# Библиотека Tkinter

Разработка графических пользовательских интерфейсов

# Tkinter

Это 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.

GUI - это 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 вместе называют виджетами (widgets). Этапы разработки интерфейса с GUI следующие:

1. Импорт библиотеки;
2. Создание главного окна;
3. Создание виджетов;
4. Установка их свойств;
5. Определение событий;
6. Определение обработчиков событий;
7. Расположение виджетов на главном окне;
8. Отображение главного окна.

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()
```

Чтобы добавить текст в наш предыдущий пример, мы создадим `lbl` , с помо

```
lbl
```

Зате кем ее сл

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

```
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()
```

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))
```

# 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.

# Дс

```
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()
```

# 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")
```

# 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)
```

```
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()
```

```
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()
```

Затем

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

## 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')
```

# 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)
```

```
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)
```

# Позиционирование. Pack

Для позиционирования виджетов в контейнере применяются различные способы. Один из них представляет вызов у виджета метода **pack()**. Этот метод принимает следующие параметры:

## Растяжение виджета

Для растяжения виджета применяется параметру **expand** передается значение True (или соответствующее число). Причем при отсутствии других параметров позиционирования значение `expand=True` позволяет поместить виджет по центру:

## Заполнение контейнера

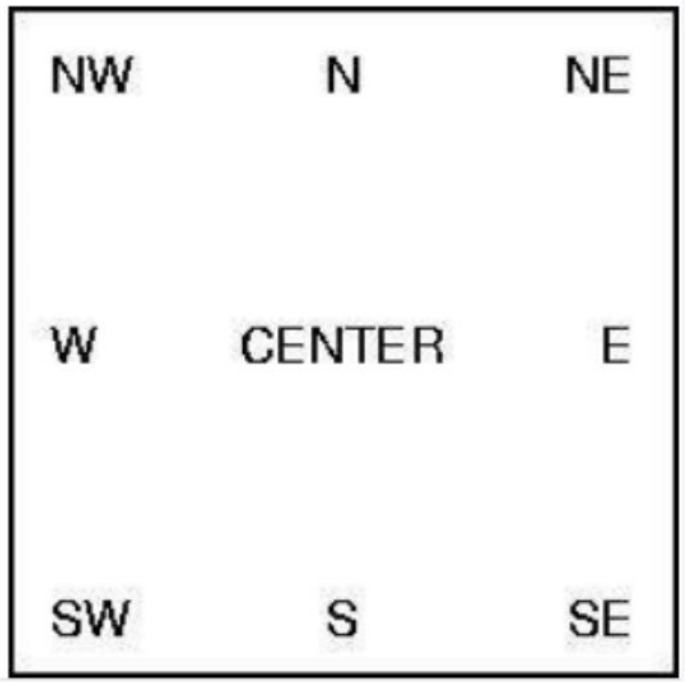
Параметр **fill** позволяет заполнить пространство контейнер по вертикали (значение X), по вертикали (значение Y) или по обеим сторонам (значение BOTH). По умолчанию значение NONE, при котором заполнение контейнера отсутствует. Например, заполним все пространство контейнера по горизонтали

## Anchor

Параметр **anchor** помещает виджет в определенной части контейнера. Может принимать следующие значения:

- n: положение вверху по центру
- e: положение в правой части контейнера по центру
- s: положение внизу по центру
- w: положение в левой части контейнера по центру
- nw: положение в верхнем левом углу
- ne: положение в верхнем правом углу
- se: положение в нижнем правом углу
- sw: положение в нижнем левом углу
- center: положение центру

Схематически это выглядит следующим образом:




Стоит отметить, что значение в кавычках для параметра anchor передается в нижнем регистре, без кавычек - в верхнем регистре

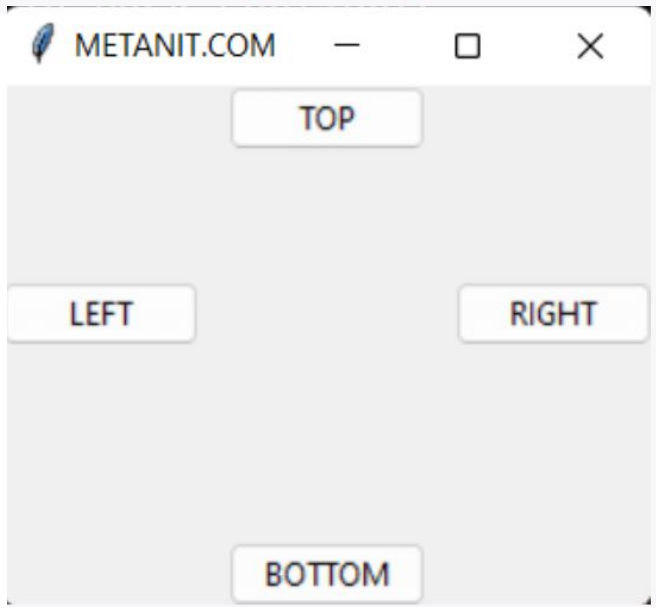
```
1 btn.pack(anchor="nw")
2 btn.pack(anchor=NW)
```

## Отступы

Параметры **padx** и **pady** позволяют указать отступы виджета от границ контейнера:

## Позиционирование по стороне

Используем параметр **side**:



Элемент Entry представляет поле для ввода текста. С помощью конструктора Entry можно установить ряд параметров, основные из них:

- **background**: фоновый цвет
- **cursor**: курсор указателя мыши при наведении на текстовое поле
- **foreground**: цвет текста
- **font**: шрифт текста
- **justify**: устанавливает выравнивание текста. Значение LEFT выравнивает текст по левому краю, CENTER - по центру, RIGHT - по правому краю
- **show**: задает маску для вводимых символов
- **state**: состояние элемента, может принимать значения NORMAL (по умолчанию) и DISABLED
- **textvariable**: устанавливает привязку к элементу StringVar
- **width**: ширина элемента

Элемент Entry имеет ряд методов. Основные из них:

- **insert(index, str)**: вставляет в текстовое поле строку по определенному индексу
- **get()**: возвращает введенный в текстовое поле текст
- **delete(first, last=None)**: удаляет символ по индексу first. Если указан параметр last, то удаление производится до индекса last. Чтобы удалить до конца, в качестве второго параметра можно использовать значение END.
- **focus()**: установить фокус на текстовое поле

# Меню

Для создания иерархического меню в tkinter применяется виджет **Menu**. Основные параметры Menu:

- **activebackground**: цвет активного пункта меню
- **activeborderwidth**: толщина границы активного пункта меню
- **activeforeground**: цвет текста активного пункта меню
- **background / bg**: фоновый цвет
- **bd**: толщина границы
- **cursor**: курсор указателя мыши при наведении на меню

- **disabledforeground**: цвет, когда меню находится в состоянии DISABLED
- **font**: шрифт текста
- **foreground / fg**: цвет текста
- **tearoff**: меню может быть отсоединено от графического окна. В частности, при создании подменю а скриншоте можно увидеть прерывающуюся линию в верху подменю, за которую его можно отсоединить. Однако при значении `tearoff=0` подменю не сможет быть отсоединено.

Меню может содержать много элементов, причем эти элементы сами могут представлять меню и содержать другие элементы. В зависимости от того, какой тип элементов мы хотим добавить в меню, будет отличаться метод, используемый для их добавления. В частности, нам доступны следующие методы:

- **add_command(options)**: добавляет элемент меню через параметр options
- **add_cascade(options)**: добавляет элемент меню, который в свою очередь может представлять подменю
- **add_separator()**: добавляет линию-разграничитель
- **add_radiobutton(options)**: добавляет в меню переключатель
- **add_checkbutton(options)**: добавляет в меню флажок

Создадим простейшее меню:

```
1  from tkinter import *
2
3  root = Tk()
4  root.title("METANIT.COM")
5  root.geometry("250x150")
6
7  main_menu = Menu()
8  main_menu.add_cascade(label="File")
9  main_menu.add_cascade(label="Edit")
10 main_menu.add_cascade(label="View")
11
12 root.config(menu=main_menu)
13 root.mainloop()
```

# Добавим подменю

```
file_menu = Menu()
file_menu.add_command(label="New")
file_menu.add_command(label="Save")
file_menu.add_command(label="Open")
file_menu.add_separator()
file_menu.add_command(label="Exit")

main_menu.add_cascade(label="File", menu=file_menu)
main_menu.add_cascade(label="Edit")
main_menu.add_cascade(label="View")
```

```
root.option_add("*tearOff", FALSE)
```

# Взаимодействие с меню

Отличительной особенностью элементов меню является способность реагировать на нажатия пользователя. Для этого у каждого элемента меню можно задать параметр **command**, который устанавливает ссылку на функцию, выполняемую при нажатии.

```
10 def edit_click():
11     messagebox.showinfo("GUI Python", "Нажата опция Edit")
```

```
15 main_menu.add_cascade(label="File")
16 main_menu.add_cascade(label="Edit", command=edit_click)
17 main_menu.add_cascade(label="View")
```

```
2 from tkinter import messagebox
```