Подробное введение в Python часть 5

СОЗДАНИЕ GUI НА PYTHON

1. Что такое Tkinter

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

Под графическим интерфейсом пользователя (GUI) подразумеваются все те окна, кнопки, текстовые поля для ввода, скроллеры, списки, радиокнопки, флажки и др., которые вы видите на экране, открывая то или иное приложение. Через них вы взаимодействуете с программой и управляете ею. Все эти элементы интерфейса будем называть виджетами (widgets).

В настоящее время почти все приложения, которые создаются для конечного пользователя, имеют GUI. Редкие программы, подразумевающие взаимодействие с человеком, остаются консольными. Ранее мы писали только консольные программы.

Существует множество библиотек GUI, среди которых Tk не самый популярный инструмент, хотя с его помощью написано немало проектов. Он был выбран для Python по-умолчанию. Установочный файл интерпретатора Питона обычно уже включает пакет tkinter в составе стандартной библиотеки.

Tkinter можно представить как переводчик с языка Python на язык Tcl. Вы пишете программу на Python, а код модуля **tkinter** переводит ваши инструкции на язык Tcl, который понимает библиотека Tk.

Программы с графическим интерфейсом пользователя *событийно-ориентированные*. Вы уже должны иметь представление о структурном и желательно объектно-ориентированном программировании. Событийно-ориентированное ориентировано на события. То есть та или иная часть программного кода начинает выполняться лишь тогда, когда случается то или иное событие.

Событийно-ориентированное программирование базируется на объектно-ориентированном и структурном. Даже если мы не создаем собственных классов и объектов, то все-равно ими пользуемся. Все виджеты – объекты, порожденные встроенными классами.

События бывают разными. Сработал временной фактор, кто-то кликнул мышкой или нажал Enter, начал вводить текст, переключил радиокнопки, прокрутил страницу вниз и т.

д. Когда случается что-то подобное, то, если был создан соответствующий обработчик, происходит срабатывание определенной части программы, что приводит к какому-либо результату.

Tkinter импортируется стандартно для модуля Python любым из способов:

- `import tkinter`
- `from tkinter import *`
- `import tkinter as tk`

Можно импортировать отдельные классы, что делается редко. В данном курсе мы будем использовать выражение `from tkinter import *`.

Если необходимо, узнать установленную версию Tk можно через константу TkVersion:

```
>>> from tkinter import *
>>> TkVersion
```

8.6

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.

Последовательность не обязательно такая, но первый и последний пункты всегда остаются на своих местах.

В современных операционных системах любое пользовательское приложение заключено в окно, которое можно назвать главным, так как в нем располагаются все остальные виджеты. Объект окна верхнего уровня создается от класса **`Tk`** модуля **`tkinter`**. Переменную, связываемую с объектом, часто называют *root* (корень):

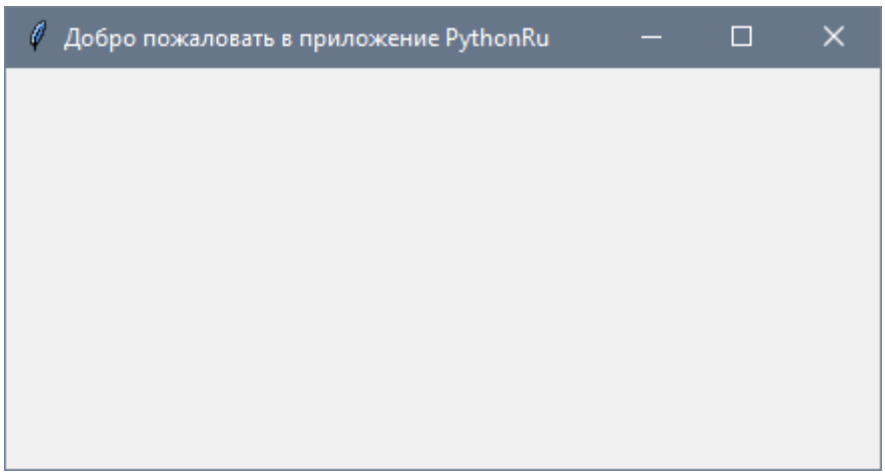
```
root = Tk()
```

2. Создание своего первого графического интерфейса

Для начала, следует импортировать Tkinter и создать окно, в котором мы зададим его название:

```
from tkinter import *
window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.mainloop()
```

Результат будет выглядеть следующим образом:



Последняя строка вызывает функцию mainloop. Эта функция вызывает бесконечный цикл окна, поэтому окно будет ждать любого взаимодействия с пользователем, пока не будет закрыто.

В случае, если вы забудете вызвать функцию mainloop , для пользователя ничего не отобразится.

2.1 Создание виджета Label

Чтобы добавить текст в наш предыдущий пример, мы создадим lbl , с помощью класса Label, например:

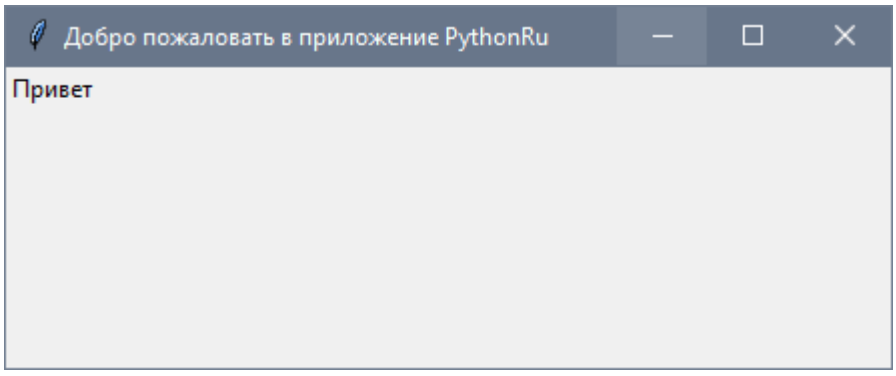
```
lbl = Label(window, text="Привет")
```

Затем мы установим позицию в окне с помощью функции grid и укажем ее следующим образом:

```
lbl.grid(column=0, row=0)
Полный код, будет выглядеть следующим образом:
from tkinter import *
window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
lbl = Label(window, text="Привет")
```

```
lbl.grid(column=0, row=0)
window.mainloop()
```

И вот как будет выглядеть результат:

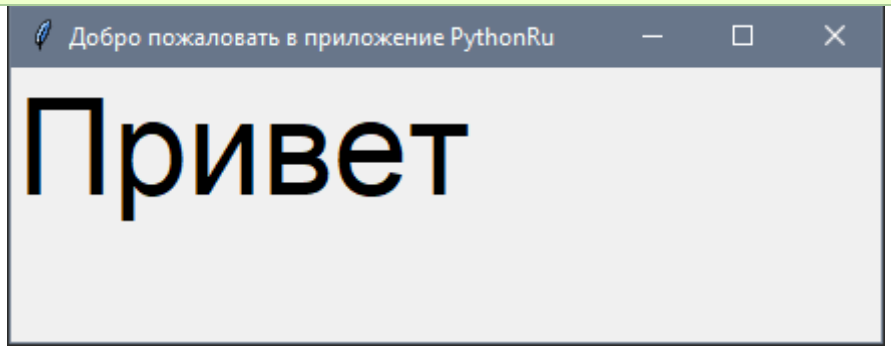


Если функция grid не будет вызвана, текст не будет отображаться.

Настройка размера и шрифта текста

Вы можете задать шрифт текста и размер. Также можно изменить стиль шрифта. Для этого передайте параметр font таким образом:

```
lbl = Label(window, text="Привет", font=("Arial Bold", 50))
```



Обратите внимание, что параметр font может быть передан любому виджету, для того, чтобы поменять его шрифт, он применяется не только к Label.

Отлично, но стандартное окно слишком мало. Как насчет настройки размера окна?

Настройка размеров окна приложения

Мы можем установить размер окна по умолчанию, используя функцию geometry следующим образом:

```
window.geometry('400x250')
```

В приведенной выше строке устанавливается окно шириной до 400 пикселей и высотой до 250 пикселей.

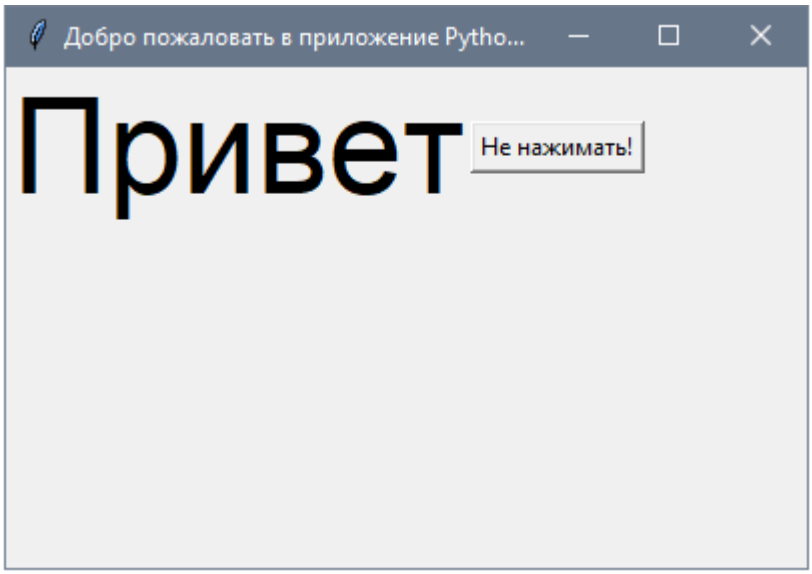
Попробуем добавить больше виджетов GUI, например, кнопки и посмотреть, как обрабатывается нажатие кнопок.

2.2 Добавление виджета Button

Начнем с добавления кнопки в окно. Кнопка создается и добавляется в окно так же, как и метка:

```
btn = Button(window, text="Не нажимать!")
btn.grid(column=1, row=0)
Наш код будет выглядеть вот так:
from tkinter import *

window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
lbl = Label(window, text="Привет", font=("Arial Bold", 50))
lbl.grid(column=0, row=0)
btn = Button(window, text="Не нажимать!")
btn.grid(column=1, row=0)
window.mainloop()
```

Результат будет следующим:



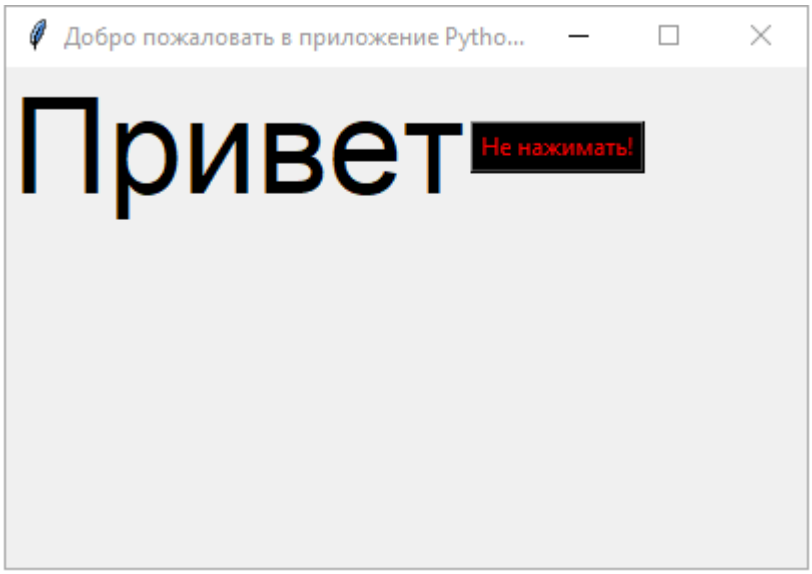
Обратите внимание, что мы помещаем кнопку во второй столбец окна, что равно 1. Если вы забудете и поместите кнопку в том же столбце, который равен 0, он покажет только кнопку.

Изменение цвета текста и фона у Button

Вы можете поменять цвет текста кнопки или любого другого виджета, используя свойство fg.

Кроме того, вы можете поменять цвет фона любого виджета, используя свойство bg.

```
btn = Button(window, text="Не нажимать!", bg="black", fg="red")
```



Теперь, если вы попытаетесь щелкнуть по кнопке, ничего не произойдет, потому что событие нажатия кнопки еще не написано.

Кнопка Click

Для начала, мы запишем функцию, которую нужно выполнить при нажатии кнопки:

```
def clicked():
    lbl.configure(text="Я же просил...")
```

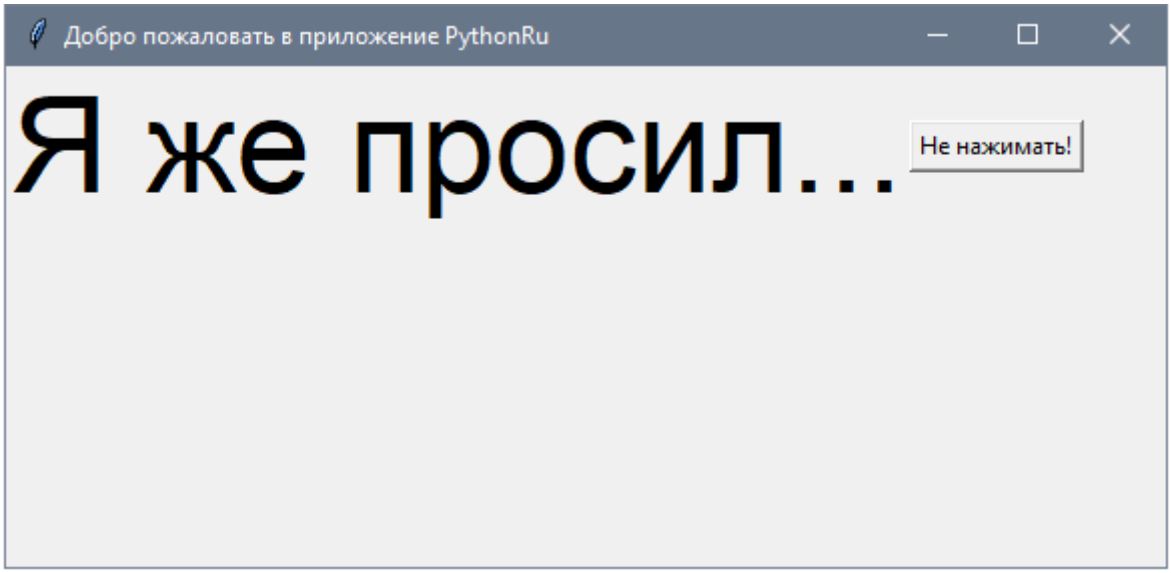
Затем мы подключим ее с помощью кнопки, указав следующую функцию:

```
btn = Button(window, text="Не нажимать!", command=clicked)
```

Обратите внимание: мы пишем clicked, а не clicked()с круглыми скобками. Теперь полный код будет выглядеть так:

```
from tkinter import *


def clicked():
    lbl.configure(text="Я же просил...")


window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
lbl = Label(window, text="Привет", font=("Arial Bold", 50))
lbl.grid(column=0, row=0)
btn = Button(window, text="Не нажимать!", command=clicked)
btn.grid(column=1, row=0)
window.mainloop()
```

При нажатии на кнопку, результат, как и ожидалось, будет выглядеть следующим образом:



2.3 Получение ввода с использованием класса Entry (

В предыдущих примерах GUI Python мы ознакомились со способами добавления простых виджетов, а теперь попробуем получить пользовательский ввод, используя класс Tkinter Entry (текстовое поле Tkinter).

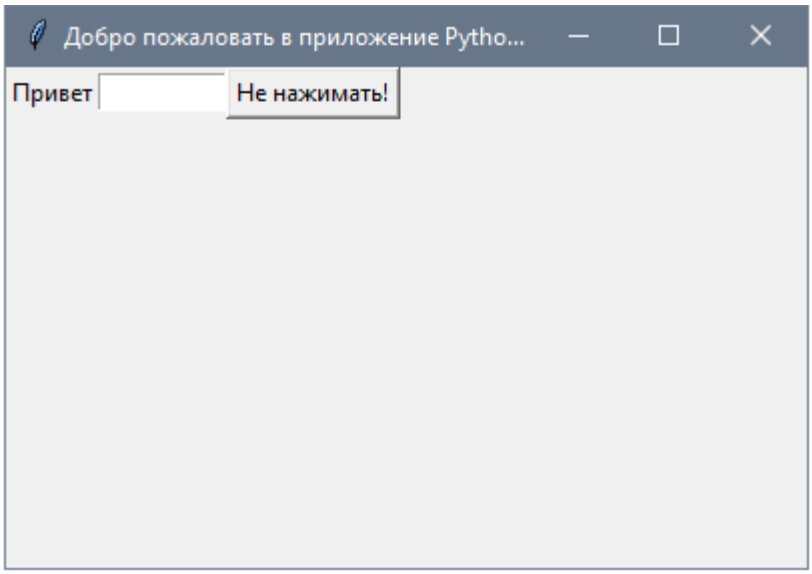
Вы можете создать текстовое поле с помощью класса Tkinter Entry следующим образом:

```
txt = Entry(window, width=10)
```

Затем вы можете добавить его в окно, используя функцию grid. Наше окно будет выглядеть так:

```
from tkinter import *


def clicked():
    lbl.configure(text="Я же просил...")

window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
lbl = Label(window, text="Привет")
lbl.grid(column=0, row=0)
txt = Entry(window,width=10)
txt.grid(column=1, row=0)
btn = Button(window, text="Не нажимать!", command=clicked)
btn.grid(column=2, row=0)
window.mainloop()
```

Полученный результат будет выглядеть так:



Теперь, если вы нажмете кнопку, она покажет то же самое старое сообщение, но что же будет с отображением введенного текста в виджет Entry?

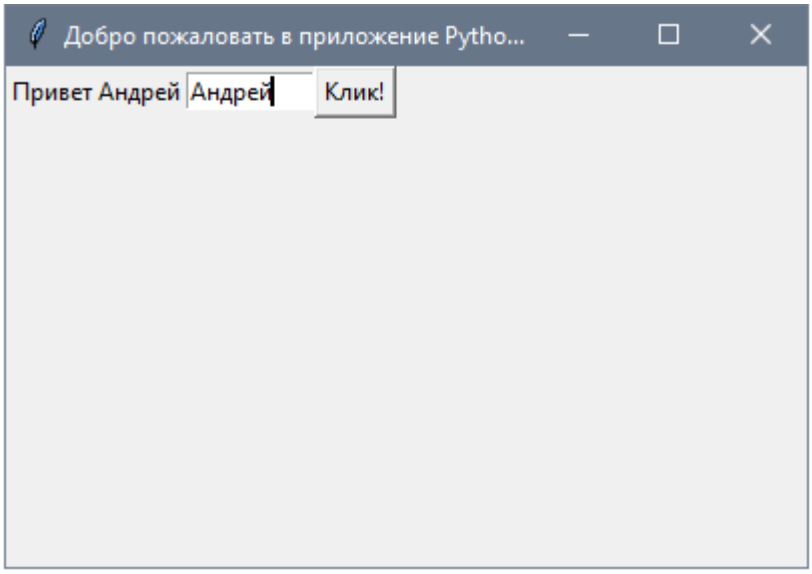
Во-первых, вы можете получить текст ввода, используя функцию get. Мы можем записать код для выбранной функции таким образом:

```
def clicked():
    res = "Привет {}".format(txt.get())
    lbl.configure(text=res)
```

Если вы нажмете на кнопку — появится текст «Привет » вместе с введенным текстом в виджете записи. Вот полный код:

```
from tkinter import *

def clicked():
    res = "Привет {}".format(txt.get())
    lbl.configure(text=res)

window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
lbl = Label(window, text="Привет")
lbl.grid(column=0, row=0)
txt = Entry(window,width=10)
txt.grid(column=1, row=0)
btn = Button(window, text="Клик!", command=clicked)
btn.grid(column=2, row=0)
window.mainloop()
```

Запустите вышеуказанный код и проверьте результат:



Каждый раз, когда мы запускаем код, нам нужно нажать на виджет ввода, чтобы настроить фокус на ввод текста, но как насчет автоматической настройки фокуса?

2.4 Установка фокуса виджета ввода

Здесь все очень просто, ведь все, что нам нужно сделать, — это вызвать функцию focus:

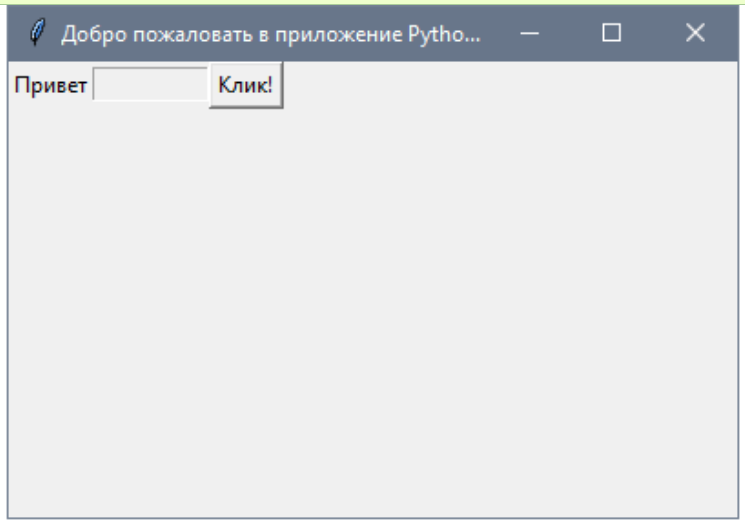
```
txt.focus()
```

Когда вы запустите свой код, вы заметите, что виджет ввода в фокусе, который дает возможность сразу написать текст.

Отключить виджет ввода

Чтобы отключить виджет ввода, отключите свойство состояния:

```
txt = Entry(window,width=10, state='disabled')
```



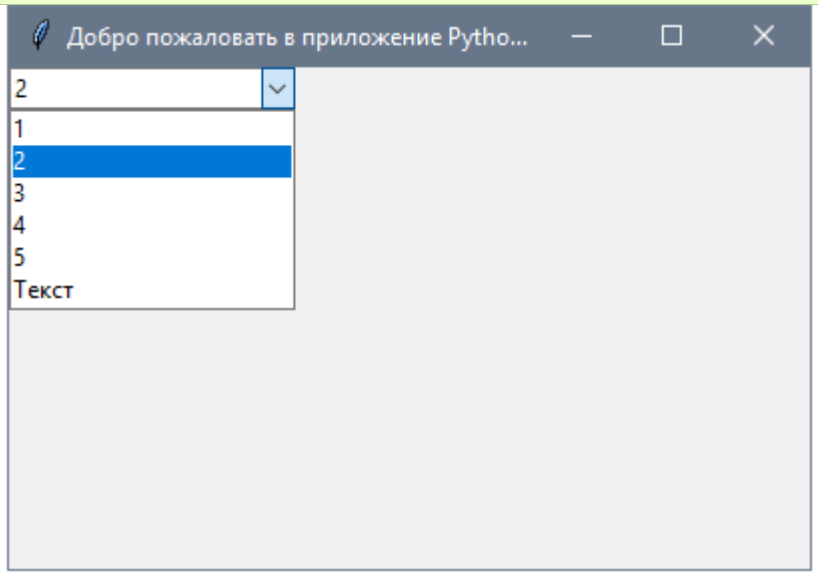
2.5 Добавление виджета Combobox

Чтобы добавить виджет поля с выпадающем списком, используйте класс Combobox из ttk следующим образом:

```
from tkinter.ttk import Combobox
combo = Combobox(window)
```

Затем добавьте свои значения в поле со списком.

```
from tkinter import *
from tkinter.ttk import Combobox

window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
combo = Combobox(window)
combo['values'] = (1, 2, 3, 4, 5, "Текст")
combo.current(1)  # установите вариант по умолчанию
combo.grid(column=0, row=0)
window.mainloop()
```



Как видите с примера, мы добавляем элементы combobox, используя значения tuple.

Чтобы установить выбранный элемент, вы можете передать индекс нужного элемента текущей функции.

Чтобы получить элемент select, вы можете использовать функцию get вот таким образом:

```
combo.get()
```

2.6 Добавление виджета Checkbutton (чекбокса)

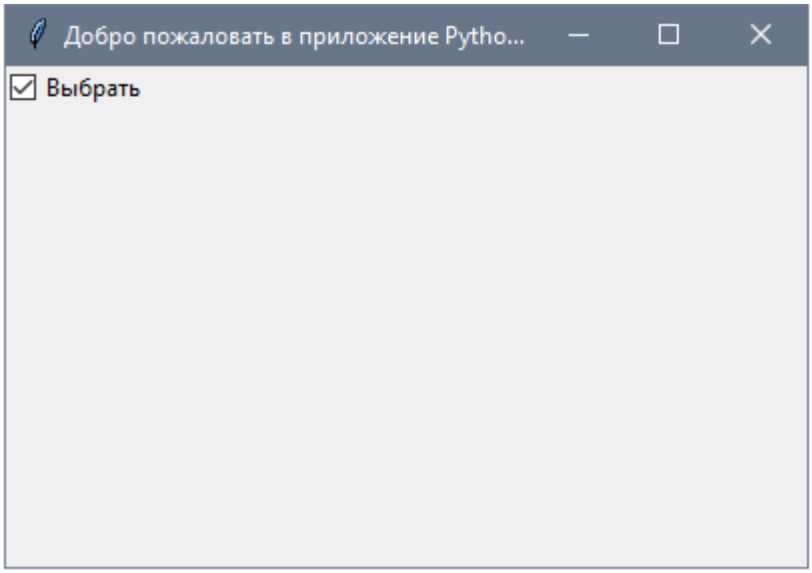
С целью создания виджета checkbutton, используйте класс Checkbutton:

```
from tkinter.ttk import Checkbutton

chk = Checkbutton(window, text='Выбрать')
```

Кроме того, вы можете задать значение по умолчанию, передав его в параметр var в Checkbutton:

```
from tkinter import *
from tkinter.ttk import Checkbutton

window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
chk_state = BooleanVar()
chk_state.set(True)  # задайте проверку состояния чекбокса
chk = Checkbutton(window, text='Выбрать', var=chk_state)
chk.grid(column=0, row=0)
window.mainloop()
```

Посмотрите на результат:



2.7 Установка состояния Checkbutton

Здесь мы создаем переменную типа BooleanVar, которая не является стандартной переменной Python, это переменная Tkinter, затем передаем ее классу Checkbutton, чтобы установить состояние чекбокса как True в приведенном выше примере.

Вы можете установить для BooleanVar значение false, что бы чекбокс не был отмечен.

Так же, используйте IntVar вместо BooleanVar и установите значения 0 и 1.

```
chk_state = IntVar()
chk_state.set(0) # False
chk_state.set(1) # True
```

Эти примеры дают тот же результат, что и BooleanVar.

Добавление виджетов Radio Button

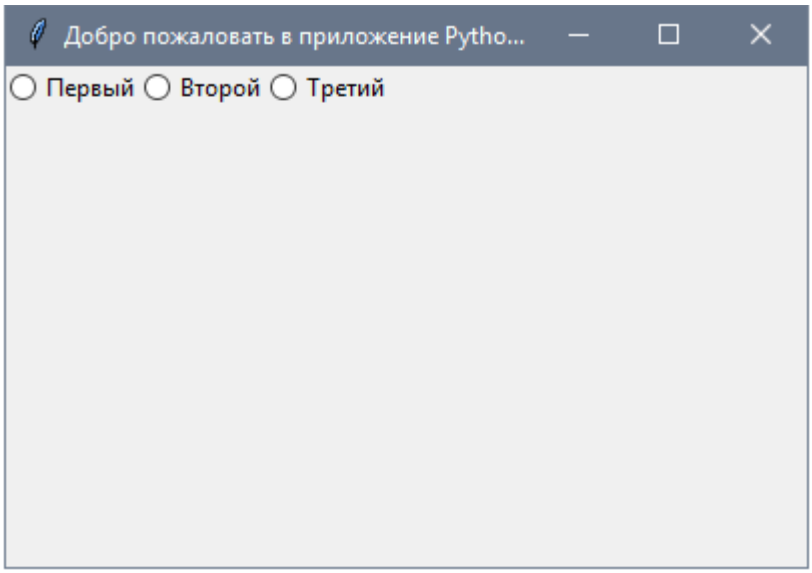
Чтобы добавить radio кнопки, используйте класс RadioButton:

```
rad1 = Radiobutton(window,text='Первый', value=1)
```

Обратите внимание, что вы должны установить value для каждой radio кнопки с уникальным значением, иначе они не будут работать.

```
from tkinter import *
from tkinter.ttk import Radiobutton

window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
rad1 = Radiobutton(window, text='Первый', value=1)
rad2 = Radiobutton(window, text='Второй', value=2)
rad3 = Radiobutton(window, text='Третий', value=3)
rad1.grid(column=0, row=0)
rad2.grid(column=1, row=0)
rad3.grid(column=2, row=0)
window.mainloop()
```

Результатом вышеприведенного кода будет следующий:



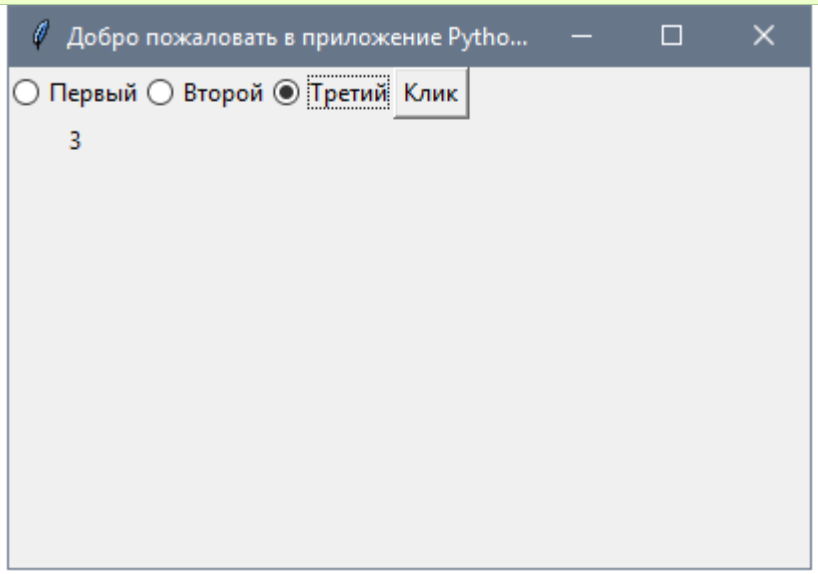
Кроме того, вы можете задать command любой из этих кнопок для определенной функции. Если пользователь нажимает на такую кнопку, она запустит код функции. Вот пример:

```
rad1 = Radiobutton(window,text='Первая', value=1, command=clicked)

def clicked():
```

2.8 Получение значения Radio Button (Избранная Radio Button)

Чтобы получить текущую выбранную radio кнопку или ее значение, вы можете передать параметр переменной и получить его значение.

```
from tkinter import *
from tkinter.ttk import Radiobutton

def clicked():
    lbl.configure(text=selected.get())
window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
selected = IntVar()
rad1 = Radiobutton(window,text='Первый', value=1,
variable=selected)
rad2 = Radiobutton(window,text='Второй', value=2,
variable=selected)
rad3 = Radiobutton(window,text='Третий', value=3,
variable=selected)
btn = Button(window, text="Клик", command=clicked)
lbl = Label(window)
rad1.grid(column=0, row=0)
rad2.grid(column=1, row=0)
rad3.grid(column=2, row=0)
btn.grid(column=3, row=0)
lbl.grid(column=0, row=1)
window.mainloop()
```



Каждый раз, когда вы выбираете radio button, значение переменной будет изменено на значение кнопки.

2.9 Добавление виджета ScrolledText (текстовая область Tkinter)

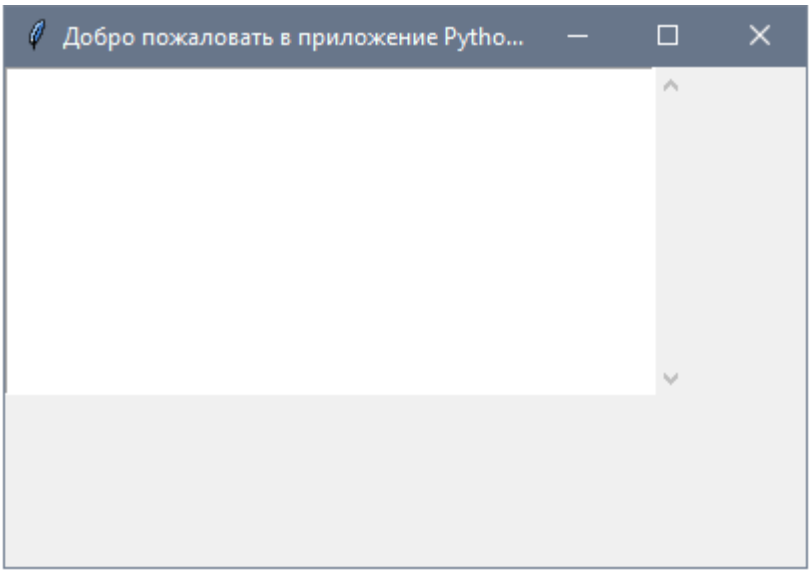
Чтобы добавить виджет ScrolledText, используйте класс ScrolledText:

```
from tkinter import scrolledtext
txt = scrolledtext.ScrolledText(window,width=40,height=10)
```

Здесь нужно указать ширину и высоту ScrolledText, иначе он заполнит все окно.

```
from tkinter import *
from tkinter import scrolledtext

window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
txt = scrolledtext.ScrolledText(window, width=40, height=10)
txt.grid(column=0, row=0)
window.mainloop()
```

Результат:



Используйте метод insert, чтобы настроить содержимое Scrolledtext:

```
txt.insert(INSERT, 'Текстовое поле')
```

Удаление/Очистка содержимого Scrolledtext

Чтобы очистить содержимое данного виджета, используйте метод delete:

```
txt.delete(1.0, END)  # мы передали координаты очистки
```

2.10 Создание всплывающего окна с сообщением

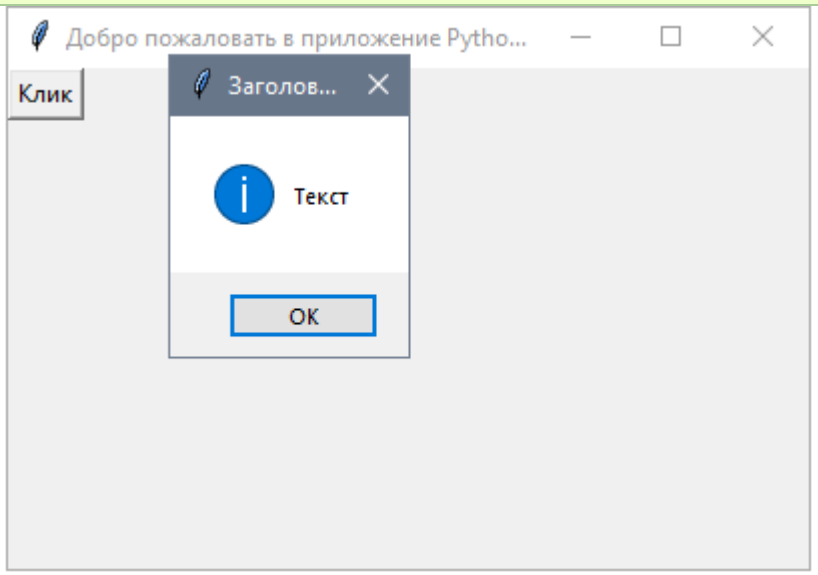
Чтобы показать всплывающее окно с помощью Tkinter, используйте messagebox следующим образом:

```
from tkinter import messagebox
```

```
messagebox.showinfo('Заголовок', 'Текст')
```

Давайте покажем окно сообщений при нажатии на кнопку пользователем.

```
from tkinter import *
from tkinter import messagebox

def clicked():
    messagebox.showinfo('Заголовок', 'Текст')

window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
btn = Button(window, text='Клик', command=clicked)
btn.grid(column=0, row=0)
window.mainloop()
```



Когда вы нажмете на кнопку, появится информационное окно.

Вы можете показать предупреждающее сообщение или сообщение об ошибке таким же образом. Единственное, что нужно изменить—это функция сообщения.

```
messagebox.showwarning('Заголовок', 'Текст')  # показывает
предупреждающее сообщение
messagebox.showerror('Заголовок', 'Текст')  # показывает сообщение
об ошибке
```

2.11 Показ диалоговых окон с выбором варианта

Чтобы показать пользователю сообщение “да/нет”, вы можете использовать одну из следующих функций messagebox:

```
from tkinter import messagebox

res = messagebox.askquestion('Заголовок', 'Текст')
res = messagebox.askyesno('Заголовок', 'Текст')
```

```
res = messagebox.askyesnocancel('Заголовок', 'Текст')
res = messagebox.askokcancel('Заголовок', 'Текст')
res = messagebox.askretrycancel('Заголовок', 'Текст')
```

Вы можете выбрать соответствующий стиль сообщения согласно вашим потребностям. Просто замените строку функции showinfo на одну из предыдущих и запустите скрипт. Кроме того, можно проверить, какая кнопка нажата, используя переменную результата.

Если вы кликнете **OK**, **yes** или **retry**, значение станет **True**, а если выберете **no** или **cancel**, значение будет **False**.

Единственной функцией, которая возвращает одно из трех значений, является функция askyesnocancel; она возвращает **True/False/None**.

2.12 Добавление SpinBox (Виджет спинбокс)

Для создания виджета спинбокса, используйте класс Spinbox:

```
spin = Spinbox(window, from_=0, to=100)
```

Таким образом, мы создаем виджет Spinbox, и передаем параметры from и to, чтобы указать диапазон номеров.

Кроме того, вы можете указать ширину виджета с помощью параметра width:

```
spin = Spinbox(window, from_=0, to=100, width=5)
```

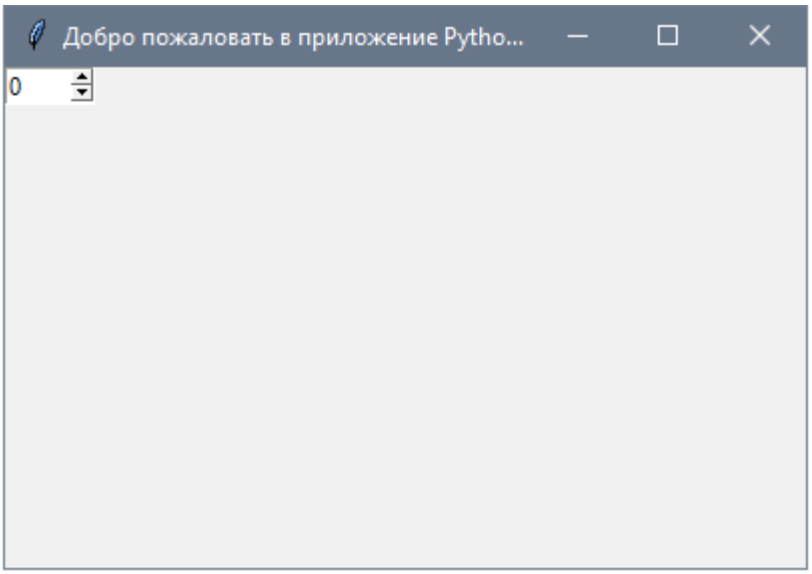
Проверим пример полностью:

```
from tkinter import *


window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
spin = Spinbox(window, from_=0, to=100, width=5)
spin.grid(column=0, row=0)
window.mainloop()
```

Вы можете указать числа для Spinbox, вместо использования всего диапазона следующим образом:

```
spin = Spinbox(window, values=(3, 8, 11), width=5)
```

Виджет покажет только эти 3 числа: 3, 8 и 11.



В случае, если вам нужно задать значение по умолчанию для Spinbox, вы можете передать значение параметру textvariable следующим образом:

```
var = IntVar()
var.set(36)
spin = Spinbox(window, from_=0, to=100, width=5, textvariable=var)
```

Теперь, если вы запустите программу, она покажет 36 как значение по умолчанию для Spinbox.

2.13 Добавление виджета Progressbar

Чтобы создать данный виджет, используйте класс progressbar :

```
from tkinter.ttk import Progressbar

bar = Progressbar(window, length=200)
```

Установите значение progressbar таким образом:

```
bar['value'] = 70
```

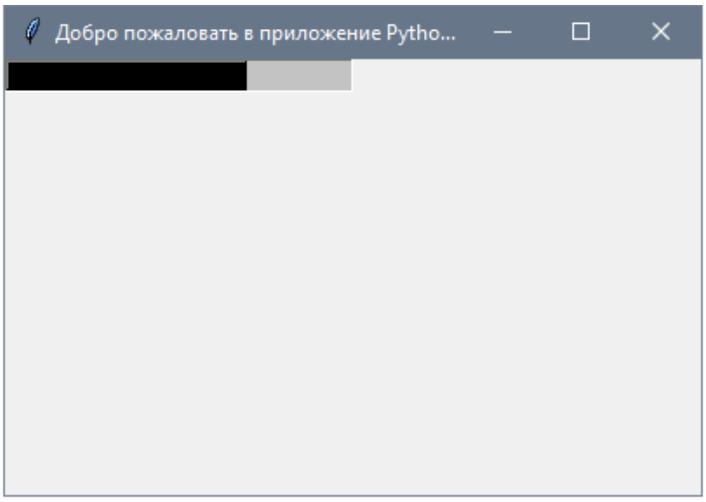
Вы можете установить это значение на основе любого процесса или при выполнении задачи.

Изменение цвета Progressbar немного сложно. Сначала нужно создать стиль и задать цвет фона, а затем настроить созданный стиль на Progressbar. Посмотрите следующий пример:

```
from tkinter import *
from tkinter.ttk import Progressbar
from tkinter import ttk

window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
```

```
window.geometry('400x250')
style = ttk.Style()
style.theme_use('default')
style.configure("black.Horizontal.TProgressbar",
background='black')
bar = Progressbar(window, length=200,
style='black.Horizontal.TProgressbar')
bar['value'] = 70
bar.grid(column=0, row=0)
window.mainloop()
```

И в результате вы получите следующее:



2.14 Добавление поля загрузки файла

Для добавления поля с файлом, используйте класс filedialog:

```
from tkinter import filedialog

file = filedialog.askopenfilename()
```

После того, как вы выберете файл, нажмите “Открыть”; переменная файла будет содержать этот путь к файлу. Кроме того, вы можете запросить несколько файлов:

```
files = filedialog.askopenfilenames()
```

Указание типа файлов (расширение фильтра файлов)

Возможность указания типа файлов доступна при использовании параметра filetypes, однако при этом важно указать расширение в tuples.

```
file = filedialog.askopenfilename(filetypes = (("Text
files","*.txt"),("all files","*.*")))
```

Вы можете запросить каталог, используя метод askdirectory :

```
dir = filedialog.askdirectory()
```

Вы можете указать начальную директорию для диалогового окна файла, указав initialdir следующим образом:

```
from os import path
file = filedialog.askopenfilename(initialdir=
path.dirname(__file__))
```

2.15 Добавление панели меню

Для добавления панели меню, используйте класс menu:

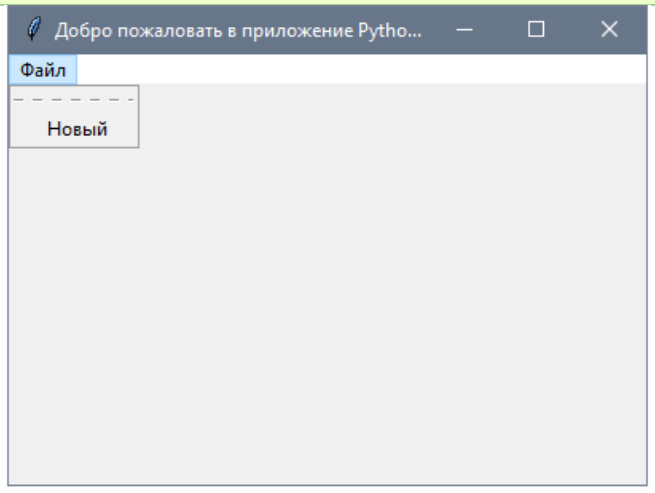
```
from tkinter import Menu

menu = Menu(window)
menu.add_command(label='Файл')
window.config(menu=menu)
```

Сначала мы создаем меню, затем добавляем наш первый пункт подменю. Вы можете добавлять пункты меню в любое меню с помощью функции add_cascade() таким образом:

```
menu.add_cascade(label='Автор', menu=new_item)
```

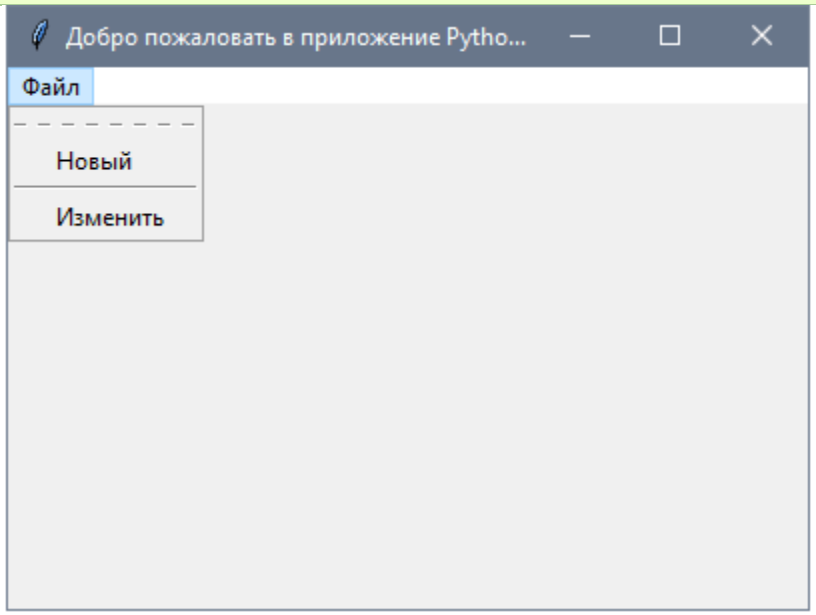
Наш код будет выглядеть так:

```
from tkinter import *
from tkinter import Menu


window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
menu = Menu(window)
new_item = Menu(menu)
new_item.add_command(label='Новый')
menu.add_cascade(label='Файл', menu=new_item)
window.config(menu=menu)
window.mainloop()
```



Таким образом, вы можете добавить столько пунктов меню, сколько захотите.

```
from tkinter import *

window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
menu = Menu(window)
new_item = Menu(menu)
new_item.add_command(label='Новый')
new_item.add_separator()
new_item.add_command(label='Изменить')
menu.add_cascade(label='Файл', menu=new_item)
window.config(menu=menu)
window.mainloop()
```



Теперь мы добавляем еще один пункт меню “Изменить” с разделителем меню. Вы можете заметить пунктирную линию в начале, если вы нажмете на эту строку, она отобразит пункты меню в небольшом отдельном окне.

Можно отключить эту функцию, с помощью tearoff подобным образом:

```
new_item = Menu(menu, tearoff=0)
```

Просто отредактируйте new_item, как в приведенном выше примере и он больше не будет отображать пунктирную линию.

Вы так же можете ввести любой код, который работает, при нажатии пользователем на любой элемент меню, задавая свойство команды.

```
new_item.add_command(label='Новый', command=clicked)
```

2.16 Добавление виджета Notebook (Управление вкладкой)

Для удобного управления вкладками реализуйте следующее:

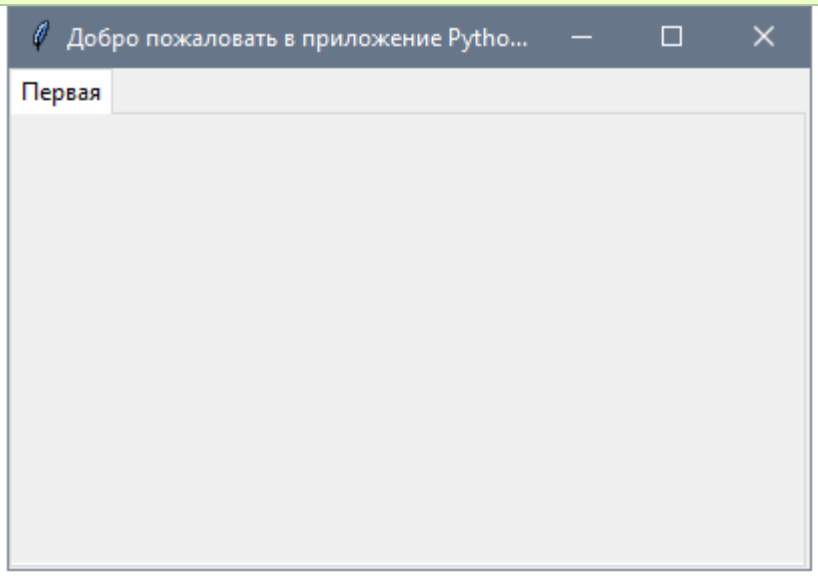
1 Для начала, создается элемент управления вкладкой, с помощью класса Notebook .

2 Создайте вкладку, используя класс Frame.

3 Добавьте эту вкладку в элемент управления вкладками.

4 Запакуйте элемент управления вкладкой, чтобы он стал видимым в окне.

```
from tkinter import *
from tkinter import ttk


window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
tab_control = ttk.Notebook(window)
tab1 = ttk.Frame(tab_control)
tab_control.add(tab1, text='Первая')
tab_control.pack(expand=1, fill='both')
window.mainloop()
```

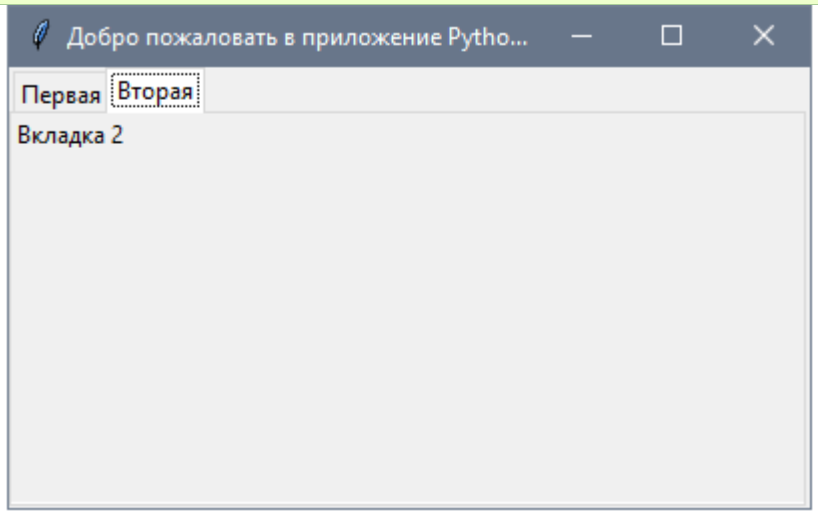


Таким образом, вы можете добавлять столько вкладок, сколько нужно.

После создания вкладок вы можете поместить виджеты внутри этих вкладок, назначив родительское свойство нужной вкладке.

```
from tkinter import *
from tkinter import ttk


window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
tab_control = ttk.Notebook(window)
tab1 = ttk.Frame(tab_control)
tab2 = ttk.Frame(tab_control)
tab_control.add(tab1, text='Первая')
tab_control.add(tab2, text='Вторая')
```

```
lbl1 = Label(tab1, text='Вкладка 1')
lbl1.grid(column=0, row=0)
lbl2 = Label(tab2, text='Вкладка 2')
lbl2.grid(column=0, row=0)
tab_control.pack(expand=1, fill='both')
window.mainloop()
```



Вы можете добавить отступы для элементов управления, чтобы они выглядели хорошо организованными с использованием свойств padx и pady.

Передайте padx и pady любому виджету и задайте значение.

```
lbl1 = Label(tab1, text= 'label1', padx=5, pady=5)
```

2.17 Виджет Text — ввод большого текста в Tkinter

Виджеты Text используются для ввода текста. Text может содержать несколько строчек текста. С виджетом Text пользователь может вводить целые параграфы или страницы текста. Как и в случае с виджетами Entry, над виджетами Text можно провести три основные операции:

- Получение текста через метод .get()
- Удаление текста через метод .delete()
- Вставка текста через метод .insert()

Разберем пример **создания виджета Text** и посмотрим на деле, что он может делать.

В оболочке Python нужно создать новое пустое окно, а внутри нее с помощью метода .pack() разместить текстовой виджет:

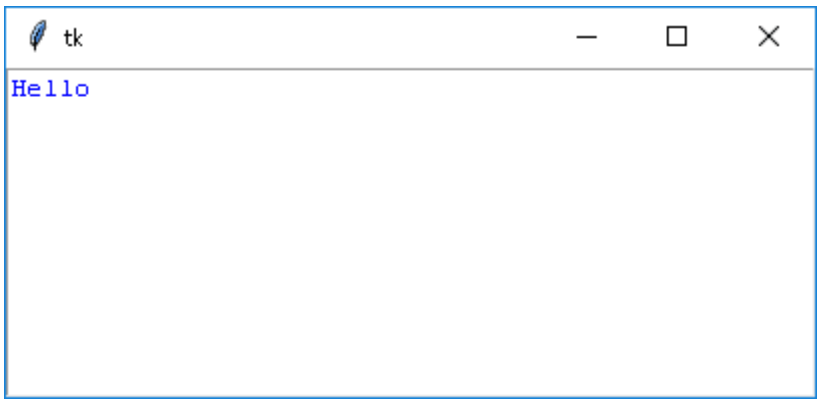
Python

```
from tkinter import *
root = Tk()

text = Text(width=50, height=10, bg="white",
fg='blue', wrap=WORD)
```

```
text.pack()
root.mainloop()
```

По умолчанию текстовые боксы значительно больше виджетов однострочного ввода текста Entry.

Для активации текстового бокса нажмите на любую точку внутри окна. Введите слово "Hello". Нажмите на клавиатуре кнопку *Enter*, после чего введите на второй строке слово "World". Окно будет выглядеть следующим образом:



Вы можете получить текст их виджета Text с помощью метода .get(). Однако, вызов .get() без аргументов не возвращает весь текст как это происходит с виджетами однострочного ввода текста Entry. Метод .get() для текстового виджета запрашивает хотя бы один аргумент. Вызов .get() с единственным индексом возвращает один символ. Для получения нескольких символов требуется передать начальный и конечный индексы.

Индексы в виджетах Text . Так как виджеты Text могут получить несколько строк текста, индекс должен содержать следующие два пункта:

Номер строки где находится символа;

Позиция символа в строке.

Номера строк начинаются с 1, а позиция символа с 0. Для получения индекса создается строка формата "<line>.<char>", где <line> заменяется номером строки, а <char> номером символа. К примеру, "1.0" означает первый символ на первой строке, а "2.3" представляет четвертый символ на второй строке.

Используем индекс "1.0" для получения первой буквы созданного ранее текстового бокса:

```
>>> text_box.get("1.0")
'H'
```

В слове "Hello" 5 букв, буква о стоит под индексом 4, так как отсчет символов начинается с 0, и слово "Hello" начинается с первой строки в текстовом боксе.

Как при **слайсинге строк в Python**, для получения целого слова "Hello" из текстового бокса, конечный индекс должен быть на один больше последнего читаемого символа.

Таким образом, для получения слова "Hello" из текстового бокса используется "1.0" для первого индекса, и "1.5" для второго индекса:

```
>>> text_box.get("1.0", "1.5")
'Hello'
```

Для **получения всего текста из текстового виджета** установите индекс на "1.0" и используйте специальную константу tk.END для второго индекса:

```
>>> text_box.get("1.0", tk.END)
```

Обратите внимание, что текст, возвращаемый через метод .get(), включает символы перехода на новую строку \n. Вы также можете увидеть в данном примере, что каждая строка в виджете Text содержит в конце символ новой строки, включая последнюю строку текста в текстовом боксе.

Метод .delete() используется для удаления символов из текстового виджета. Работает точно так же, как и метод .delete() для виджетов Entry. Есть два способа использования .delete():

С одним аргументом;

С двумя аргументами.

Используя вариант с одним аргументом, вы передаете индекс символа для удаления в .delete(). К примеру, следующий код удаляет первый символ H из текстового бокса:

```
>>> text_box.delete("1.0")
```

Теперь в первой строке текста значится "ello":

В версии с двумя аргументами передается два индекса для удаления группы символов, начиная с первого символа и заканчивая вторым, но не включая его.

К примеру, для удаления оставшейся части "ello" из первой строки текстового бокса используются индексы "1.0" и "1.4":

```
>>> text_box.delete("1.0", "1.4")
```

Вы также можете вставить текст в текстовый виджет с помощью метода .insert().

```
>>> text_box.insert("1.0", "Hello")
```

Метод вставляет слово "Hello" в начало текстового виджета. Используется уже знакомый формат "<line>.<column>" как и у метода .get() для уточнения позиции вставки текста.

```
>>> text_box.insert("2.0", "World")
```

Вместо вставки слова на вторую строку текст ставится в конце первой строки.

Если вы хотите поставить текст на новую строку, тогда понадобится вставить символ новой строки \n вручную.

```
>>> text_box.insert("2.0", "\nWorld")
```

Теперь слово "World" находится на второй строке текстового виджета.

Метод .insert() делает следующие две вещи:

Вставляет текст в указанной позиции, если в этой позиции или после неё уже находится текст;

Добавляет текст в указанную строку, если номер символа превышает индекс последнего символа в текстовом боксе.

Обычно нецелесообразно пытаться отследить индекс последнего символа. Лучший способ вставить текст в конец виджета Text — передать tk.END как первый параметр в методе .insert():

```
text_box.insert(tk.END, "Вставь меня в самом конце!")
```

Не забудьте включить символ новой строки (\n) в начале текста, если вы хотите расположить его на новой строке:

```
text_box.insert(tk.END, "\nВставь меня в новую строку!")
```