Базовый проект

Основные виджеты tkinter

Меню

- Меню – это виджет, который присутствует во многих пользовательских приложениях. Находится оно под строкой заголовка и представляет собой выпадающие списки под словами-пунктами меню. Пункты конечных списков представляют собой команды, обычно выполняющие какое-либо действия или открывающие диалоговые окна.

Меню

- В tkinter экземпляр меню создается от класса Menu, далее его надо привязать к виджету, на котором оно будет расположено. Обычно таковым выступает главное окно приложения.
- Создать его можно следующим образом:
- from tkinter import *
- Form = Tk()
- mainmenu = Menu(Form)
- Form.config(menu=mainmenu)
- Form.mainloop()

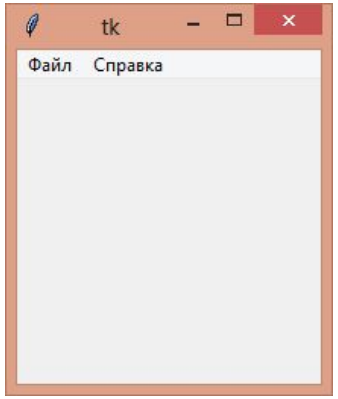
Меню

- Если выполнить данный код, то никакого меню вы не увидите. Только тонкую полоску под заголовком окна, ведь ни одного пункта меню не было создано. Метод add_command() добавляет пункт меню:
-
- mainmenu.add_command(label='Файл')
- mainmenu.add_command(label='Справка')
-

Меню

- В итоге получаем следующий код:
- from tkinter import *
- Form = Tk()
- mainmenu = Menu(Form)
- Form.config(menu=mainmenu)
- mainmenu.add_command(label='Файл')
- mainmenu.add_command(label='Справка')
-
- Form.mainloop()

- И окно с пунктами меню



Меню

- В данном случае "Файл" и "Справка" – это команды. К ним можно добавить опцию command, связав тем самым с какой-либо функцией-обработчиком клика.
- Хотя такой вариант меню имеет право на существование, в большинстве приложений панель меню содержит выпадающие списки команд, а сами пункты на панели командами по сути не являются.
- Клик по ним приводит лишь к раскрытию соответствующего списка.

Меню

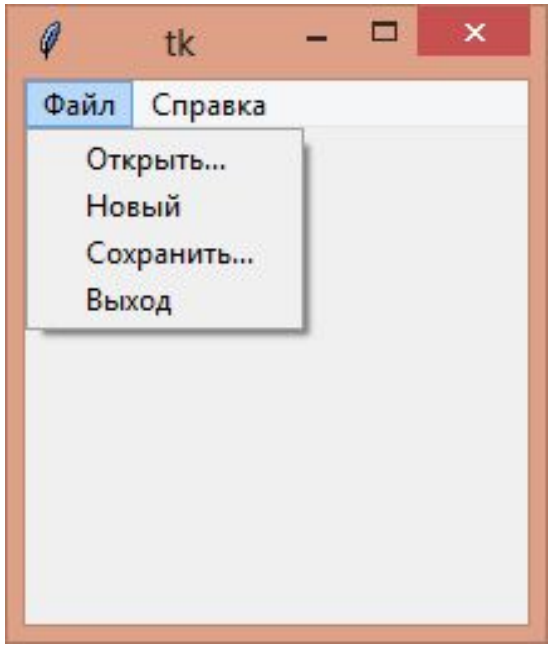
- В tkinter проблема решается созданием новых экземпляров Menu и подвязыванием их к главному меню с помощью метода add_cascade() или add_command(). Их различия вы увидите в нижележащем списке:
-
- add_cascade(options): добавляет элемент меню, который в свою очередь может представлять подменю
- add_command(options): добавляет элемент меню через параметр options
- add_separator(): добавляет линию-разграничитель пунктов меню

Меню

- Если необходимо настроить меню, то мы можем задать в конструкторе Menu следующие опции:
- activebackground: цвет активного пункта меню
- activeborderwidth: толщина границы активного пункта меню
- activeforeground: цвет текста активного пункта меню
- 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
- 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

```
from tkinter import *  #Пример конструирования меню
def v_click():
    Form.destroy()
Form = Tk()
mainmenu = Menu(Form)
Form.config(menu=mainmenu)
filemenu = Menu(mainmenu, tearoff=0) #Создаем каскад для "Файл"
filemenu.add_command(label="Открыть")
filemenu.add_command(label="Новый")
filemenu.add_command(label="Сохранить")
filemenu.add_command(label="Выход",command=v_click)#Закрытие окна
helpmenu = Menu(mainmenu, tearoff=0) #Создаем каскад для "Помощь"
helpmenu.add_command(label="Помощь")
helpmenu.add_command(label="О программе")
mainmenu.add_cascade(label="Файл", menu=filemenu) #Привязываем подменю к меню
mainmenu.add_cascade(label="Справка", menu=helpmenu)
 Form.mainloop()
```



Меню

- На основное меню (mainmenu), добавляются не команды, а другие меню. У filemenu и helpmenu в качестве родительского виджета указывается не Form, а mainmenu. Команды добавляются только к дочерним меню. Значение 0 опции tearoff отключает возможность откреплення подменю, иначе его можно было бы делать плавающим кликом мыши по специальной линии (в случае tearoff=0 она отсутствует).
- Точно также можно подвязывать дочерние меню к filemenu и helpmenu, создавая многоуровневые списки пунктов меню

Меню

- В tkinter можно создать всплывающее меню, оно же контекстное (если настроить его появление по клику правой кнопкой мыши). Для этого экземпляр меню подвязывается не через опцию menu к родительскому виджету, а к меню применяется метод *post(),* аргументами которого являются координаты того места, где должно появляться меню.

Многооконные программы

- Во многих случаях (даже в большинстве случаев) GUI имеет в своем составе не одно, а несколько окон. Первое (главное, main) окно является производным от класса Tk. Оно является базовым (родительским) для всех остальных окон-потомков.
- Закрытие главного окна приводит к выходу из программы. При закрытии «потомка» выполнение программы продолжается.
- Разумеется, у «детей» могут быть «внуки» - окна, для которых окно-потомок является базовым.

Многооконные программы

- Создать *Form2* – окно-потомок ,базового окна *Form* можно следующим способом
-
- Form2=Toplevel(Form)
-
- Методы у всех окон одинаковы. Приведем здесь те, которые нам на этом этапе представляются наиболее необходимыми:

Многооконные программы

- title –заголовок окна.
- withdraw – "спрятать" (сделать невидимым, а не закрыть!) окно. Для того, чтобы снова показать его, надо использовать метод *deiconify*
- resizable – может ли пользователь изменять размер окна. Принимает два аргумента - возможность изменения размера по горизонтали и по вертикали.
- geometry – устанавливает геометрию окна в формате ширинахвысота+x+y (пример: geometry("600x400+40+80") - поместить окно в точку с координатам 40,80 и установить размер в 600x400). Размер или координаты могут быть опущены (geometry("600x400") - только изменить размер, geometry("+40+80") - только переместить окно).
- destroy - уничтожение виджета и всех его потомков. Общий метод для всех виджетов.

Многооконные программы

- Приведем пример двухоконной программы. Пример также демонстрирует взаимодействие между двумя окнами:
- s=txt.get()#Запрашиваем текст из Entry второго окна
- Label1.configure(text=s)#И передаем его в Label базового
- Он демонстрирует передачу данных из одного окна в другое. Подобного рода методы часто используются в программах tkinter.

Многооконные программы

- #Многооконное приложение
- from tkinter import*
- Form = Tk()#Создаем базовое окно
- Form.title('Два окна')
- Form.geometry('400x200+200+200')
- Form.resizable(False, False)
- #Label базового окна
- Label1 = Label(Form, text='Привет',fg='blue',font=("Candara", 30))
- Label1.place(x=120,y=60)

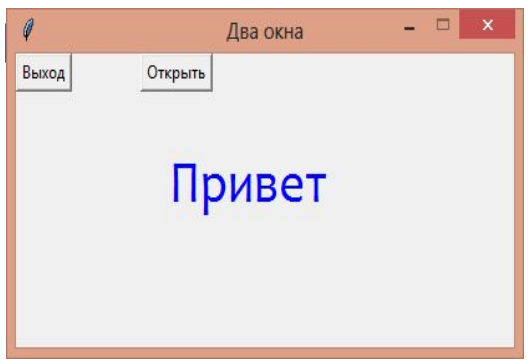
Многооконные программы

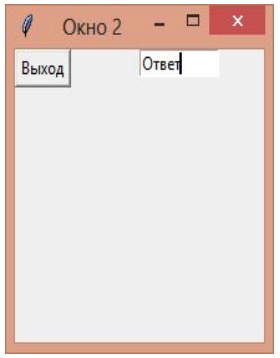
- def clicked1(): #Удаление базового окна, закрытие программы
- Form.destroy()
- def clicked2():#Функция создания и работы второго окна
- def clicked3():#Удаление второго окна
- s=txt.get()#Запрашиваем текст из Entry второго окна
- Label1.configure(text=s)#И передаем его в Label базового
- Form2.destroy()#удаление второго окна

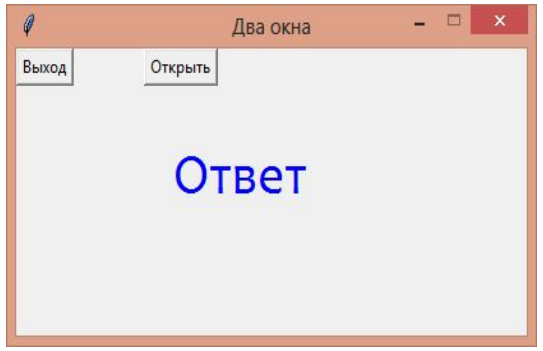
Многооконные программы

- Form2=Toplevel(Form) #Создание второго окна
- Form2.title('Окно 2')
- Form2.geometry('200x200+600+300')
- Button3 = Button(Form2, text="Выход",command=clicked3)#Выход окна 2
- Button3.place(x=0, y=0)
- txt = Entry(Form2, width=10)#Entry окна 2
- txt.place(x=100, y=0)

Многооконные программы

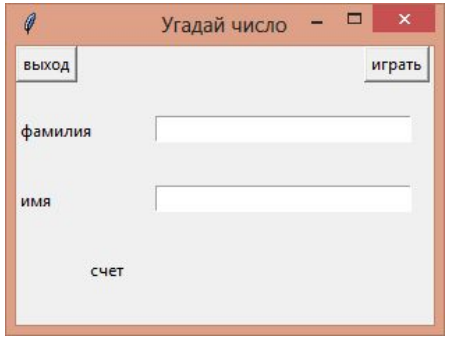
- Button1 = Button(Form, text="Выход",command=clicked1)#Выход окна 1
- Button1.place(x=0, y=0)
- Button2 = Button(Form, text="Открыть",command=clicked2))#Открытие окна 2
- Button2.place(x=100, y=0)
- 

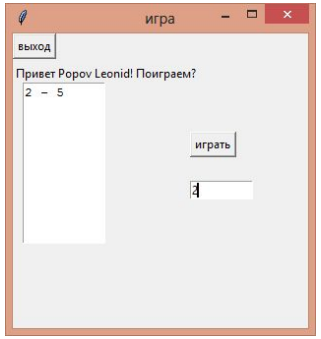




Учебная программа – игра «Угадай число»

- Эта учебная программа поможет вам в освоении методов виджетов, о которых речь шла в предыдущих разделах.
- Суть игры проста: игрок загадывает и вводит цифру, а программа генерирует случайное число. Если эти числа совпали, игрок получает бонусы. Код программы с комментариями приведен ниже. Ее окна имеют следующий





Учебная программа – игра «Угадай число»

- from tkinter import *
- import random
- Form=Tk()
- Form.geometry('300x200+200+200')
- Form.title("Угадай число")

Учебная программа – игра «Угадай число»

- #Игрок вводит фамилию и имя
- Label1 = Label(Form, text="фамилия")
- Label1.place(x=0,y=50)
- Label2 = Label(Form, text="имя")
- Label2.place(x=0,y=100)
- txt1 = Entry(Form, width=30)
- txt1.place(x=100, y=50)
- txt2 = Entry(Form, width=30)
- txt2.place(x=100, y=100)

Учебная программа – игра «Угадай число»

```
def clicked1():#Выход из игры
    Form.destroy()
def clicked2():#Игра
    def clicked3():#Закрываем вторую форму, передаем очки на первую форму
        global k
        s="счет = "+str(k)
        Label4.configure(text=s)
        Form2.destroy()
```

Учебная программа – игра «Угадай число»

```
def clicked4():#Делаем ход
    global k,c
    c=c+1#Считаем ходы
    s=txt3.get()#Запрашиваем введенное игроком число
    f=s.isnumeric()#Если в строке только числа
```

Учебная программа – игра «Угадай число»

```
    if f:
        d=random.randint(1,6)#Генерируем случайное число
        if d==int(s):#Проверка совпадения
            k=k+1
        s=s+" - "+str(d)+"\n"
        text1.insert('end',s)#Выводим через тире цифру игрока и случайную
    else:
        txt3.delete(0,END)#Если ввели не цифру
    if c==5:#Действия после пятого хода
        txt3.config(state=DISABLED)
        Button4.config(state=DISABLED)
        c=0
```

Учебная программа – игра «Угадай число»

- Form2=Toplevel(Form)
- Form2.title(“Игра")
- Form2.geometry('300x300+600+300')
- s="Привет "+txt1.get()+" "+txt2.get()+"! Поиграем?"
- Label3 = Label(Form2, text=s)
- Label3.place(x=0,y=30)
- Button3 = Button(Form2, text=“Выход",command=clicked3)
- Button3.place(x=0, y=0)

Учебная программа – игра «Угадай число»

- txt3 = Entry(Form2, width=10)
- txt3.place(x=180, y=150)
- text1 = Text(Form2, width=10,height=10,wrap=WORD)
- text1.place(x=10, y=50)
- Button4 = Button(Form2, text=“Играть",command=clicked4)
- Button4.place(x=180, y=100)

Учебная программа – игра «Угадай число»

- Button1 = Button(Form, text=“Выход",command=clicked1)
- Button1.place(x=0, y=0)
- Button2 = Button(Form, text="Играть",command=clicked2)
- Button2.place(x=250, y=0)
- Label4 = Label(Form, text="Счет")
- Label4.place(x=50,y=150)
- Form.mainloop()

Предлагаем самостоятельно добавить сохранение результатов игрока в текстовом файле.