатие клавиши на клавиатуре.
- **KeyRelease**: освобождение нажатой клавиши
- **ButtonPress**: нажатие кнопки мыши.
- **ButtonRelease**: освобождение кнопки мыши.
- **Motion**: движение мыши.
- **Configure**: изменение размера и положения виджета
- **Destroy**: удаление виджета
- **FocusIn**: получение фокуса
- **FocusOut**: потеря фокуса.
- **Enter**: указатель мыши вошел в пределы виджета.
- **Leave**: указатель мыши покинул виджет.

Обработка событий

Для привязки события к виджету применяется метод **bind()**: `bind(событие, функция`

```
from tkinter import *
from tkinter import ttk


root = Tk()
root.title("Events example")
root.geometry("250x200")

def entered(event):
    btn["text"] ="Entered"

def left(event):
    btn["text"] ="Left"

btn = ttk.Button(text="Click")
btn.pack(anchor=CENTER, expand=1)

btn.bind("<Enter>", entered)
btn.bind("<Leave>", left)

root.mainloop()
```





Сравнение Tkinter и PyQt6

	PyQt6	Tkinter
Лицензия	PyQt доступен по лицензии Riverbank Commercial License и GPL (но есть PySide6).	Tkinter имеет открытый исходный код и бесплатен для коммерческого использования.
Установка и настройка	Для использования средств Qt требуется установка соответствующей библиотеки Python.	Tkinter входит в стандартную библиотеку Python, поэтому для разработки установка дополнительных пакетов не требуется.
Документация и поддержка сообщества	PyQt также имеет исчерпывающую документацию, и его сообщество растет. Поскольку PyQt основан на платформе Qt framework, вы можете найти обширные ресурсы, связанные с разработкой Qt, которые могут быть полезны при работе с PyQt.	У Tkinter обширная документация и большое сообщество. Доступно множество онлайн-ресурсов, руководств и примеров, которые помогут новичкам начать работу.
Легкость понимания	Требуется много времени, чтобы разобраться во всех тонкостях PyQt.	Tkinter прост в понимании и освоении благодаря небольшой библиотеке.

Сравнение Tkinter и PyQt6

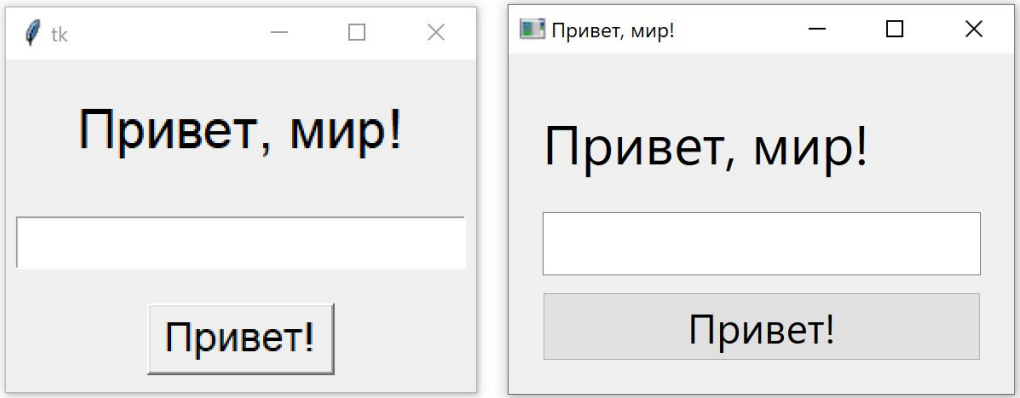
	PyQt6	Tkinter
Функции и возможности	PyQt предлагает широкий спектр дополнительных функций, таких как пользовательские виджеты, поддержка многопоточности и обширное управление макетом. Она подходит для сложных приложений и профессиональных проектов, где требуется расширенная функциональность.	Tkinter предоставляет базовые виджеты (кнопки, метки, текстовые поля и т.д.) И простое управление макетом. Она подходит для небольших проектов и простых приложений, но может быть ограничена для более сложных проектов.
Производительность	PyQt обладает лучшей производительностью по сравнению с Tkinter, что делает его подходящим для более крупных проектов и приложений, требующих быстрого обновления пользовательского интерфейса.	Tkinter обладает достойной производительностью для приложений малого и среднего размера. Однако с большими проектами или приложениями, требующими быстрого обновления пользовательского интерфейса, могут возникнуть проблемы.
Виджеты	Поставляется с множеством мощных и продвинутых виджетов.	Виджетов меньше, чем в PyQt.
Конструктор пользовательского интерфейса	В PyQt есть инструмент Qt Designer, который мы можем использовать для создания графических интерфейсов, а затем получать код Python этого графического интерфейса.	Аналогичного инструмента у Tkinter нет.

Сравнение Tkinter и PyQt6

Внешний вид:

Внешний вид Tkinter по умолчанию устарел и не всегда хорошо сочетается с современными операционными системами. Тем не менее, можно настроить внешний вид с помощью тем и стилей. (Также есть ttk и Customtkinter).

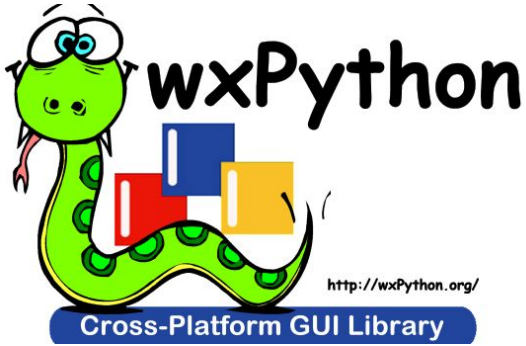
Приложения PyQt имеют собственный внешний вид на всех поддерживаемых платформах, что делает их визуально привлекательными и совместимыми с пользовательской ОС. PyQt также предоставляет расширенные параметры стиля, позволяющие осуществлять обширную настройку.



Другие средства для работы с GUI на Python

WxPython — кросплатформенная обёртка библиотек графического интерфейса пользователя для Python, основанная на wxWidgets, написанная на языке программирования C++. Одна из альтернатив Tkinter, которая поставляется вместе с Python.

wxWidgets (ранее известная как wxWindows) — это кросс-платформенная библиотека инструментов с открытым исходным кодом для разработки кроссплатформенных на уровне исходного кода приложений, в частности для построения графического интерфейса пользователя (GUI).



https://www.wxpython.org/

Другие средства для работы с GUI на Python

Kivy — это библиотека для разработки кроссплатформенных приложений на Python. Она предоставляет мощные инструменты для создания графических интерфейсов и взаимодействий.

Kivy позволяет создавать мобильные приложения на языке Python.

К недостаткам Kivy можно отнести ненативный специфический пользовательский интерфейс.

Полезные ссылки

Документация: https://docs.python.org/3/library/tk.html

Руководства по Tkinter:

https://www.pythonguis.com/tkinter/

https://metanit.com/python/tkinter/