ПРИЕМЕНИЕ TKINTER ГРАФИЧЕСКОГО ИНТЕРФЕЙСА *КОМПОНОВЩИКА PYGUBU*

Абишева Айгуль Амантаевна
Магистр, старший преподаватель
Казахский Университет Экономики,
финансов и Международной торговли
г. Нур-Султан. Казахстан

Tkinter кросс-платформенная графическая библиотека на основе средств Tk, написанная Стином Лумхольтом и Гвидо ван Россумом. Входит в стандартную библиотеку Python.

Pygubu является инструментом RAD для того, чтобы *быстро* и *легко развитие пользовательских интерфейсов* для Pythonа tkinter модуля.

Разработанные пользовательские интерфейсы сохраняются в виде файлов XML, и с помощью *компоновщика pygubu* они могут загружаться приложениями динамически по мере необходимости.

Tk / Tcl давно является неотъемлемой частью Python. Это обеспечивает надежный платформы, оконный инструментарий, который доступен для программистов Python с использованием tkinter пакета, и его расширение tkinter.tix и tkinter.ttk модули [4].

tkinter пакет представляет собой тонкий объектно-ориентированный слой поверх Tcl / Tk. Для использования tkinter вам не нужно писать код Tcl, но вам нужно будет обратиться к документации Tk, а иногда и к документации Tcl. tkinter представляет собой набор оболочек, которые реализуют виджеты Tk как классы Python. Кроме того, внутренний модуль tkinter обеспечивает потокобезопасный механизм, который позволяет Python и Tcl взаимодействовать[1].

Его главные достоинства в том, что он быстрый и обычно поставляется в комплекте с Python.

tkinter пакет («Tk интерфейс») является стандартным интерфейсом Python для инструментария Tk GUI. Tk и tkinter доступны на большинстве платформ Unix, а также в системах Windows. (Сам Tk не является частью Python; он поддерживается на ActiveState.) Вы можете проверить, tkinterправильно ли установлен в вашей системе, запустив из командной строки; это должно открыть окно, демонстрирующее простой интерфейс Tk.python -m tkinter[3]

Вы можете установить pygubu, используя:

```
pip install pygubu
Чтобы запустить дизайнер введите в терминале следующею команду:
C:\Python34\Scripts\pygubu-designer.exe
```

где C: \ Python34 - это путь к вашему каталогу установки Python, после чего появится приложение pygubu-designer как указано на рисунке 1.

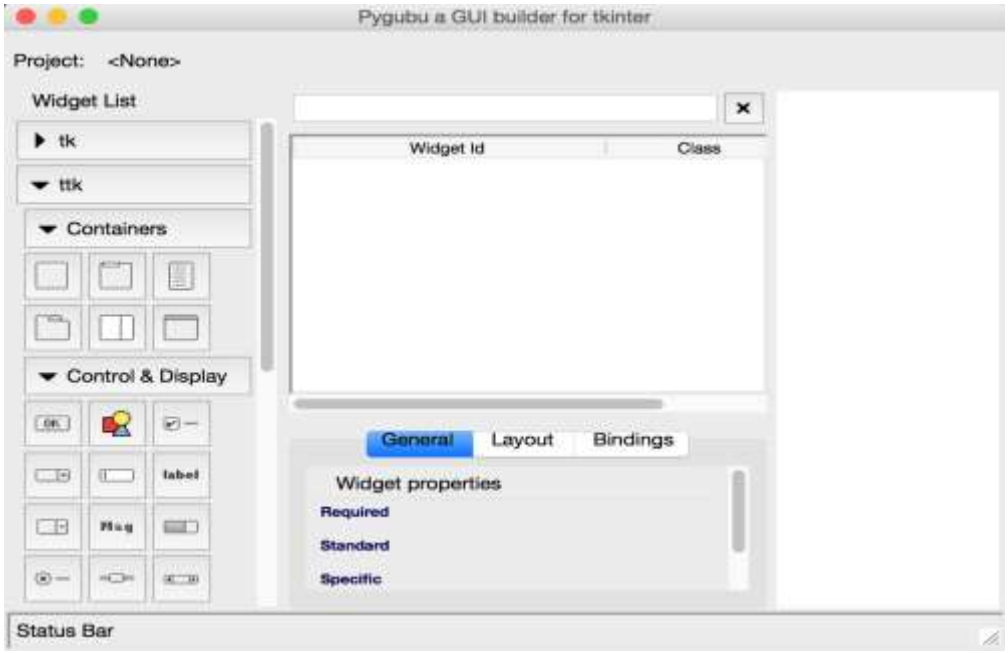

Рисунок 1. Основной интерфейс pygubu.

Модули Tkinter

В большинстве случаев tkinter это все, что вам действительно нужно, но также доступен ряд дополнительных модулей. Интерфейс Tk находится в двоичном модуле с именем tkinter[4].

В дополнение к модулю интерфейса Tk, tkinter включает в себя ряд модулей Python, tkinter.constants являясь одним из наиболее важных. Импорт tkinter автоматически импортирует tkinter.constants, поэтому, как правило, для использования Tkinter достаточно простого оператора import:

import tkinter

или чаще:

from tkinter import *

class tkinter.**Tk**(screenName = Нет, baseName = Нет, className = 'Tk', useTk = 1)

Tk класс инстанциируется без аргументов. Это создает виджет верхнего уровня Tk, который обычно является главным окном приложения. Каждый экземпляр имеет свой собственный интерпретатор Tcl.

tkinter.Tcl(screenName = Нет, baseName = Нет, className = 'Tk', useTk = 0)

Tcl() Функцией является заводом, который создает объект много подобного созданного Tk класса, за исключением того, что он не инициализировать подсистему Tk. Это чаще всего полезно при управлении интерпретатором Tcl в среде, где никто не хочет создавать посторонние окна верхнего уровня или где это невозможно. Объект, созданный Tcl() объектом, может создать окно Toplevel (и инициализировать подсистему Tk), вызвав его loadtk()метод.

Таблица 1.

Модули Tk

Наименование модуля	Назначение модулей
tkinter.scrolledtext	текстовый виджет со встроенной вертикальной полосой прокрутки.
tkinter.colorchooser	диалог, чтобы позволить пользователю выбрать цвет
tkinter.commondialog	базовый класс для диалогов, определенных в других модулях, перечисленных здесь.
tkinter.filedialog	общие диалоговые окна, позволяющие пользователю указать файл для открытия или сохранения.
tkinter.font	утилиты, помогающие работать со шрифтами.
tkinter.messagebox	доступ к стандартным диалоговым окнам Tk.
tkinter.simpledialog	основные диалоги и удобные функции.

Упаковщик является одним из механизмов управления геометрией Tk. Менеджеры геометрии используются для указания относительного позиционирования расположения виджетов внутри их контейнера - их общего *хозяина*. Упаковщик имеет качественную спецификацию отношения - *выше, слева, наполнение* и т.д. - и работает все, чтобы определить точные координаты размещения для тебя. Размер любого главного виджета определяется размером «подчиненных виджетов» внутри. Упаковщик используется для управления тем, где подчиненные виджеты появляются внутри главного устройства, в которое они упакованы. Вы можете упаковать виджеты в фреймы, а фреймы - в другие фреймы, чтобы достичь желаемого макета. Кроме того, расположение динамически настраивается, чтобы приспособить постепенные изменения к конфигурации, как только это упаковано[1].

Виджеты не появляются, пока их геометрия не будет указана с помощью менеджера геометрии. Распространенной ошибкой является раннее упущение спецификации геометрии, а затем удивление, когда виджет создается, но ничего не появляется. Виджет появится только после того, как к нему применен, например, метод упаковщика pack().

Метод pack () можно вызывать с помощью пар «ключевое слово-опция/ значение», которые управляют тем, где виджет должен отображаться в своем контейнере, и как он должен вести себя при изменении размера главного окна приложения. На рисунке 2 представлены основные разделы интерфейса.

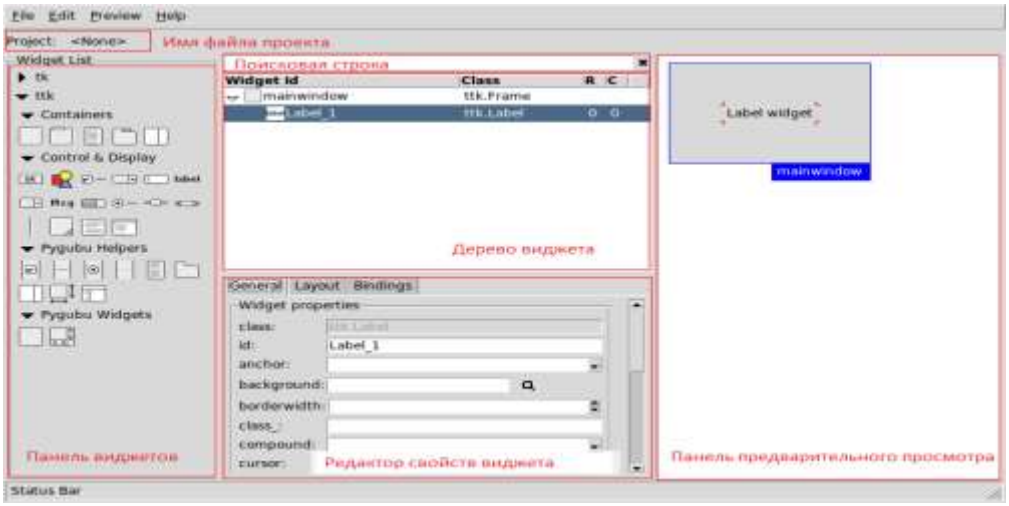

Рисунок 2. Основные разделы интерфейса

Основной интерфейс состоит из следующих разделов:

Панель виджета - в этом разделе показаны доступные виджеты. Он разделен на два основных раздела: виджеты tk и виджеты ttk.

Containers - это единственные виджеты, которые можно вставлять в качестве корневых элементов дерева виджета. Вам нужно добавить одно из них, чтобы продолжать добавлять виджеты в виде дочерних элементов виджета контейнера.

Control & Display - это стандартные виджеты управления, включенные в tk.

Pygubu helpers - эти виджеты используются для облегчения создания других виджетов, таких как Menuitems, вкладки Notebook, Panedwindow Panes, Treeview Columns и т. д.

Pygubu widgets - это пользовательские виджеты, используемые для создания главного окна pygubu и доступны для использования.

Дерево виджета - показывает виджеты, включенные в проект, в качестве иерархии деревьев.

Добавление виджета - чтобы добавить виджет в дерево, просто выберите виджет, который будет родителем нового виджета, и щелкните значок нового виджета на панели виджета.

Для добавления нового контейнер toplevel, просто убедитесь, что в дереве не выбран ни один элемент, и щелкните значок контейнера.

Теперь вы можете начать создание своего приложения tkinter с помощью виджетов, которые вы найдете в левой панели, называемой Widget List. После того как вы закончили создание своего определения пользовательского интерфейса, сохраните его в файле .ui, перейдя в верхнее меню File > Save.

Затем вы должны создать свой сценарий приложения, как показано ниже:

```
import tkinter as tk
import pygubu #1: Создание строителя
builder = builder = pygubu.Builder() #2: Загруза ui файла
builder.add_from_file('helloworld.ui') #3: Создайте виджет, используя мастер как родительский
mainwindow = builder.get_object('mainwindow')
root = tk.Tk()
app = Application(root)
root.mainloop()
```

Обратите внимание, что вместо helloworld.ui в следующей строке:

builder.add_from_file('helloworld.ui') Вы должны вставить имя файла (или путь) только что сохраненного пользовательского интерфейса.

```
import tkinter as tk
import pygubu
root = tk.Tk()
builder = pygubu.Builder()
builder.add_from_file('summa.ui')
mainwindow = builder.get_object('Frame_1')
result = builder.get_object('result')
a = builder.get_object('number_1')
b = builder.get_object('number_2')
def sum():
 c = int(a.get()) + int(b.get())
 result['text'] = str(c)
callbacks = {
 'sum': sum,
}
builder.connect_callbacks(callbacks)
root.mainloop()
result = builder.get_object('result')
```

С помощью данной команды, мы получаем виджет, который называется result, так же поступаем и остальными виджетами.

Затем мы создаем функцию **sum()**. В это функций мы суммируем введенные числа a и b и выводим результат.

```
callbacks = {
 'sum': sum,
}
builder.connect_callbacks(callbacks)
```

Здесь мы присвоим имена для функции, которые будут доступны в дизайнере. Так как мы назвали функцию sum так же, то в дизайнере для кнопки в поле command нужно написать sum. Запустив программу, увидим следующий результат.

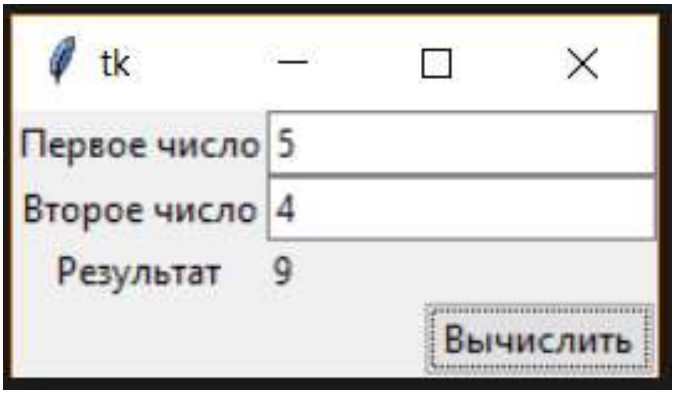

Рисунок 3. Результат программы.

Список литературы

1. Васильев А **Издательство:** Наука и техника. 2016г. 432с

2. Златопольский Д М. ДМК Пресс.: Москва. 2017г.-285с

3. Любанович Билл. Простой Python. Современный стиль рограммирования. - СПб.: Питер, 2016. - 480 с.

4. Федоров Д. Ю. Основы программирования на примере языка Python: учеб.пособие / Д. Ю. Федоров. – СПб., 2016. – 176 с.