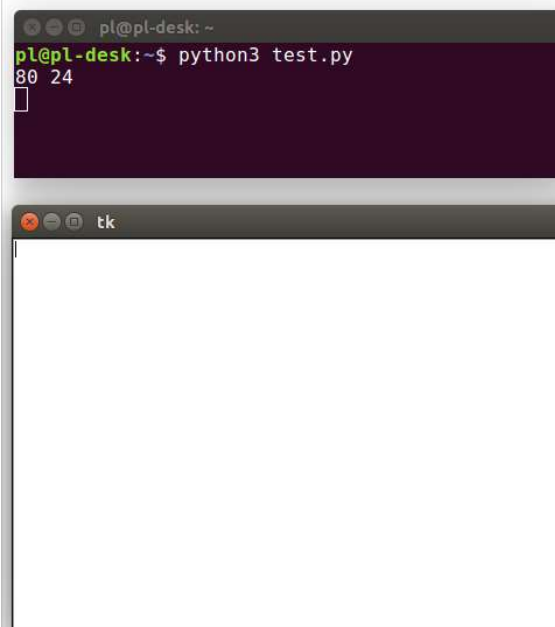


Урок 4. Text – многострочное текстовое поле

В этом уроке рассмотрим, как с помощью Tkinter запрограммировать такой элемент интерфейса как многострочное текстовое поле. Этот виджет часто встречается при заполнении веб-форм. В приложениях для десктопов он редок, если не считать программы "Терминал", где по-сути вы работаете в большом текстовом поле.

В tkinter многострочное текстовое поле создается от класса Text. По умолчанию его размер равен 80-ти знакам по горизонтали и 24-м по вертикали.

```
from tkinter import *
root = Tk()
text = Text()
print(text['width'], text['height'])
text.pack()
root.mainloop()
```

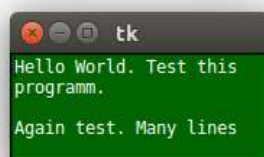


Однако эти свойства можно изменять с помощью опций width и height. Есть возможность конфигурировать шрифт, цвета и другое.

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

text = Text(width=25, height=5, bg="darkgreen", fg='white', wrap=WORD)

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



Значение WORD опции wrap позволяет переносить слова на новую строку целиком, а не по буквам.

Text и Scrollbar

Если в текстовое поле вводится больше линий текста, чем его высота, то оно само будет прокручиваться вниз. При просмотре прокручивать вверх-вниз можно с помощью колеса мыши и стрелками на клавиатуре. Однако бывает удобнее пользоваться скроллером – полосой прокрутки.

В tkinter скроллеры производятся от класса `Scrollbar`. Объект-скроллер связывают с виджетом, которому он требуется. Это не обязательно многострочное текстовое поле. Часто полосы прокрутки бывают нужны спискам, которые будут рассмотрены позже.

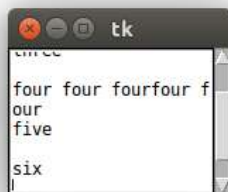
```
from tkinter import *
root = Tk()

text = Text(width=20, height=7)
text.pack(side=LEFT)

scroll = Scrollbar(command=text.yview)
scroll.pack(side=LEFT, fill=Y)

text.config(yscrollcommand=scroll.set)

root.mainloop()
```



Здесь создается скроллер, к которому с помощью опции `command` привязывается прокрутка текстового поля по оси `y` – `text.yview`. В свою очередь текстовому полю опцией `yscrollcommand` устанавливается ранее созданный скроллер – `scroll.set`.

Методы Text

Основные методы у `Text` такие же как у `Entry` – `get()`, `insert()`, `delete()`. Однако, если в случае однострочного текстового поля было достаточно указать один индекс элемента при вставке или удалении, то в случае многострочного надо указывать два – номер строки и номер символа в этой строке (другими словами, номер столбца). При этом нумерация строк начинается с единицы, а столбцов – с нуля.

```
from tkinter import *

def insertText():
    s = "Hello World"
    text.insert(1.0, s)

def getText():
    s = text.get(1.0, END)
    label['text'] = s
```

```

def deleteText():
    text.delete(1.0, END)

root = Tk()

text = Text(width=25, height=5)
text.pack()

frame = Frame()
frame.pack()

b_insert = Button(frame, text="Вставить", command=insertText)
b_insert.pack(side=LEFT)

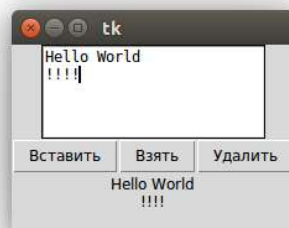
b_get = Button(frame, text="Взять", command=getText)
b_get.pack(side=LEFT)

b_delete = Button(frame, text="Удалить", command=deleteText)
b_delete.pack(side=LEFT)

label = Label()
label.pack()

root.mainloop()

```



Методы `get()` и `delete()` могут принимать не два, а один аргумент. В таком случае будет обрабатываться только один символ в указанной позиции.

Теги

Особенностью текстового поля библиотеки Tk является возможность форматировать текст в нем, то есть придавать его разным частям разное оформление. Делается это с помощью методов `tag_add()` и `tag_config()`. Первый добавляет тег, при этом надо указать его произвольное имя и отрезок текста, к которому он будет применяться. Метод `tag_config()` настраивает тегу стили оформления.

```

from tkinter import *
root = Tk()

text = Text(width=50, height=10)
text.pack()
text.insert(1.0, "Hello world!\nline two")

text.tag_add('title', 1.0, '1.end')
text.tag_config('title', font=("Verdana", 24, 'bold'), justify=CENTER)

root.mainloop()

```



Вставка виджетов в текстовое поле

В Text можно вставлять другие виджеты помощью метода `window_create()`. Потребность в этом не велика, однако может быть интересна с объектами типа Canvas. Данный класс будет изучен позже. В примере ниже вставляется метка в текущую (INSERT) позицию курсора.

```
from tkinter import *

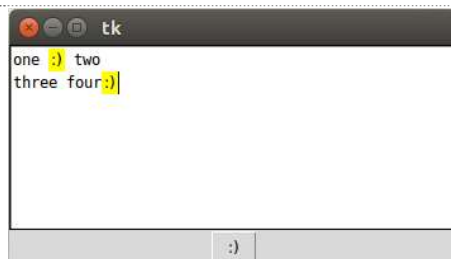
def smile():
    label = Label(text=":)", bg="yellow")
    text.window_create(INSERT, window=label)

root = Tk()

text = Text(width=50, height=10)
text.pack()

button = Button(text=":)", command=smile)
button.pack()

root.mainloop()
```



Размещение метки в функции позволяет каждый раз при вызове функции создавать новую метку. Иначе, если бы метка была в основной ветке программы, предыдущая исчезала бы.

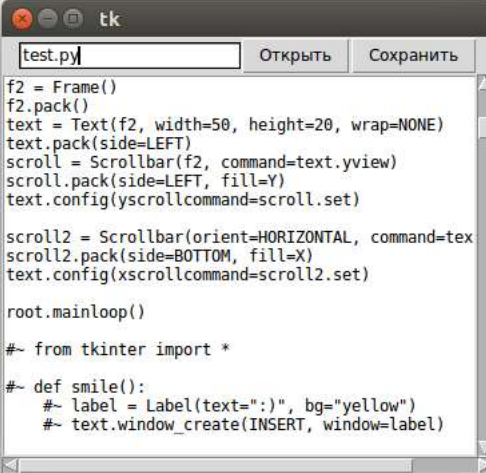
Практическая работа

Напишите программу, состоящую из однострочного и многострочного текстовых полей и двух кнопок "Открыть" и "Сохранить". При клике на первую должен открываться на чтение файл, чье имя указано в поле класса Entry, а содержимое файла должно загружаться в поле типа Text.

При клике на вторую кнопку текст, введенный пользователем в экземпляр Text, должен сохраняться в файле под именем, которое пользователь указал в однострочном текстовом поле.

Файлы будут читаться и записываться в том же каталоге, что и файл скрипта, если указывать имена файлов без адреса.

Для выполнения практической работы вам понадобится функция `open()` языка Python и методы файловых объектов чтения и записи. Освежить знания о них можно [здесь](https://younglinux.info/python/files) (<https://younglinux.info/python/files>).



```
f2 = Frame()
f2.pack()
text = Text(f2, width=50, height=20, wrap=NONE)
text.pack(side=LEFT)
scroll = Scrollbar(f2, command=text.yview)
scroll.pack(side=LEFT, fill=Y)
text.config(yscrollcommand=scroll.set)

scroll2 = Scrollbar(orient=HORIZONTAL, command=text.xview)
scroll2.pack(side=BOTTOM, fill=X)
text.config(xscrollcommand=scroll2.set)

root.mainloop()

#~ from tkinter import *

#~ def smile():
#~     label = Label(text=":", bg="yellow")
#~     text.window_create(INSERT, window=label)
```