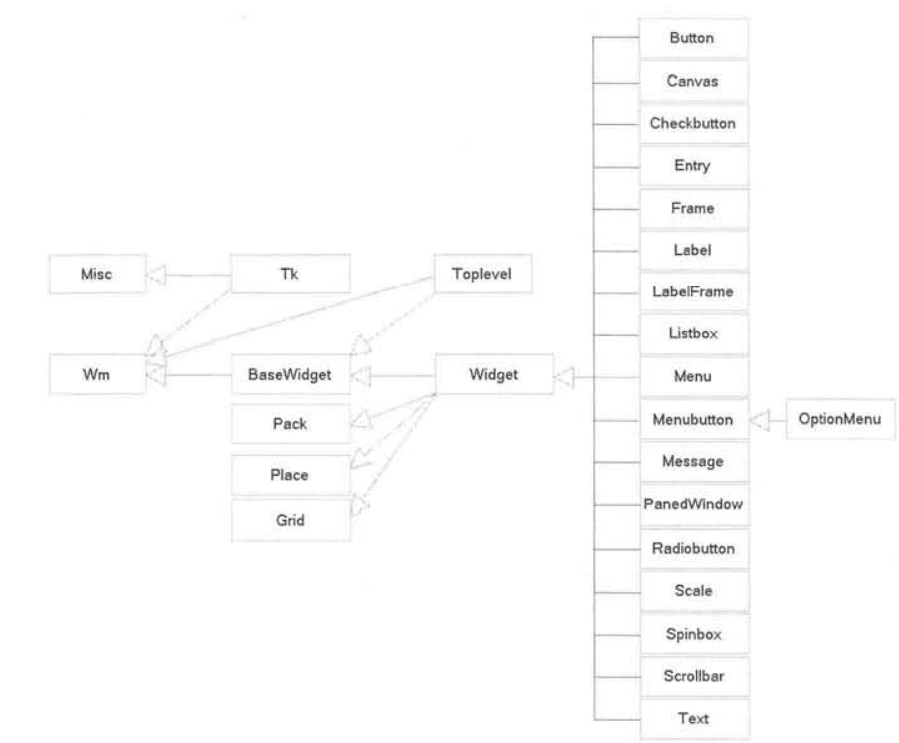
apt install python3-pip -y

apt search python3-tk

apt install python3-tk

# Tkinter的GUI组件之间的继承关系



# 各GUI组件的功能



# Label

from tkinter import \*

root = Tk()

root.title('windows title')

w = Label(root, text="hello Tkinter!")

w.pack()

root.mainloop()

# Button

from tkinter import \*

class Application(Frame):

def \_\_init\_\_(self, master=None):

Frame.\_\_init\_\_(self, master)

self.pack()

self.initWidgets()

def initWidgets(self):

w = Label(self)

bm = PhotoImage(file='serial.png')

w.x = bm

w['image'] = bm

w.pack()

btn = Button(self, text="Sure")

btn['background'] = 'yellow'

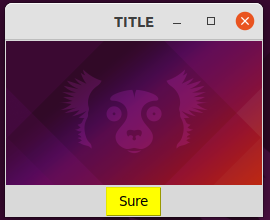
btn.pack()

app = Application()

print(type(app.master))

app.master.title('TITLE')

app.mainloop()



<class ‘tkinter.Tk’>

# 布局管理器

## Pack布局管理器

from tkinter import \*

root = Tk()

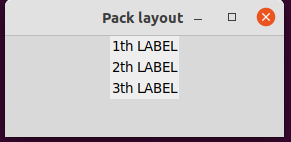
root.title('Pack layout')

for i in range(3):

lbl = Label(root, text="%dth LABEL" % (i+1), bg='#eeeeee')

lbl.pack()

root.mainloop()



from tkinter import \*

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

frame1 = Frame(self.master)

frame1.pack(side=LEFT,fill=BOTH,expan=YES)

Button(frame1, text='1st').pack(side=TOP,fill=X,expan=YES)

Button(frame1, text='2nd').pack(side=TOP,fill=X,expan=YES)

Button(frame1, text='3rd').pack(side=TOP,fill=X,expan=YES)

#从上到下

#从右到左

frame2 = Frame(self.master)

frame2.pack(side=LEFT,padx=10,expan=YES)

Button(frame2, text='1ST').pack(side=RIGHT,fill=Y,expan=YES)

Button(frame2, text='2ND').pack(side=RIGHT,fill=Y,expan=YES)

Button(frame2, text='3RD').pack(side=RIGHT,fill=Y,expan=YES)

#从下到上

frame3= Frame(self.master)

frame3.pack(side=RIGHT,padx=10,expan=YES)

Button(frame3, text='11').pack(side=BOTTOM,fill=Y,expan=YES)

Button(frame3, text='22').pack(side=BOTTOM,fill=Y,expan=YES)

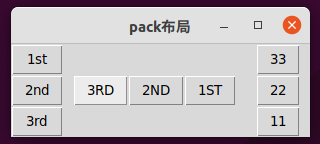
Button(frame3, text='33').pack(side=BOTTOM,fill=Y,expan=YES)

root = Tk()

root.title("pack布局")

display=App(root)

root.mainloop()



## grid布局管理器

from tkinter import \*

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

e = Entry(relief=SUNKEN, font=('Courier New', 24), width=25)

e.pack(side=TOP, pady=10)

p = Frame(self.master)

p.pack(side=TOP)

names=("0","1","2","3","4","5","6","7","8","9","+","-","\*","/",".","=" )

for i in range(len(names)):

b = Button(p, text=names[i], font=('Verdana', 20), width=6)

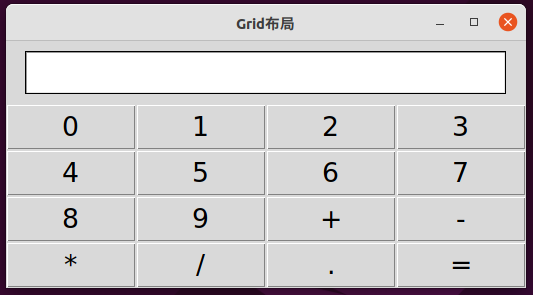
b.grid(row=i//4, column=i%4)

root=Tk()

root.title("Grid布局")

App(root)

root.mainloop()



## Place布局管理器 – 绝对布局

from tkinter import \*

import random

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

books = ('python', 'swift', 'Kotlin', '', 'java2', 'Ruby')

for i in range(len(books)):

ct = [random.randrange(256) for x in range(3)]

grayness = int(round(0.299\*ct[0] + 0.587\*ct[1] + 0.114\*ct[2]))

bg\_color = "#%02x%02x%02x" % tuple(ct)

lbl = Label(root, text=books[i],

fg='White' if grayness<125 else 'Black',

bg=bg\_color)

lbl.place(x=20, y=36+i\*36, width=180, height=30)

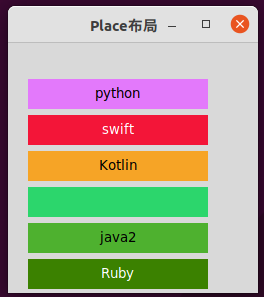
root=Tk()

root.title("Place布局")

root.geometry("250x250+30+30")

App(root)

root.mainloop()



# 事件处理

## 简单的事件处理

缺陷：不能区分事件的类型

from tkinter import \*

import random

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

self.label=Label(self.master, width=30)

self.label['font'] = ('Courier', 20)

self.label['bg'] = 'white'

self.label.pack()

btn = Button(self.master, text="hit me", command=self.change)

btn.pack()

def change(self):

self.label['text'] = 'welcome to study python'

ct = [random.randrange(256) for x in range(3)]

grayness = int(round(0.299\*ct[0] + 0.587\*ct[1] + 0.114\*ct[2]))

bg\_color = "#%02x%02x%02x" % tuple(ct)

self.label['bg'] = bg\_color

self.label['fg'] = 'White' if grayness<125 else 'Black'

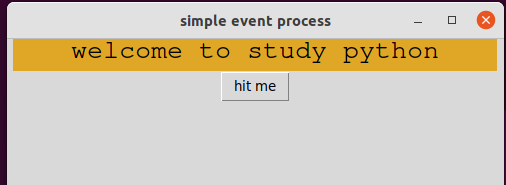
root=Tk()

root.title("simple event process")

root.geometry("250x250+30+30")

App(root)

root.mainloop()



## button事件绑定

from tkinter import \*

import random

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

self.label=Label(self.master, width=30, bg='white', font=('times', 20))

#self.label['font'] = ('Courier', 20)

#self.label['bg'] = 'white'

self.label.pack()

#btn = Button(self.master, text="hit me", command=self.change)

btn = Button(self.master, text="单击或双击我")

btn.pack(fill=BOTH, expand=YES)

btn.bind('<Button-1>', self.one)

btn.bind('<Double-1>', self.double)

def one(self, event):

self.label['text'] = "左键单击:%s" % event.widget['text']

def double(self, event):

print("左键双击，退出程序:", event.widget['text'])

import sys;sys.exit()

root=Tk()

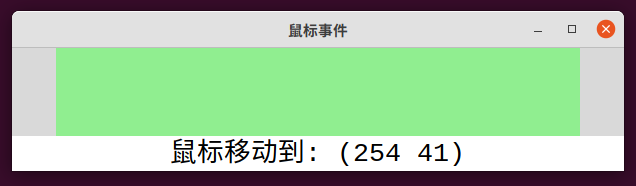
root.title("simple bind")

App(root)

root.mainloop()



## mouse事件绑定



from tkinter import \*

import random

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

lbl = Label(self.master, width=40, height=3)

lbl.config(bg='lightgreen', font=('Times', 20))

lbl.bind('<Motion>', self.motion)

lbl.bind('<B1-Motion>', self.press\_motion)

lbl.pack()

self.lbl = Label(self.master, width=38, height=1)

self.lbl.config(bg='white', font=('Courier New', 20))

self.lbl.pack()

def motion(self, event):

self.lbl['text'] = "鼠标移动到: (%s %s)" % (event.x, event.y)

def press\_motion(self, event):

self.lbl['text'] = "按住鼠标的位置为: (%s %s)" % (event.x, event.y)

root=Tk()

root.title("鼠标事件")

App(root)

root.mainloop()

## button事件绑定\_计算器

root@pc:/home/test# cat test.py

from tkinter import \*

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

self.expr = None

def initWidgets(self):

#self.e = Entry(relief=SUNKEN, font=('Courier New', 24), width=25)

self.e = Label(relief=SUNKEN, font=('Courier New', 24), width=25, bg='white', anchor=E)

self.e.pack(side=TOP, pady=10)

p = Frame(self.master)

p.pack(side=TOP)

names=("0","1","2","3","4","5","6","7","8","9","+","-","\*","/",".","=" )

for i in range(len(names)):

b = Button(p, text=names[i], font=('Verdana', 20), width=6)

b.grid(row=i//4, column=i%4)

b.bind('<Button-1>', self.click)

if b['text'] == '=': b.bind('<Double-1>', self.clean)

def click(self, event):

if event.widget['text'] == '=':

if self.expr not in ("0","1","2","3","4","5","6","7","8","9"):

print("invalid input: ", self.expr)

else:

print("do\_calc: ", self.e['text'], "=")

self.e['text'] = str(eval(self.e['text']))

else:

self.e['text'] = self.e['text'] + event.widget['text']

self.expr = event.widget['text']

def clean(self, event):

self.expr = None

self.e['text'] = ''

root=Tk()

root.title("Grid布局")

App(root)

root.mainloop()



# Tkinter常用组件

## 使用ttk组件

前面程序都是直接使用tkinter模块下的GUI组件，特别丑

Tkinter后来引入了一个ttk组件作为补充（主要就是简单包装，美化一下）

使用Combobox取代了Listbox，且新增LabeledScale, Notebook, progressbar, Treeview等

## \*Variable类

GUI组件和变量进行绑定

该类包含几个子类:

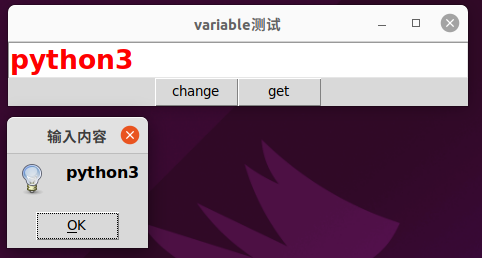
StringVar

IntVar

DoubleVar

BooleanVar

对于Variable类而言，保存使用get方法，设置使用set方法



from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

self.x = StringVar()

ttk.Entry(self.master, textvariable=self.x,

width=24, font=('StSong', 20, 'bold'), foreground='red').pack(fill=BOTH, expand=YES)

f = Frame(self.master)

f.pack()

ttk.Button(f, text='change', command=self.change).pack(side=LEFT)

ttk.Button(f, text='get', command=self.get).pack(side=LEFT)

def change(self):

books=('python3', 'kotlin', 'swift')

import random

self.x.set(books[random.randint(0,2)])

def get(self):

from tkinter import messagebox

messagebox.showinfo(title='输入内容', message=self.x.get())

root=Tk()

root.title("variable测试")

App(root)

root.mainloop()

## compound选项

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

bm = PhotoImage('serial.png')

self.lbl = ttk.Label(self.master, text='crazy books', image=bm, font=('StSong', 20, 'bold'), foreground='red')

self.lbl.bm = bm

self.lbl['compound'] = None

self.lbl.pack()

f = ttk.Frame(self.master)

f.pack(fill=BOTH, expand=YES)

compounds = ('None', 'LEFT', 'RIGHT', 'TOP', 'BOTTOM', 'CENTER')

self.var= StringVar()

self.var.set('None')

for val in compounds:

rb = Radiobutton(f, text=val, padx=20, variable=self.var, command=self.change\_compound, value=val).pack(side=LEFT, anchor=CENTER)

def change\_compound(self):

self.lbl['compound'] = self.var.get().lower()

print(self.var.get().lower())

root=Tk()

root.title("")

App(root)

root.mainloop()



## Entry和Text

root@pc:/home/zcw# cat test.py

from tkinter import \*

from tkinter import ttk

from tkinter import messagebox

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

self.entry = ttk.Entry(self.master, width=44, font=('StSong', 14), foreground='green')

self.entry.pack(fill=BOTH, expan=YES)

self.text = Text(self.master, width=44, height=4, font=('StSong', 14), foreground='gray')

self.text.pack(fill=BOTH, expand=YES)

f = Frame(self.master)

f.pack()

ttk.Button(f, text='begin insert', command=self.insert\_start).pack(side=LEFT)

ttk.Button(f, text='edit insert', command=self.insert\_edit).pack(side=LEFT)

ttk.Button(f, text='end insert', command=self.insert\_end).pack(side=LEFT)

ttk.Button(f, text='obtain entry', command=self.get\_entry).pack(side=LEFT)

ttk.Button(f, text='obtain text', command=self.get\_text).pack(side=LEFT)

def insert\_start(self):

self.entry.insert(0, 'swift')

self.text.insert(0.0, 'swift')

def insert\_edit(self):

self.entry.insert(INSERT, 'python3')

self.text.insert(INSERT, 'python3')

def insert\_end(self):

self.entry.insert(END, 'java2')

self.text.insert(END, 'java2')

def get\_entry(self):

messagebox.showinfo(title='输入内容', message=self.entry.get())

def get\_text(self):

messagebox.showinfo(title='输入内容', message=self.text.get(0.0, END))

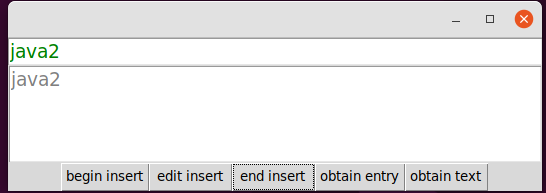
root=Tk()

root.title("")

App(root)

root.mainloop()

root@pc:/home/zcw#



## Scrollbar

root@pc:/home/zcw# cat test.py

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

text1 = Text(self.master, height=27, width=32)

text1.pack(side=LEFT, fill=BOTH, expand=YES)

bm1 = PhotoImage(file='java.png')

text1.bm = bm1

text1.insert(END, '\n')

text1.image\_create(END, image=bm1)

text2 = Text(self.master, height=33, width=50)

text2.pack(side=LEFT, fill=BOTH, expand=YES)

bm2 = PhotoImage(file='serial.png')

text2.bm = bm2

self.text = text2

#将垂直滚动条和text2关联

scroll = Scrollbar(self.master, command=text2.yview)

scroll.pack(side=RIGHT, fill=Y)

text2.configure(yscrollcommand=scroll.set)

text2.tag\_configure('title', font=('楷体', 20, 'bold'), foreground='red', justify=CENTER, spacing3=20)

text2.tag\_configure('detail', font=('微软雅黑', 11, 'bold'), foreground='darkgray', justify=CENTER, spacing2=10, spacing3=15)

text2.insert(END, '\n')

text2.insert(END, 'crazy java lessons\n', 'title')

details = ('窗前明月光'+'\n', '疑是地上霜'+'\n', '举头望明月'+'\n', '低头思故乡'+'\n')

for i in details:

text2.image\_create(END, image=bm2)

text2.insert(END, i, 'detail')

url = ['https://www.baidu.com', 'https://www.sina.com.cn']

name = ['baidu', 'sina']

m = 0

for i in name:

text2.tag\_configure(m, font=('微软雅黑', 13, 'bold'), underline=True, foreground='blue')

text2.tag\_bind(m, '<Enter>', self.show\_arrow\_cursor)

text2.tag\_bind(m, '<Leave>', self.show\_common\_cursor)

text2.tag\_bind(m, '<Button-1>', self.handlerAdapter(self.click, x=url[m]))

text2.insert(END, i+'\n', m)

m += 1

def show\_arrow\_cursor(self, event):

self.text.config(cursor='arrow')

def show\_common\_cursor(self, event):

self.text.config(cursor='xterm')

def click(self, event, x):

import webbrowser

print("open: %s" % x)

webbrowser.open(x)

def handlerAdapter(self, fun, \*\*kwds):

return lambda event,fun=fun,kwds=kwds: fun(event, \*\*kwds)

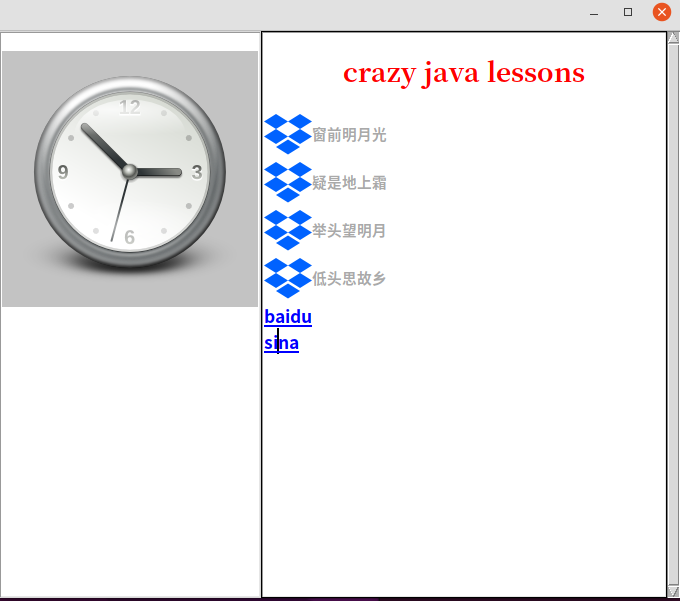
root=Tk()

root.title("")

App(root)

root.mainloop()

root@pc:/home/zcw#



## Radiobutton

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

ttk.Label(self.master, text='choose your favorite books:').pack(fill=BOTH, expan=YES)

self.intVar = IntVar()

books = ('crazy kotlin', 'crazy python3', 'crazy swift', 'crazy java2')

i = 1

for e in books:

ttk.Radiobutton(self.master, text=e, variabl=self.intVar, command=self.change, value=i).pack(anchor=W)

i += 1

self.intVar.set(2)

def change(self):

from tkinter import messagebox

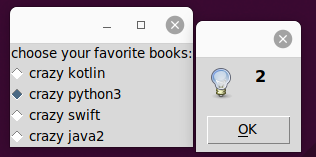
messagebox.showinfo(title=None, message=self.intVar.get())

root=Tk()

root.title("")

App(root)

root.mainloop()



## 带图像的Radiobutton

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

ttk.Label(self.master, text='choose your favorite kind:').pack(fill=BOTH, expan=YES)

self.intVar = IntVar()

imgs= ('z.png', 'p.png', 't.png')

names = ('虫族', '神族', '人族')

for i in range(3):

bm = PhotoImage(file='images/' + imgs[i])

r = ttk.Radiobutton(self.master, image=bm, text=names[i], compound=RIGHT, variabl=self.intVar, command=self.change, value=i+1)

r.bm = bm

r.pack(anchor=W)

self.intVar.set(2)

def change(self):

#from tkinter import messagebox

#messagebox.showinfo(title=None, message=self.intVar.get())

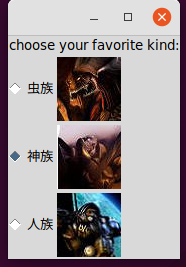
pass

root=Tk()

root.title("")

App(root)

root.mainloop()



## Checkbutton – 复选框

from tkinter import \*

from tkinter import ttk

from tkinter import messagebox

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

ttk.Label(self.master, text='choose your favorite people:').pack(fill=BOTH, expan=YES)

self.chars = []

names = ('sunwukong', 'zhubajie', 'tangseng', 'niumowang')

for i in range(4):

intVar = IntVar()

self.chars.append(intVar)

ttk.Checkbutton(self.master, text=names[i], variable=intVar, command=self.change).pack(anchor=W)

#print('------------------------------------------')

ttk.Label(self.master, text='choose your favorite book:').pack(fill=BOTH, expan=YES)

self.books = []

books = ('crazy python', 'crazy kotlin', 'crazy swift', 'crazy java2')

ss = ('python', 'kotlin', 'swift', 'java2')

for i in range(4):

strVar = StringVar()

self.books.append(strVar)

ttk.Checkbutton(self.master, text=books[i], variable=strVar, onvalue=ss[i], offvalue='无', command=self.books\_change).pack(anchor=W)

def change(self):

new\_li = [str(e.get()) for e in self.chars]

st = ','.join(new\_li)

messagebox.showinfo(title=None, message=st)

def books\_change(self):

new\_li = [str(e.get()) for e in self.books]

st = ','.join(new\_li)

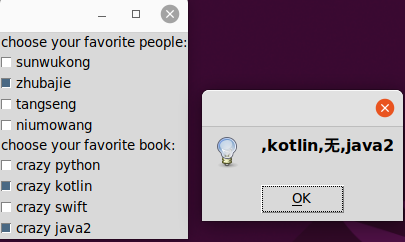
messagebox.showinfo(title=None, message=st)

root=Tk()

root.title("")

App(root)

root.mainloop()



## Listbox

Listbox下拉框

Combobox组合框 = Listbox + 文本框

Listbox除了有insert方法，还有如下方法：

selection\_set(self, first,last=None):

selection\_clear(self, first,last=None):

delete(self, first,last=None):

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

tf = Frame(self.master)

tf.pack(fill=Y, expand=YES)

self.lb = Listbox(tf)

self.lb.pack(side=LEFT, fill=Y, expand=YES)

for item in ['python3', 'kotlin', 'swift', 'ruby']:

self.lb.insert(END, item) #一次插入一行

self.lb.insert(ANCHOR, 'python3', 'kotlin', 'swift', 'ruby') #一次插入多行

#下拉列表框和垂直滚动条关联

scroll = Scrollbar(tf, command=self.lb.yview)

scroll.pack(side=RIGHT, fill=Y)

self.lb.configure(yscrollcommand=scroll.set)

f = Frame(self.master)

f.pack()

Label(f, text='选择模式:').pack(side=LEFT)

modes = ('multiple', 'browse', 'single', 'extended')

self.strVar = StringVar()

for m in modes:

ttk.Radiobutton(f, text=m, value=m, variable=self.strVar, command=self.choose\_mode).pack(side=LEFT)

self.strVar.set('browse')

def choose\_mode(self):

print(self.strVar.get())

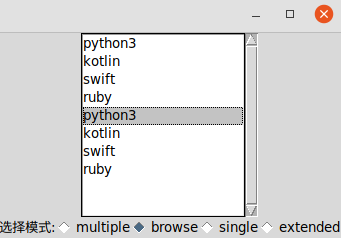
self.lb['selectmode'] = self.strVar.get()

root=Tk()

root.title("")

App(root)

root.mainloop()



## Listbox的listvariable选项与变量绑定

Listbox下拉框

Combobox组合框 = Listbox + 文本框

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

tf = Frame(self.master)

tf.pack(fill=Y, expand=YES)

self.v = StringVar()

self.lb = Listbox(tf, listvariable=self.v)

self.lb.pack(side=LEFT, fill=Y, expand=YES)

for i in range(20):

self.lb.insert(END, str(i))

scroll = Scrollbar(tf, command=self.lb.yview)

scroll.pack(side=RIGHT, fill=Y)

self.lb.configure(yscrollcommand=scroll.set)

f = Frame(self.master)

f.pack()

Button(f, text="选中10项", command=self.select).pack(side=LEFT)

Button(f, text="清除选中3项", command=self.clear\_select).pack(side=LEFT)

Button(f, text="删除3项", command=self.delete).pack(side=LEFT)

Button(f, text="绑定变量", command=self.var\_select).pack(side=LEFT)

def select(self):

self.lb.selection\_set(0,9)

def clear\_select(self):

self.lb.selection\_clear(1,3)

def delete(self):

self.lb.delete(5, 8)

def var\_select(self):

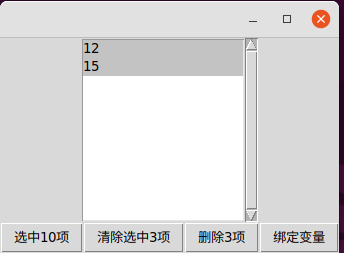
self.v.set(('12', '15'))

root=Tk()

root.title("")

App(root)

root.mainloop()



## Listbox事件绑定-bind

Listbox不支持command绑定方式，支持bind绑定

command通常和控件绑定（例如按钮），函数形如: def xxx(self)

bind和控件事件绑定，函数形如: def xxx(self, event)

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

tf = Frame(self.master)

tf.pack(fill=Y, expand=YES)

self.lb = Listbox(tf)

self.lb.pack(side=LEFT, fill=Y, expand=YES)

for i in range(20):

self.lb.insert(END, str(i))

scroll = Scrollbar(tf, command=self.lb.yview)

scroll.pack(side=RIGHT, fill=Y)

self.lb.configure(yscrollcommand=scroll.set)

self.lb.bind("<Double-1>", self.click)

def click(self, event):

from tkinter import messagebox

messagebox.showinfo(title=None, message=str(self.lb.curselection()))

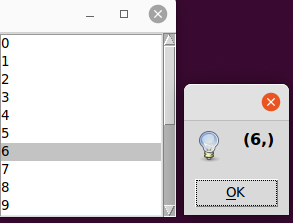
root=Tk()

root.title("")

#root.iconbitmap('images/fklogo.ico')

App(root)

root.mainloop()



## Combobox

Listbox下拉框

Combobox组合框 = Listbox + 文本框

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

self.strVar = StringVar()

self.cb = ttk.Combobox(self.master, textvariable=self.strVar, postcommand=self.choose)

self.cb.pack(side=TOP)

self.cb['values'] = ['python','ruby','kotlin','swift']

f = Frame(self.master)

f.pack()

self.isreadonly = IntVar()

#只改变只读状态值，除了按钮会被灰选，什么都不会发生

Checkbutton(f, text='是否只读:', variable=self.isreadonly, command=self.change).pack(side=LEFT)

Button(f, text='绑定变量设置', command=self.setvalue).pack(side=LEFT)

def choose(self):

from tkinter import messagebox

messagebox.showinfo(title=None, message=str(self.cb.get()))

def change(self):

self.cb['state'] = 'readonly' if self.isreadonly.get() else 'enable'

def setvalue(self):

self.strVar.set('I love python3')

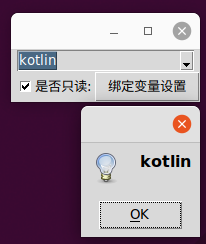
root=Tk()

root.title("")

#root.iconbitmap('images/fklogo.ico')

App(root)

root.mainloop()



## Spinbox-选钮控件

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

ttk.Label(self.master, text='min-max-step').pack()

sb1 = Spinbox(self.master, from\_=20, to=100, increment=5)

sb1.pack(fill=X, expand=YES)

ttk.Label(self.master, text='指定values').pack()

self.sb2 = Spinbox(self.master, values=('111', '222', '333', '444'), command=self.press)

self.sb2.pack(fill=X, expand=YES)

ttk.Label(self.master, text='绑定变量').pack()

self.intVar = IntVar()

sb3 = Spinbox(self.master, values=list(range(20, 100, 4)), textvariable=self.intVar, command=self.press)

sb3.pack(fill=X, expand=YES)

self.intVar.set(33)

def press(self):

print(self.sb2.get())

root=Tk()

root.title("")

App(root)

root.mainloop()



## Scale – 滑动条

#将scale和两个按钮关联起来

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

self.scale = Scale(self.master, from\_=-100, to=100, resolution=5, label='demo',length=400,

width=30, troughcolor='lightblue', sliderlength=20, sliderrelief=SUNKEN, showvalue=YES, orient=HORIZONTAL)

self.scale.pack()

f = Frame(self.master)

f.pack(fill=X, expand=YES, padx=10)

Label(f, text='是否显示值:').pack(side=LEFT)

i=0

self.showVar =IntVar()

self.showVar.set(1)

for s in ('不显示', '显示'):

Radiobutton(f, text=s, value=i, variabl=self.showVar, command=self.switch\_show).pack(side=LEFT)

i+=1

f = Frame(self.master)

f.pack(fill=X, expand=YES, padx=10)

Label(f, text='方向:').pack(side=LEFT)

i=0

self.orientVar = IntVar()

self.orientVar.set(0)

for s in ('水平', '垂直'):

Radiobutton(f, text=s, value=i, variable=self.orientVar, command=self.switch\_orient).pack(side=LEFT)

i+=1

def switch\_show(self):

self.scale['showvalue'] = self.showVar.get()

def switch\_orient(self):

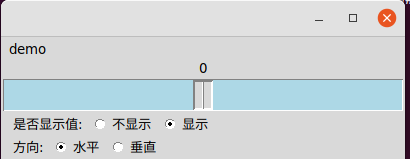
self.scale['orient'] = VERTICAL if self.orientVar.get() else HORIZONTAL

root=Tk()

root.title("")

App(root)

root.mainloop()



## scale绑定变量

#滑动条直接绑定变量，command绑定的事件方法处理方法

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

self.doubleVar=DoubleVar()

self.scale = Scale(self.master, from\_=-100, to=100, resolution=5, label='demo',length=400,

width=30, digits=10, command=self.change, variable=self.doubleVar, orient=HORIZONTAL)

self.scale.pack()

self.scale.set(20)

def change(self, value):

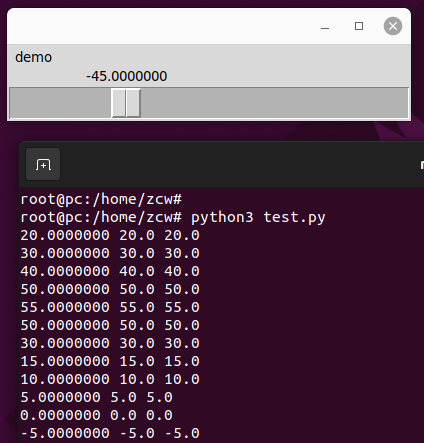
print(value, self.scale.get(), self.doubleVar.get())

root=Tk()

root.title("")

App(root)

root.mainloop()



## LabeledScale – 滑动标签

#标签位置随着滑块同步变化

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

self.scale = ttk.LabeledScale(self.master, from\_=-100, to=100, compound=BOTTOM)

self.scale.pack(fill=X, expand=YES)

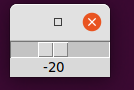
self.scale.value = -20

root=Tk()

root.title("")

App(root)

root.mainloop()



## Labelframe

说到底，它还是Frame

但该Frame把其他控件当标签处理

### 例1

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

lf = ttk.Labelframe(self.master, text='please choose books', padding=20)

lf.pack(fill=BOTH, expand=YES, padx=10, pady=10)

books=['111', '222', '333', '444']

i = 0

self.intVar = IntVar()

for item in books:

Radiobutton(lf, text=item, value=i, variable=self.intVar).pack(side=LEFT)

i+=1

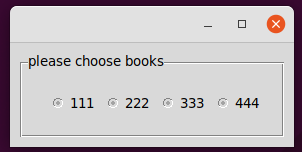
#self.intVar.set(2)

root=Tk()

root.title("")

App(root)

root.mainloop()



### 例2 “标签”是图片，且图片位置可以变化

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

self.lf = ttk.Labelframe(self.master, padding=20)

self.lf.pack(fill=BOTH, expand=YES, padx=10, pady=10)

bm =PhotoImage(file='images/z.png')

lbl = Label(self.lf, image=bm)

lbl.bm = bm

self.lf['labelwidget'] = lbl

#总共有12个常量值得，代表图片的方位

#这些值都是由python的ttk组件规定的，不能是任意值

self.books=['e', 's', 'w', 'n', 'es']

i = 0

self.intVar = IntVar()

for item in self.books:

Radiobutton(self.lf, text=item, value=i, command=self.change, variable=self.intVar).pack(side=LEFT)

i+=1

self.intVar.set(2)

def change(self):

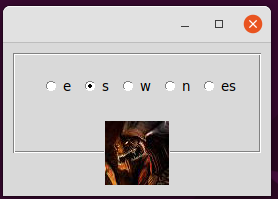
self.lf['labelanchor'] = self.books[self.intVar.get()]

root=Tk()

root.title("")

App(root)

root.mainloop()



## Panedwindow

paned：窗格，拼做的

Panedwindow： 推拉窗

Panedwindow组件是一个管理窗口布局的容器，它允许添加多个子组件（不需要使用pack，grid或者place布局），并为每个子组件划分一个区域，用户可用鼠标移动各区域的分隔线来改变各子组件的大小。

root@pc:/home/zcw# cat test.py

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

style = ttk.Style()

style.configure('fkit.TPanedwindow', background='darkgray', relief=RAISED)

pw = ttk.Panedwindow(self.master, orient=VERTICAL, style='fkit.TPanedwindow')

pw.pack(fill=BOTH, expand=1)

first = ttk.Label(pw, text='first label')

pw.add(first)

#点击按钮，则Panedwindow控件删除按钮

second = ttk.Button(pw, text='second button', command=lambda:pw.remove(second))

pw.add(second)

third = ttk.Entry(pw, width=30)

pw.add(third)

#第一个参数是插入位置，第二个参数是插入的控件

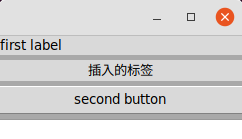
pw.insert(1, Label(pw, text='插入的标签'))

root=Tk()

root.title("")

App(root)

root.mainloop()



## Panedwindow嵌套

paned：窗格，拼做的

Panedwindow： 推拉窗

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

style = ttk.Style()

style.configure('fkit.TPanedwindow', background='darkgray', relief=RAISED)

pw1 = ttk.Panedwindow(self.master, orient=HORIZONTAL, style='fkit.TPanedwindow')

pw1.pack(fill=BOTH, expand=1)

pw2 = ttk.Panedwindow(pw1, orient=VERTICAL)

first = ttk.Label(pw1, text='first label', background='pink')

second = ttk.Label(pw1, text='second label', background='lightgreen')

third = ttk.Label(pw1, text='third label', background='lightblue')

pw1.add(first)

pw1.add(pw2)

pw2.add(second)

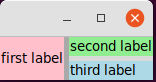
pw2.add(third)

root=Tk()

root.title("")

App(root)

root.mainloop()



## OptionMenu

from tkinter import \*

from tkinter import ttk

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

self.sv = StringVar()

self.om = ttk.OptionMenu(root, self.sv, '111', '222', '333', '444', '555', '666', command=self.print\_option)

self.om.pack()

lf = ttk.Labelframe(self.master, padding=20, text='请选择则菜单方向')

lf.pack(fill=BOTH, expand=YES, padx=10, pady=10)

self.directions = ['below', 'above', 'left', 'right', 'flush']

i = 0

self.intVar = IntVar()

for item in self.directions:

Radiobutton(lf, text=item, value=i, command=self.change, variable=self.intVar).pack(side=LEFT)

i += 1

self.intVar.set(2)

def print\_option(self, val):

#通过两种方法获得菜单选项的值

#self.sv.get()和val的值相同

print(self.sv.get(), val)

def change(self):

print(self.om['direction'])

#选项菜单弹出后出现的位置在锚点的左边，还是右边

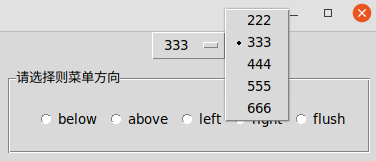
self.om['direction'] = self.directions[self.intVar.get()]

root=Tk()

root.title("")

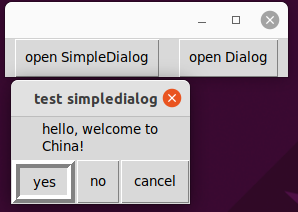
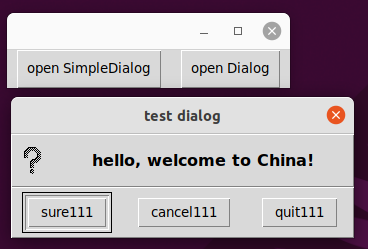
App(root)

root.mainloop()



# 对话框

## simpledialog和dialog

from tkinter import \*

from tkinter import ttk

from tkinter import simpledialog

from tkinter import dialog

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

self.msg = 'hello, welcome to China!'

ttk.Button(self.master, text="open SimpleDialog", command=self.open\_simpledialog).pack(side=LEFT, ipadx=5, ipady=5, padx=10)

ttk.Button(self.master, text="open Dialog", command=self.open\_dialog).pack(side=LEFT, ipadx=5, ipady=5, padx=10)

def open\_simpledialog(self):

# simpledialog右上角的x按钮对应3

d = simpledialog.SimpleDialog(self.master, title="test simpledialog", text=self.msg, buttons=['yes', 'no', 'cancel'], cancel=3, default=0) #缺省焦点是0

print(d.go()) #三个按钮分别对应0，1，2

def open\_dialog(self):

d = dialog.Dialog(self.master, {'title':'test dialog', 'text':self.msg, 'bitmap':'question', 'default':0, 'strings':('sure111', 'cancel111', 'quit111')})

print(d.num) #三个按钮分别对应0，1，2

root=Tk()

root.title("")

App(root)

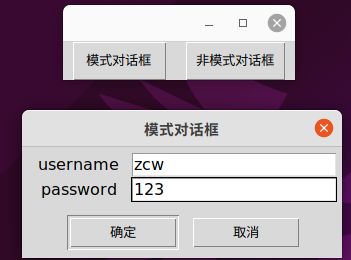
root.mainloop()

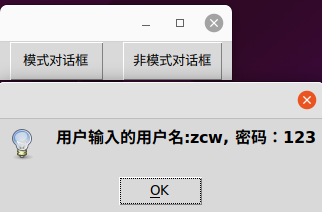
## 自定义模式和非模式对话框

模态对话框：如果不关闭，则不能和父窗口的控件交互

例如：打开模态对话框，则不能执行”非模态对话框”按钮

打开非模态对话框，仍能执行“模态对话框”按钮





from tkinter import \*

from tkinter import ttk

from tkinter import messagebox

class MyDialog(Toplevel):

def \_\_init\_\_(self, parent, title=None, modal=True):

Toplevel.\_\_init\_\_(self, parent)

self.transient(parent)

if title: self.title(title)

self.parent = parent

self.result = None

frame = Frame(self)

frame.pack(padx=5, pady=5)

self.initial\_focus = self.init\_widgets(frame)

if not self.initial\_focus:

self.initial\_focus = self

self.init\_buttons()

if modal: self.grab\_set()

self.protocol("WM\_DELETE\_WINDOW", self.cancel\_click)

self.geometry("+%d+%d" % (parent.winfo\_rootx()+50, parent.winfo\_rooty()+50))

print(self.initial\_focus)

self.initial\_focus.focus\_set()

self.wait\_window(self)

def init\_widgets(self, master):

Label(master, text='username', font=12,width=10).grid(row=1, column=0)

self.name\_entry = Entry(master, font=16)

self.name\_entry.grid(row=1, column=1)

Label(master, text='password', font=12,width=10).grid(row=2, column=0)

self.pass\_entry = Entry(master, font=16)

self.pass\_entry.grid(row=2, column=1)

def init\_buttons(self):

f = Frame(self)

w = Button(f, text='确定', width=10, command=self.ok\_click, defaul=ACTIVE)

w.pack(side=LEFT, padx=5, pady=5)

w = Button(f, text='取消', width=10, command=self.cancel\_click)

w.pack(side=LEFT, padx=5, pady=5)

self.bind("<Return>", self.ok\_click)

self.bind("<Escape>", self.cancel\_click)

f.pack()

def validate(self):

return True

def process\_input(self):

user\_name = self.name\_entry.get()

user\_pass = self.pass\_entry.get()

messagebox.showinfo(message='用户输入的用户名:%s, 密码：%s' % (user\_name, user\_pass))

def ok\_click(self, event=None):

print("sure")

if not self.validate():

self.initial\_focus.focus\_set()

return

self.withdraw()

self.update\_idletasks()

self.process\_input()

self.parent.focus\_set()

self.destroy()

def cancel\_click(self, event=None):

print('cancel')

self.parent.focus\_set()

self.destroy()

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

ttk.Button(self.master, text='模式对话框', command=self.open\_modal).pack(side=LEFT, ipadx=5, ipady=5, padx=10)

ttk.Button(self.master, text='非模式对话框', command=self.open\_none\_modal).pack(side=LEFT, ipadx=5, ipady=5, padx=10)

def open\_modal(self):

d = MyDialog(self.master, title='模式对话框')

def open\_none\_modal(self):

d = MyDialog(self.master, title='非模式对话框', modal=False)

root=Tk()

root.title("")

App(root)

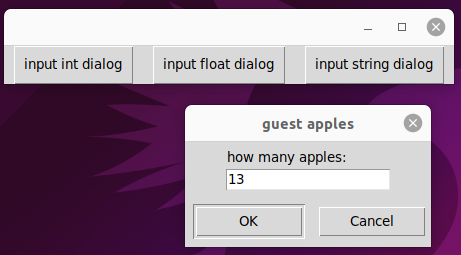
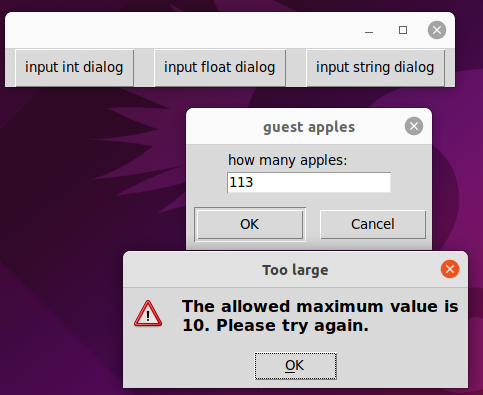
root.mainloop()

## simpledialog之输入对话框

simpledialog.askinteger

simpledialog.askfloat

simpledialog.askstring

from tkinter import \*

from tkinter import ttk

from tkinter import simpledialog

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

ttk.Button(self.master, text='input int dialog', command=self.open\_integer).pack(side=LEFT,ipadx=5,ipady=5,padx=10)

ttk.Button(self.master, text='input float dialog', command=self.open\_float).pack(side=LEFT,ipadx=5,ipady=5,padx=10)

ttk.Button(self.master, text='input string dialog', command=self.open\_string).pack(side=LEFT,ipadx=5,ipady=5,padx=10)

def open\_integer(self):

print(simpledialog.askinteger('guest apples', 'how many apples:', initialvalue=3, minvalue=1, maxvalue=10))

def open\_float(self):

print(simpledialog.askfloat('guest price', 'how much apple:', initialvalue=6.8, minvalue=1, maxvalue=100))

def open\_string(self):

print(simpledialog.askstring('guest name', 'what is your name:', initialvalue='zcw'))

root=Tk()

root.title("")

App(root)

root.mainloop()

## filedialog文件对话框

root@pc:/home/zcw# touch /home/test/1.txt

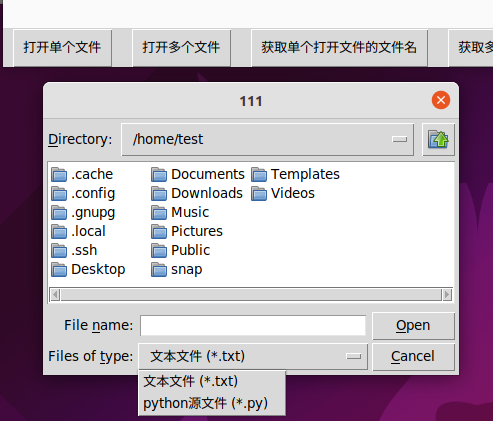
root@pc:/home/zcw# touch /home/test/2.txt

root@pc:/home/zcw# touch /home/test/3.txt

root@pc:/home/zcw# echo "111" > /home/test/1.txt

root@pc:/home/zcw# echo "222" > /home/test/2.txt

root@pc:/home/zcw# echo "33" > /home/test/3.txt



from tkinter import \*

from tkinter import ttk

from tkinter import filedialog

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

ttk.Button(self.master, text='打开单个文件', command=self.open\_file).pack(side=LEFT,ipadx=5,ipady=5,padx=10)

ttk.Button(self.master, text='打开多个文件', command=self.open\_files).pack(side=LEFT,ipadx=5,ipady=5,padx=10)

ttk.Button(self.master, text='获取单个打开文件的文件名', command=self.open\_filename).pack(side=LEFT,ipadx=5,ipady=5,padx=10)

ttk.Button(self.master, text='获取多个打开文件的文件名', command=self.open\_filenames).pack(side=LEFT,ipadx=5,ipady=5,padx=10)

ttk.Button(self.master, text='获取保存文件', command=self.save\_file).pack(side=LEFT,ipadx=5,ipady=5,padx=10)

ttk.Button(self.master, text='获取保存文件的文件名', command=self.save\_filename).pack(side=LEFT,ipadx=5,ipady=5,padx=10)

ttk.Button(self.master, text='打开路径', command=self.open\_dir).pack(side=LEFT,ipadx=5,ipady=5,padx=10)

def open\_file(self):

print(filedialog.askopenfile(title='111', filetypes=[("文本文件", '\*.txt'), ('python源文件', '\*.py')], initialdir='/home/test'))

def open\_files(self):

print(filedialog.askopenfiles(title='222', filetypes=[("文本文件", '\*.txt'), ('python源文件', '\*.py')], initialdir='/home/test'))

def open\_filename(self):

print(filedialog.askopenfilename(title='333', filetypes=[("文本文件", '\*.txt'), ('python源文件', '\*.py')], initialdir='/home/test'))

def open\_filenames(self):

print(filedialog.askopenfilenames(title='444', filetypes=[("文本文件", '\*.txt'), ('python源文件', '\*.py')], initialdir='/home/test'))

def save\_file(self):

print(filedialog.asksaveasfile(title='555', filetypes=[("文本文件", '\*.txt'), ('python源文件', '\*.py')], initialdir='/home/test'))

def save\_filename(self):

print(filedialog.asksaveasfilename(title='666', filetypes=[("文本文件", '\*.txt'), ('python源文件', '\*.py')], initialdir='/home/test'))

def open\_dir(self):

print(filedialog.askdirectory(title='777', initialdir='/home/test'))

root=Tk()

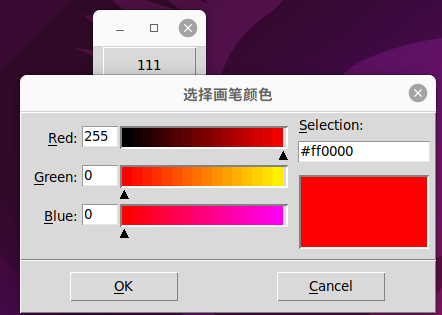
root.title("")

App(root)

root.mainloop()

## colorchooser颜色对话框

colorchooser.askcolor



from tkinter import \*

from tkinter import ttk

from tkinter import colorchooser

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

ttk.Button(self.master, text='111', command=self.choose\_color).pack(side=LEFT,ipadx=5,ipady=5,padx=10)

def choose\_color(self):

print(colorchooser.askcolor(parent=self.master, title='选择画笔颜色', color='red'))

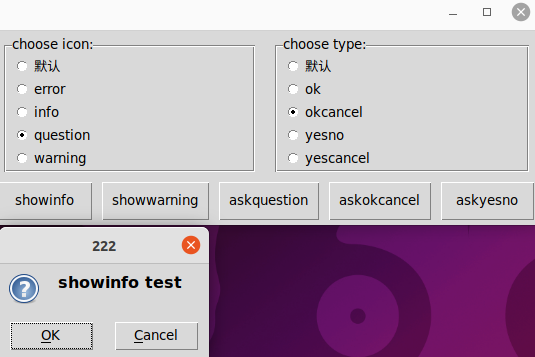
root=Tk()

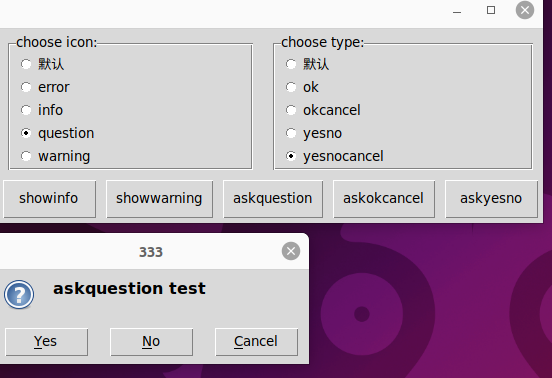
root.title("")

App(root)

root.mainloop()

## 消息对话框





from tkinter import \*

from tkinter import ttk

from tkinter import messagebox as msgbox

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

tf1 = Frame(self.master)

tf1.pack(fill=BOTH)

lf1 = ttk.Labelframe(tf1, text='choose icon:')

lf1.pack(side=LEFT, fill=BOTH, expand=YES, padx=10, pady=5)

lf2 = ttk.Labelframe(tf1, text='choose type:')

lf2.pack(side=LEFT, fill=BOTH, expand=YES, padx=10, pady=5)

i=0

self.iconVar=IntVar()

self.icons = [None, 'error', 'info', 'question', 'warning']

for item in self.icons:

Radiobutton(lf1, text=item if item is not None else '默认', value=i, variable=self.iconVar).pack(side=TOP, anchor=W)

i+=1

self.iconVar.set(1)

i=0

self.typeVar=IntVar()

self.types = [None,"ok","okcancel","yesno","yesnocancel"]

for item in self.types:

Radiobutton(lf2, text=item if item is not None else '默认', value=i, variable=self.typeVar).pack(side=TOP, anchor=W)

i+=1

self.typeVar.set(2)

print('--------------------------')

tf2 = Frame(self.master)

tf2.pack(fill=BOTH)

ttk.Button(tf2, text="showinfo", command=self.showinfo\_clicked).pack(side=LEFT, fill=X, ipadx=5,ipady=5,padx=5,pady=5)

ttk.Button(tf2, text="showwarning", command=self.showwarning\_clicked).pack(side=LEFT, fill=X, ipadx=5,ipady=5,padx=5,pady=5)

ttk.Button(tf2, text="askquestion", command=self.askquestion\_clicked).pack(side=LEFT, fill=X, ipadx=5,ipady=5,padx=5,pady=5)

ttk.Button(tf2, text="askokcancel", command=self.askokcancel\_clicked).pack(side=LEFT, fill=X, ipadx=5,ipady=5,padx=5,pady=5)

ttk.Button(tf2, text="askyesno", command=self.askyesno\_clicked).pack(side=LEFT, fill=X, ipadx=5,ipady=5,padx=5,pady=5)

def showinfo\_clicked(self):

print(msgbox.showinfo('111', 'showinfo test', icon=self.icons[self.iconVar.get()], type=self.types[self.typeVar.get()]))

def showwarning\_clicked(self):

print(msgbox.showwarning('222', 'showwarning test', icon=self.icons[self.iconVar.get()], type=self.types[self.typeVar.get()]))

def askquestion\_clicked(self):

print(msgbox.askquestion('333', 'askquestion test', icon=self.icons[self.iconVar.get()], type=self.types[self.typeVar.get()]))

def askokcancel\_clicked(self):

print(msgbox.askokcancel('444', 'askokcancel test', icon=self.icons[self.iconVar.get()], type=self.types[self.typeVar.get()]))

def askyesno\_clicked(self):

print(msgbox.askyesnocancel('555', 'askyesnocancel test', icon=self.icons[self.iconVar.get()], type=self.types[self.typeVar.get()]))

root=Tk()

root.title("")

App(root)

root.mainloop()

# 菜单

add\_command #添加菜单项

add\_checkbutton #添加复选框菜单项

add\_radiobutton #添加菜单选钮单项

add\_separator #添加分割条

label: 文本

command: 菜单绑定的事件处理方法

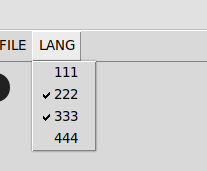
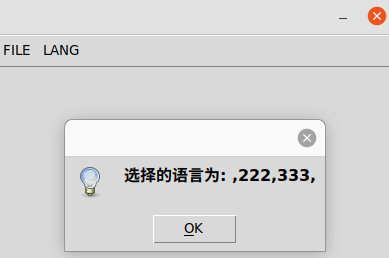
image: 菜单项的图标

compound: 菜单项中图标位于文字的哪个方位

## 窗口菜单

icon菜单

 单选按钮菜单

  复选按钮菜单

from tkinter import \*

from tkinter import ttk

from tkinter import messagebox as msgbox

class App:

def \_\_init\_\_(self, master):

self.master=master

self.init\_menu()

def init\_menu(self):

self.master.filenew\_icon = PhotoImage(file='images/z.png')

self.master.fileopen\_icon = PhotoImage(file='images/p.png')

menubar = Menu(self.master)

file\_menu = Menu(menubar, tearoff=0)

lang\_menu = Menu(menubar, tearoff=0)

sub\_menu = Menu(file\_menu, tearoff=0)

self.master['menu'] = menubar

menubar.add\_cascade(label='FILE', menu=file\_menu)

menubar.add\_cascade(label='LANG', menu=lang\_menu)

file\_menu.add\_command(label='新建', command=None, image=self.master.filenew\_icon, compound=LEFT)

file\_menu.add\_command(label='打开', command=None, image=self.master.fileopen\_icon, compound=LEFT)

file\_menu.add\_separator()

file\_menu.add\_cascade(label='选择性别', menu=sub\_menu)

self.genderVar = IntVar()

for i, item in enumerate(['male', 'female', 'secret']):

sub\_menu.add\_radiobutton(label=item, command=self.choose\_gender, variable=self.genderVar, value=i)

self.langVars = [StringVar(), StringVar(), StringVar(), StringVar()]

for i, item in enumerate(('111', '222', '333', '444')):

lang\_menu.add\_checkbutton(label=item, command=self.choose\_lang, onvalue=item, variable=self.langVars[i])

def choose\_gender(self):

msgbox.showinfo(message=('选择的性别为: %s' % self.genderVar.get()))

def choose\_lang(self):

li = [e.get() for e in self.langVars]

msgbox.showinfo(message=('选择的语言为: %s' % ','.join(li)))

root=Tk()

root.title("")

root.geometry('400x200')

#禁止改变窗口大小

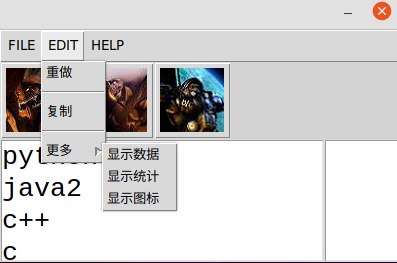
root.resizable(width=False, height=False)

App(root)

root.mainloop()

## 带工具栏的高级菜单

#左侧是列表框



from tkinter import \*

from tkinter import ttk

from tkinter import messagebox as msgbox

from collections import OrderedDict

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initwidgets()

def initwidgets(self):

self.init\_icons()

self.init\_menu()

self.init\_toolbar()

leftframe = ttk.Frame(self.master, width=40)

leftframe.pack(side=LEFT, fill=Y)

mainframe = ttk.Frame(self.master)

mainframe.pack(side=LEFT, fill=BOTH)

lb = Listbox(leftframe, font=('Courier New', 20))

lb.pack(fill=Y, expand=YES)

for i in ('python3', 'java2', 'c++', 'c', 'lua', 'go'):

lb.insert(END, i)

text = Text(mainframe, width=40, font=('Courier New', 16))

text.pack(side=LEFT, fill=BOTH)

scroll =ttk.Scrollbar(mainframe)

scroll.pack(side=LEFT, fill=Y)

scroll['command'] = text.yview

text.configure(yscrollcommand=scroll.set)

def init\_menu(self):

menus = ('FILE', 'EDIT', 'HELP')

items = (

OrderedDict([ ('新建', (self.master.filenew\_icon, None)),

('打开', (self.master.fileopen\_icon, None)),

('-1', (None, None)),

('退出', (self.master.signout\_icon, None))]),

OrderedDict([

('重做', (None, None)),

('-1', (None, None)),

('复制', (None, None)),

('-2', (None, None)),

('更多', OrderedDict([

('显示数据', (None,None)),

('显示统计', (None,None)),

('显示图标', (None,None))

]))

]),

OrderedDict([

('帮助主题', (None, None)),

('-1', (None, None)),

('关于', (None, None))

])

)

menubar = Menu(self.master)

self.master['menu'] = menubar

for i, m\_title in enumerate(menus):

m = Menu(menubar, tearoff=0)

menubar.add\_cascade(label=m\_title, menu=m)

tm = items[i]

print(type(tm))

for label in tm:

print(label)

#说明是二级菜单

if isinstance(tm[label], OrderedDict):

sm = Menu(m, tearoff=0)

m.add\_cascade(label=label, menu=sm)

sub\_dict = tm[label]

for sub\_label in sub\_dict:

if sub\_label.startswith('-'):

sm.add\_separator()

else:

sm.add\_command(label=sub\_label, image=sub\_dict[sub\_label][0], command=sub\_dict[sub\_label][1], compound=LEFT)

elif label.startswith('-'):

m.add\_separator()

else:

m.add\_command(label=label, image=tm[label][0], command=tm[label][1], compound=LEFT)

print('\*\*\*\*\*\*\*\*\*\*\*')

def init\_icons(self):

self.master.filenew\_icon = PhotoImage(file='images/z.png')

self.master.fileopen\_icon = PhotoImage(file='images/p.png')

self.master.signout\_icon = PhotoImage(file='images/t.png')

def init\_toolbar(self):

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

toolframe = Frame(self.master, height=20, bg='lightgray')

toolframe.pack(fil=X)

frame = ttk.Frame(toolframe)

frame.pack(side=LEFT)

for i,e in enumerate(dir(self.master)):

#print(e)

if e.endswith('\_icon'):

ttk.Button(frame, width=20, image=getattr(self.master, e), command=None).grid(row=0, column=i, padx=1,pady=1,sticky=E)

root=Tk()

root.title("")

root.geometry('400x200')

#禁止改变窗口大小

root.resizable(width=False, height=False)

App(root)

root.mainloop()

## 弹出式菜单



from tkinter import \*

from tkinter import ttk

from collections import OrderedDict

class App:

def \_\_init\_\_(self, master):

self.master=master

self.initWidgets()

def initWidgets(self):

self.text = Text(self.master, height=12, width=60, foreground='darkgray', font=('微软雅黑',12), spacing2=8, spacing3=12)

self.text.pack()

st = 'hello, welcome to China!'

self.text.insert(END, st)

self.text.bind('<Button-3>', self.popup)

self.popup\_menu = Menu(self.master, tearoff=0)

self.my\_items = (OrderedDict([('超大', 16), ('大', 14), ('中', 12), ('小', 10), ('超小', 8)]),

OrderedDict([ ('红色', 'red'), ('绿色', 'green'), ('蓝色','blue')]))

i=0

for k in ['字体大小', '颜色']:

m = Menu(self.popup\_menu, tearoff=0)

self.popup\_menu.add\_cascade(label=k, menu=m)

for lbl in self.my\_items[i]:

m.add\_command(label=lbl, command=self.handlerAdaptor(self.choose, x=lbl))

i+=1

def popup(self, event):

self.popup\_menu.post(event.x\_root, event.y\_root)

def choose(self, x):

if x in self.my\_items[0].keys():

self.text['font'] = ('微软雅黑', self.my\_items[0][x])

if x in self.my\_items[1].keys():

self.text['foreground'] = self.my\_items[1][x]

def handlerAdaptor(self, fun, \*\*kwds):

return lambda fun=fun,kwds=kwds: fun(\*\*kwds)

root=Tk()

root.title("")

App(root)

root.mainloop()

# Tkinter Canvas绘图

## creat\_xxx函数

create\_oval

create\_rectangle

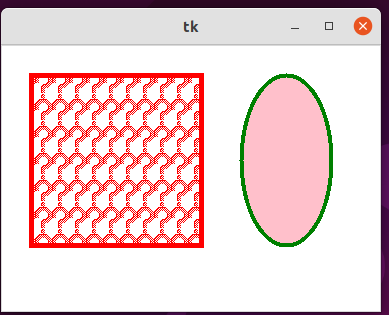
create\_arc

create\_bitmap

create\_image

create\_line

create\_polygon



from tkinter import \*

root=Tk()

cv = Canvas(root, background='white')

cv.pack(fill=BOTH, expand=YES)

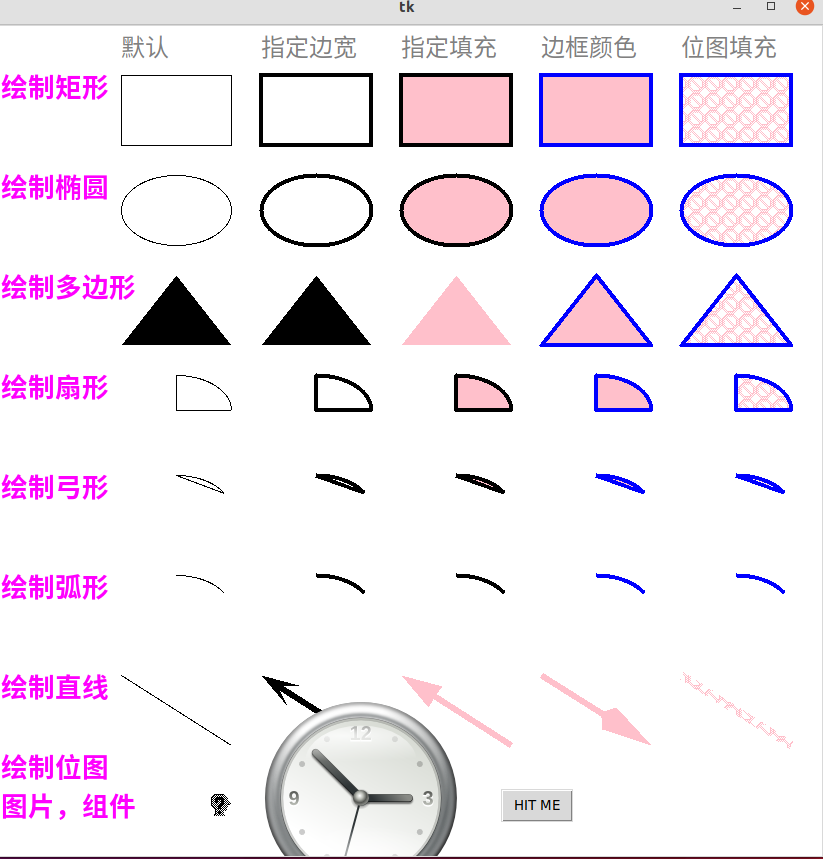
#stipple: 用位图平铺进行填充

cv.create\_rectangle(30,30,200,200, outline='red', stipple='question', fill='red', width=5)

cv.create\_oval(240,30,330,200, outline='green', fill='pink', width=4)

root.mainloop()

## creat\_xxx高级应用



from tkinter import \*

root = Tk()

root.title=('绘制图形')

cv = Canvas(root, background='white', width=830, height=830)

cv.pack(fill=BOTH, expand=YES)

columnFont=('微软雅黑', 18)

titleFont = ('微软雅黑', 20, 'bold')

for i, st in enumerate(['默认', '指定边宽','指定填充','边框颜色','位图填充']):

cv.create\_text((130+i\*140, 20), text=st, font=columnFont, fill='gray', anchor=W, justify=LEFT)

print('---------绘制矩形------------------')

cv.create\_text((10, 60), text='绘制矩形', font=titleFont, fill='magenta', anchor=W, justify=LEFT)

#定义列表，每个元素的4个值分别为: 边框厚度，填充色，边框颜色，位图填充

options=[(None, None, None, None),

(4, None, None, None),

(4, 'pink', None, None),

(4, 'pink', 'blue', None),

(4, 'pink', 'blue', 'error')]

for i, op in enumerate(options):

cv.create\_rectangle((130+i\*140, 50, 240+i\*140, 120), width=op[0], fill=op[1], outline=op[2], stipple=op[3])

print('------------绘制椭圆---------------')

cv.create\_text((10, 160), text='绘制椭圆', font=titleFont, fill='magenta', anchor=W, justify=LEFT)

options=[(None, None, None, None),

(4, None, None, None),

(4, 'pink', None, None),

(4, 'pink', 'blue', None),

(4, 'pink', 'blue', 'error')]

for i, op in enumerate(options):

cv.create\_oval((130+i\*140, 150, 240+i\*140, 220), width=op[0], fill=op[1], outline=op[2], stipple=op[3])

print('--------------绘制多边形-----------')

cv.create\_text((10, 260), text='绘制多边形', font=titleFont, fill='magenta', anchor=W, justify=LEFT)

options=[(None, None, None, None),

(4, None, None, None),

(4, 'pink', None, None),

(4, 'pink', 'blue', None),

(4, 'pink', 'blue', 'error')]

for i, op in enumerate(options):

cv.create\_polygon((130+i\*140, 320, 185+i\*140, 250, 240+i\*140, 320), width=op[0], fill=op[1], outline=op[2], stipple=op[3])

print('---------------绘制扇形------------')

cv.create\_text((10, 360), text='绘制扇形', font=titleFont, fill='magenta', anchor=W, justify=LEFT)

options=[(None, None, None, None),

(4, None, None, None),

(4, 'pink', None, None),

(4, 'pink', 'blue', None),

(4, 'pink', 'blue', 'error')]

for i, op in enumerate(options):

cv.create\_arc((130+i\*140, 350, 240+i\*140, 420), width=op[0], fill=op[1], outline=op[2], stipple=op[3])

print('-----------------绘制弓形----------')

cv.create\_text((10, 460), text='绘制弓形', font=titleFont, fill='magenta', anchor=W, justify=LEFT)

options=[(None, None, None, None),

(4, None, None, None),

(4, 'pink', None, None),

(4, 'pink', 'blue', None),

(4, 'pink', 'blue', 'error')]

for i, op in enumerate(options):

cv.create\_arc((130+i\*140, 450, 240+i\*140, 520), width=op[0], fill=op[1], outline=op[2], stipple=op[3],

start=30, extent=60, style=CHORD)

print('-----------------绘制弧形----------')

cv.create\_text((10, 560), text='绘制弧形', font=titleFont, fill='magenta', anchor=W, justify=LEFT)

options=[(None, None, None, None),

(4, None, None, None),

(4, 'pink', None, None),

(4, 'pink', 'blue', None),

(4, 'pink', 'blue', 'error')]

for i, op in enumerate(options):

cv.create\_arc((130+i\*140, 550, 240+i\*140, 620), width=op[0], fill=op[1], outline=op[2], stipple=op[3],

start=30, extent=60, style=ARC)

print('---------------绘制直线----------')

cv.create\_text((10, 660), text='绘制直线', font=titleFont, fill='magenta', anchor=W, justify=LEFT)

#5个值分别表示边框宽度，线条颜色，位图填充，箭头风格，箭头形状

options=[(None, None, None, None, None),

(6, None, None, BOTH, (20,40,10)),

(6, 'pink', None, FIRST, (40,40,10)),

(6, 'pink', None, LAST, (60,50,10)),

(8, 'pink', 'error', None, None)]

for i, op in enumerate(options):

cv.create\_line((130+i\*140, 650, 240+i\*140, 720), width=op[0], fill=op[1], stipple=op[2], arrow=op[3], arrowshape=op[4])

print('---------------绘制位图，图片，组件----------')

cv.create\_text((10, 760), text='绘制位图\n图片，组件', font=titleFont, fill='magenta', anchor=W, justify=LEFT)

#定义包括create\_bitmap, create\_image, create\_window三个方法的数组

funcs = [Canvas.create\_bitmap, Canvas.create\_image, Canvas.create\_window]

items = [{'bitmap':'questhead'},

{'image':PhotoImage(file='images/java.png')},

{'window':Button(cv, text='HIT ME', padx=10, pady=5, command=lambda:print('HIT IT')), 'anchor':W}]

for i, func in enumerate(funcs):

func(cv, 230+i\*140, 780, \*\*items[i])

root.mainloop()

## create\_xxx绘制五子棋（略）

## 操作图形项的标签

使用create\_xxx创建图形后，会以1,2,3作为图形项的id

可以为图形项传入一个 或 多个标签

总结来说，Canvas提供了如下方法来为图形项添加tag

addtag\_withtag(self, newtag, {tag|id}) #为tag或id对应图形项添加新的tag

addtag\_above(self, newtag, {tag|id}) #为tag或id对应图形项的前一个图形项添加新tag

addtag\_below(self, newtag, {tag|id}) #为tag或id对应图形项的下一个图形项添加新tag

addtag\_all(self, newtag) #为所有图形项添加tag

addtag\_closest(self, newtag, x, y) #为离(x,y)最接近的图形项添加新tag

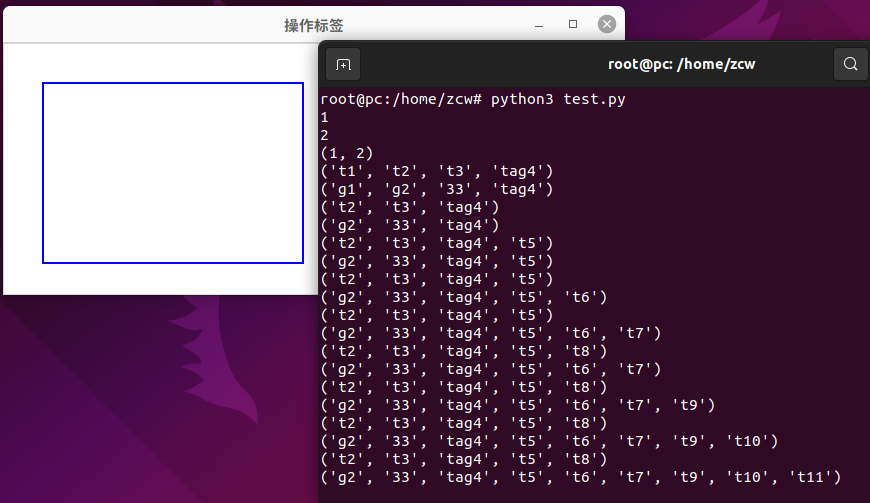
addtag\_enclosed(self, newtag, x1,y1,x2,y2) 为指定矩形区域内最上面的图形项添加新tag

addtag\_overlapping(self, newtag, x1,y1,x2,y2) 为与指定矩形区域重叠的最上面的图形项添加tag

dtag(self, \*args) 删除tag，例子: cv.dtag(1, 't1') 或 cv.dtag(oval, 'g1')

gettags(self, \*args) 获取tag，例子: cv.gettags(rt) 或 cv.gettags(2)

find\_withtag(self, {tag | id}) 查找tag，例子: cv.find\_withtag('tag4')， 返回id



from tkinter import \*

root = Tk()

root.title('操作标签')

cv = Canvas(root, background='white', width=620, height=250)

cv.pack(fill=BOTH, expand=YES)

rt = cv.create\_rectangle(40,40,300,220, outline='blue', width=2, tag=('t1', 't2','t3','tag4'))

print(rt)

oval = cv.create\_oval(350,50,580,200, outline='yellow', width=0, tag=('g1', 'g2','33','tag4'))

print(oval)

print(cv.find\_withtag('tag4'))

print(cv.gettags(rt))

print(cv.gettags(2))

cv.dtag(1, 't1')

cv.dtag(oval, 'g1')

print(cv.gettags(rt))

print(cv.gettags(2))

cv.addtag\_all('t5')

print(cv.gettags(1))

print(cv.gettags(oval))

cv.addtag\_withtag('t6', 'g2')

print(cv.gettags(1))

print(cv.gettags(oval))

cv.addtag\_above('t7', 't2')

print(cv.gettags(1))

print(cv.gettags(oval))

cv.addtag\_below('t8', 'g2')

print(cv.gettags(1))

print(cv.gettags(oval))

cv.addtag\_closest('t9', 360, 90)

print(cv.gettags(1))

print(cv.gettags(oval))

cv.addtag\_closest('t10', 30, 30, 600, 240)

print(cv.gettags(1))

print(cv.gettags(oval))

cv.addtag\_closest('t11', 250, 30, 400, 240)

print(cv.gettags(1))

print(cv.gettags(oval))

root.mainloop()

## 操作图形项

向上

向下

改变填充色

改变边框色

右下移动

位置复位

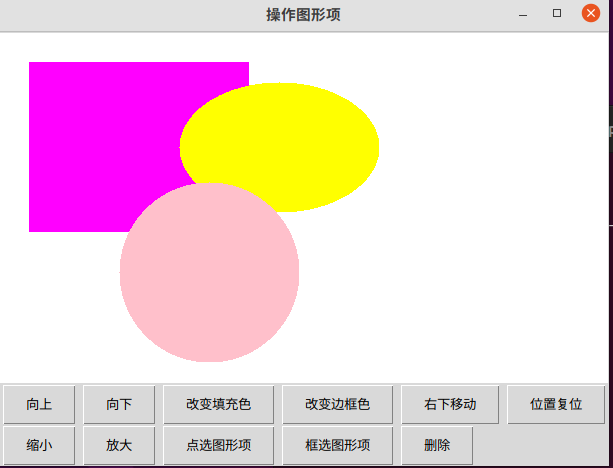
缩小

放大

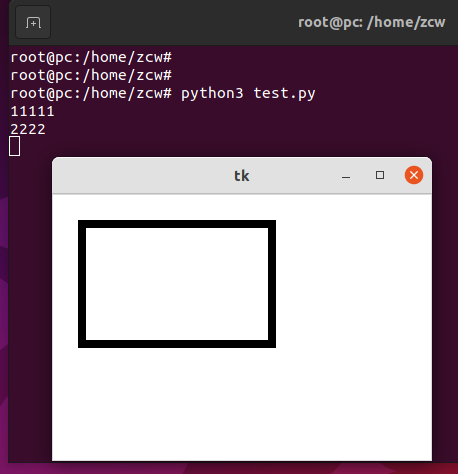
点选图形项

框选图形项

删除



## 为图形项绑定事件



#左击一次黑边框，输出两条打印

from tkinter import \*

from tkinter import ttk

root=Tk()

cv = Canvas(root,bg='white')

cv.pack()

cv.create\_rectangle(30,30,220,150,width=8,tags=('r1','r2','r3'))

def first(event):

print('11111')

def second(event):

print('2222')

cv.tag\_bind('r1', '<Button-1>', first)

cv.tag\_bind('r1', '<Button-1>', second, add=True) #add为True表示添加，否则为替代

root.mainloop()

## 绘制动画