TKINTER - GRAPHICAL USER INTERFACE(GUI)

GRAPHICAL USER INTERFACE(GUI)

This type of interface uses visual elements like windows, icons, buttons, menus, and other graphical elements to enable interaction.

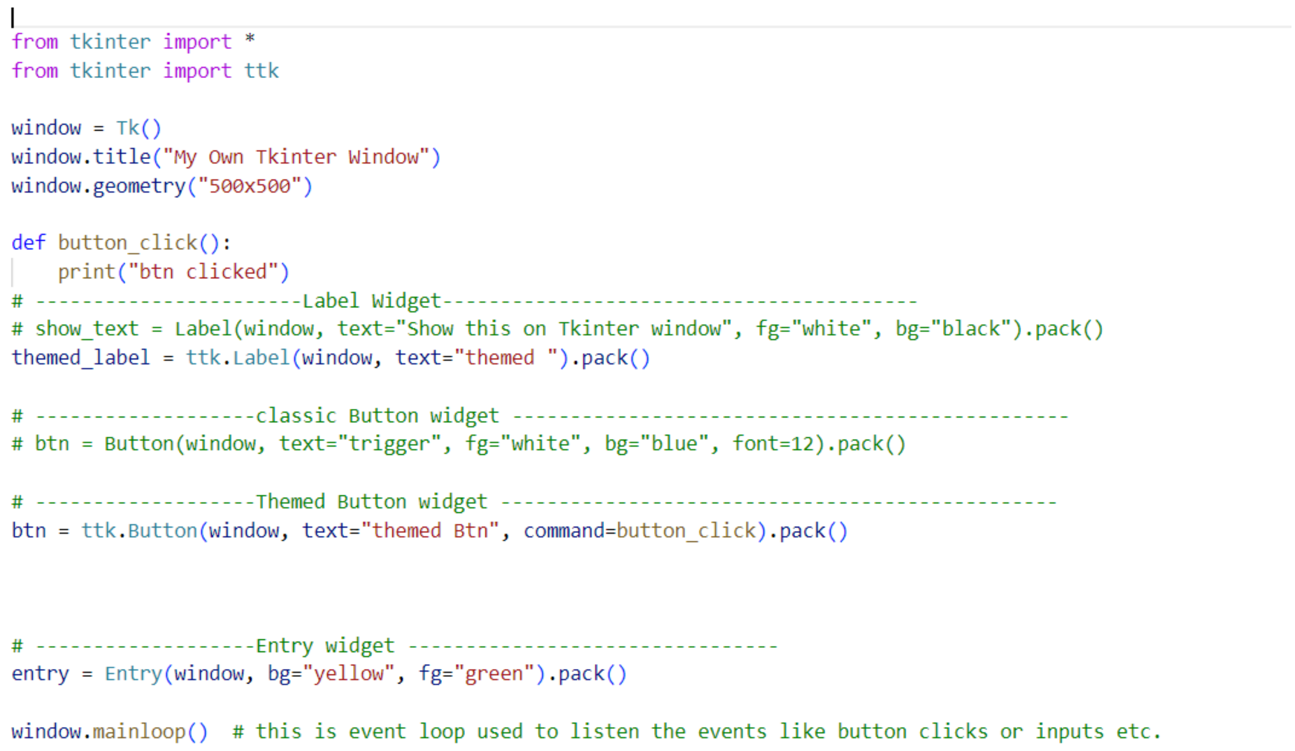
They allow users to interact with the software using a mouse, touchpad, or touch screen, making it easier for non-technical users to perform tasks.

- Tkinter is a popular and widely used graphical user interface (GUI) library in Python.
- It provides a simple way to create windows, dialogs, buttons, menus, and other GUI elements, allowing developers to build interactive applications with ease.
- One of the main strengths of Tkinter is its simplicity and ease of use, making it an excellent choice for beginners who want to dive into GUI programming.

Other python libraries for graphical user interface are:

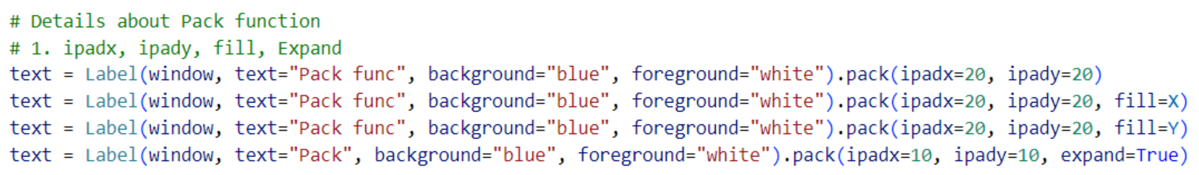
- Kivy
- Python Qt
- wxPython

Execution of Tkinter



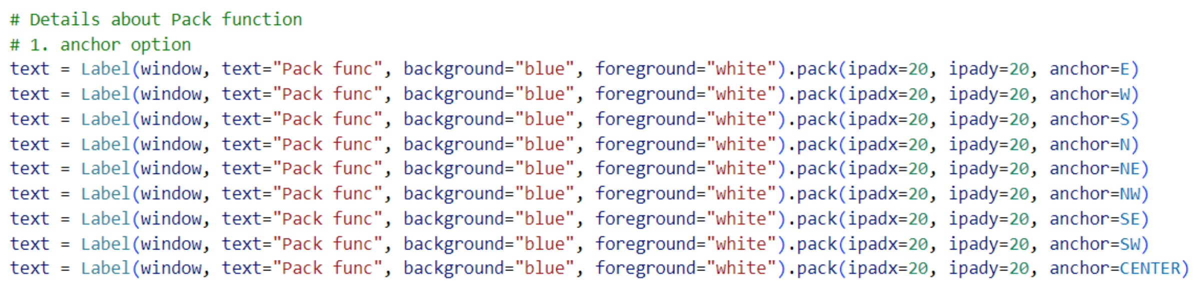
```python
from tkinter import *
from tkinter import ttk

window = Tk()
window.title("My Own Tkinter Window")
window.geometry("500x500")

def button_click():
    print("btn clicked")
# -----------------------Label Widget----------------------------------------
# show_text = Label(window, text="Show this on Tkinter window", fg="white", bg="black").pack()
themed_label = ttk.Label(window, text="themed ").pack()

# -------------------classic Button widget ------------------------------------------------
# btn = Button(window, text="trigger", fg="white", bg="blue", font=12).pack()

# -------------------Themed Button widget ------------------------------------------------
btn = ttk.Button(window, text="themed Btn", command=button_click).pack()



# -------------------Entry widget --------------------------------
entry = Entry(window, bg="yellow", fg="green").pack()

window.mainloop()  # this is event loop used to listen the events like button clicks or inputs etc.
```

Pack function



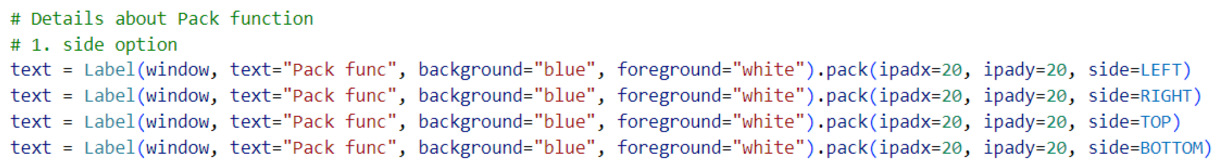
```
# Details about Pack function
# 1. ipadx, ipady, fill, Expand
text = Label(window, text="Pack func", background="blue", foreground="white").pack(ipadx=20, ipady=20)
text = Label(window, text="Pack func", background="blue", foreground="white").pack(ipadx=20, ipady=20, fill=X)
text = Label(window, text="Pack func", background="blue", foreground="white").pack(ipadx=20, ipady=20, fill=Y)
text = Label(window, text="Pack", background="blue", foreground="white").pack(ipadx=10, ipady=10, expand=True)
```

Anchor option



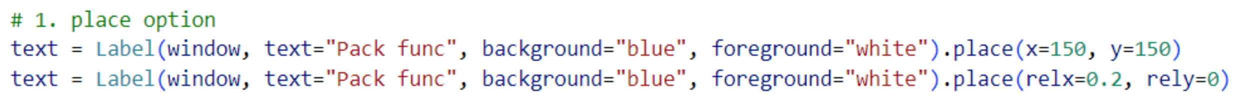
```
# Details about Pack function
# 1. anchor option
text = Label(window, text="Pack func", background="blue", foreground="white").pack(ipadx=20, ipady=20, anchor=E)
text = Label(window, text="Pack func", background="blue", foreground="white").pack(ipadx=20, ipady=20, anchor=W)
text = Label(window, text="Pack func", background="blue", foreground="white").pack(ipadx=20, ipady=20, anchor=S)
text = Label(window, text="Pack func", background="blue", foreground="white").pack(ipadx=20, ipady=20, anchor=N)
text = Label(window, text="Pack func", background="blue", foreground="white").pack(ipadx=20, ipady=20, anchor=NE)
text = Label(window, text="Pack func", background="blue", foreground="white").pack(ipadx=20, ipady=20, anchor=NW)
text = Label(window, text="Pack func", background="blue", foreground="white").pack(ipadx=20, ipady=20, anchor=SE)
text = Label(window, text="Pack func", background="blue", foreground="white").pack(ipadx=20, ipady=20, anchor=SW)
text = Label(window, text="Pack func", background="blue", foreground="white").pack(ipadx=20, ipady=20, anchor=CENTER)
```

Side option



```
# Details about Pack function
# 1. side option
text = Label(window, text="Pack func", background="blue", foreground="white").pack(ipadx=20, ipady=20, side=LEFT)
text = Label(window, text="Pack func", background="blue", foreground="white").pack(ipadx=20, ipady=20, side=RIGHT)
text = Label(window, text="Pack func", background="blue", foreground="white").pack(ipadx=20, ipady=20, side=TOP)
text = Label(window, text="Pack func", background="blue", foreground="white").pack(ipadx=20, ipady=20, side=BOTTOM)
```

Place Option (Relative and Absolute)

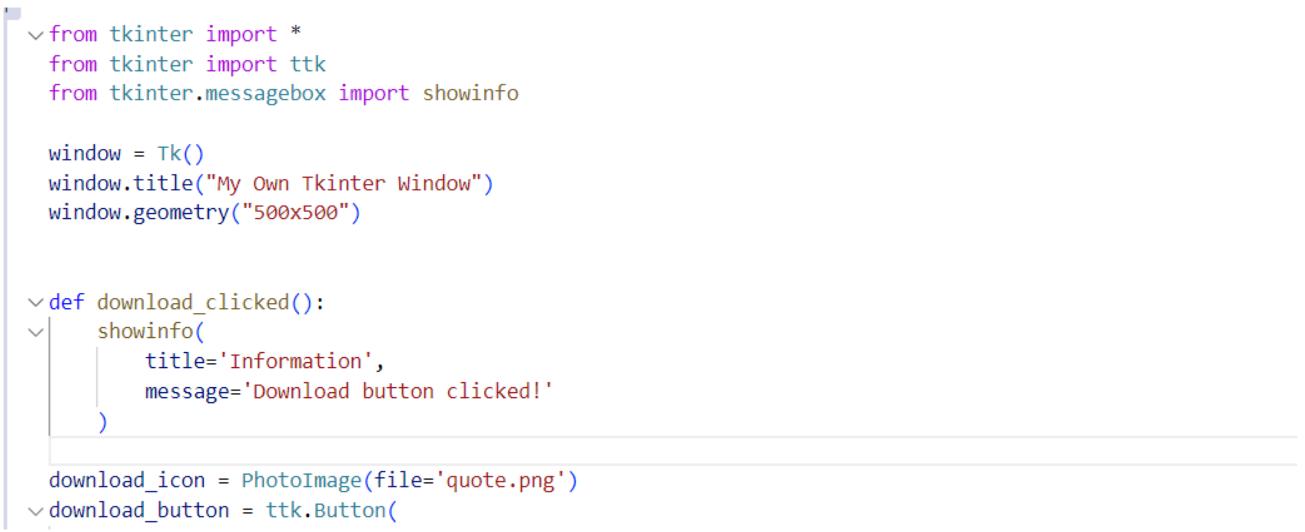


```
# 1. place option
text = Label(window, text="Pack func", background="blue", foreground="white").place(x=150, y=150)
text = Label(window, text="Pack func", background="blue", foreground="white").place(relx=0.2, rely=0)
```

x , y is absolute position

relx , rely is relative position

Button widget with image icon



```
from tkinter import *
from tkinter import ttk
from tkinter.messagebox import showinfo

window = Tk()
window.title("My Own Tkinter Window")
window.geometry("500x500")


def download_clicked():
    showinfo(
        title='Information',
        message='Download button clicked!'
    )

download_icon = PhotoImage(file='quote.png')
download_button = ttk.Button(
    window,
    image=download_icon,
    command=download_clicked,
).pack()
```

Entry Widget

```
# Entry widget (input)
Name = StringVar()
Password = StringVar()


def login():
    msg = "You are logged in with the {} and {}".format(Name.get(), Password.get())
    showinfo(
        title="Login information",
        message=msg
    )

userName = ttk.Entry(window, textvariable=Name ).pack()
password = ttk.Entry(window, textvariable=Password, show="*").pack()


# login button
btn = ttk.Button(window, text="Login", command=login).pack()
```

Checkbox widget

```
# Checkbox widget
selected = StringVar()

def checkBox_clicked():
    showinfo(
        title="Selected Language",
        message="the selected programming language is {}".format(selected.get())
    )

lang1 = ttk.Checkbutton(window,
                text='Python',
                command=checkBox_clicked,
                variable=selected,
                onvalue='python selected',
                offvalue='python not selected'
                ).pack()
lang2 = ttk.Checkbutton(window,
                text='Javascript',
                command=checkBox_clicked,
                variable=selected,
                onvalue='javascript selected',
                offvalue='javascript not selected'
                ).pack()
```

Radio button widget

```
# Radio Button widget
selected = StringVar()

def radio_clicked():
    showinfo(
        title="Selected Language",
        message="the selected programming language is {}".format(selected.get())
    )

r1 = ttk.Radiobutton(window, text='Python', value='python', variable=selected, command=radio_clicked).pack()
r2 = ttk.Radiobutton(window, text='Javascript', value='javascript', variable=selected, command=radio_clicked).pack()
r3 = ttk.Radiobutton(window, text='Go language', value='golang', variable=selected, command=radio_clicked).pack()
```

Combobox widget

```
# Combobox widget
selected = StringVar()


def selectedLanguage(event):
    showinfo(
        title="Selected Language",
        message="The selected language is %s" % selected.get()
    )

languages = ttk.Combobox(window, width=25, height=10, textvariable=selected)

languages['values'] = (
    'python', 'javascript', 'golang', 'java', 'ReactJs'
)

languages.current(0)
languages.bind('<<ComboboxSelected>>', selectedLanguage)
languages.pack()
```

Scrolledtext Widget

```
# Scrolled text widget
st = ScrolledText(window, width=50,  height=5)
st.pack(side=LEFT)


def getText():
    showinfo(
        title="laskdnfjk",
        message=st.get("1.0", END)
    )
btn = ttk.Button(window, text="getData", command=getText).pack()
```

Menu Widget

```python
# Creating menuContent
menuContent = Menu(window)

file = Menu(menuContent, tearoff=False)
menuContent.add_cascade(label ='File', menu = file)

subMenu = Menu(menuContent, tearoff=False)
subMenu.add_cascade(label ='sub menu 1')
subMenu.add_cascade(label ='sub menu 2')
subMenu.add_cascade(label ='sub menu 3')

file.add_command(label ='New File')
file.add_command(label ='Open...')
file.add_command(label ='Save')
file.add_cascade(label='preferences', menu=subMenu)
file.add_separator()
file.add_command(label ='Exit', command = window.destroy)


# Adding Edit Menu and commands
edit = Menu(menuContent, tearoff=False)
menuContent.add_cascade(label ='Edit', menu = edit)
edit.add_command(label ='Cut')
edit.add_command(label ='Copy')
edit.add_command(label ='Paste')
edit.add_command(label ='Select All')
edit.add_separator()
edit.add_command(label ='Find...')
edit.add_command(label ='Find again')

# Adding Help Menu
help_ = Menu(menuContent, tearoff=False)
menuContent.add_cascade(label ='Help', menu = help_)
help_.add_command(label ='Tk Help')
help_.add_command(label ='Demo')
help_.add_separator()
help_.add_command(label ='About Tk')

# display Menu
window.config(menu = menuContent)
```

Relief Attributes

```python
# relief attribute
# Create a button with the "raised" relief style
button_raised = Button(window, text="Raised Button", relief=RAISED)
button_raised.pack(pady=10)
```

```
# Create a button with the "sunken" relief style
button_sunken = Button(window, text="Sunken Button", relief=SUNKEN)
button_sunken.pack(pady=10)

# Create a button with the "groove" relief style
button_groove = Button(window, text="Groove Button", relief=GROOVE)
button_groove.pack(pady=10)

# Create a button with the "ridge" relief style
button_ridge = Button(window, text="Ridge Button", relief=RIDGE)
button_ridge.pack(pady=10)
```