Introduction to TKinter

Tkinter package (“Tk interface”) သည် standard Python interface ဖြစ်သည်။ Tk GUI toolkit ဟုလည်းခေါ်သည်။ GUI (Graphical User Interface)

Python နှင့် tkinter မတူပါ။

tkinter သည် Python နှင့် မတူညီသည့် language တစ်ခု ဖြစ်သည်။

tkinter ကို Python, pearl, rubby, Perl စသည့် programming language များ ဖြင့် macOS, Windwo, Linux စသည့် operation system များ အပေါ်တွင် အသုံးပြုနိုင်သည်။

Python သည် dynamically-typed language ဖြစ်သောကြောင့် variable type ကို ကြိုတင် ကြေငြာရန် မလိုအပ်ပေ။ သို့သော် Python သည် dynamically-typed language ဖြစ်သောကြောင့် variable type ကို ကြိုတင် ကြေငြာရန် မလိုအပ်ပေ။ သို့သော် tkinter တွင် variable name နှင့် variable type ကို ကြိုတင် ကြေငြာရန် လိုအပ်သည်။

Tkinter သည် Python နှင့် မတူပါ။ Tkinter တွင် user က ထည့်ပေးသည့် input များကို လက်ခံရန် အတွက် variable name နှင့် variable type ကို ကြိုတင် ကြေငြာရန် လိုအပ်သည်။

ဥပမာ - user က ထည့်ပေးသည့် နာမည်(name)ကို လက်ခံရန်အတွက် user_name နှင့် ထည့်ပေးမည့် ဒေတာသည် string ဖြစ်သည်။ String variable ဖြစ်သည်ဟု user_name = tk.StringVar() ဆိုပြီး ကြိုတင် ကြေငြာပေးရသည်။

ထို့အတူ user ၏ အသက်ကို လက်ခံရန်အတွက် user_age = tk.IntVar() ဟု ကြိုတင် ကြေငြာ ပေးရသည်။

```
x = StringVar() # Holds a string; default value ""
x = IntVar() # Holds an integer; default value 0
x = DoubleVar() # Holds a float; default value 0.0
x = BooleanVar() # Holds a boolean, returns 0 for False and 1 for True
```

Python GUI Programming With Tkinter

Python တွင် GUI framework များစွာရှိသည့်အနက် Tkinter သည် Python standard library ထဲတွင် ထည့်သွင်းထားပေးသည့် GUI framework တစ်ခု ဖြစ်သည်။

Tkinter ဖြင့် ရေးထားသည့် code များသည် Windows, macOS, Linux စသည့် operation system များတွင် ကောင်းစွာအလုပ်လုပ်သည့် cross-platform GUI framework တစ်ခု ဖြစ်သည်။

Tkinter ၏ element များသည် native operating system element ဖြစ်သောကြောင့် Tkinter ကို run သည့် operating system ကို လိုက်၍ Tkinter ၏ ပုံစံ၊ အသွင်နှင့် အမြင် ပြောင်းလဲသည်။

Tkinter သည် ပေါ့ပါးပြီး အသုံးပြုရန်အတွက် အလွယ်တကူ သင်ယူနိုင်သည်။ ထို့ကြောင့် Python application များအတွက် GUI များတည်ဆောက်ရာတွင် လူကြိုက်များသည့် framework တစ်ခု ဖြစ်သည်။ အထူးသဖြင့် functional and cross-platform ဖြစ်ရမည့် application များအတွက် အသင့်လျော်ဆုံးသော framework တစ်ခု ဖြစ်သည်။

Building Your First Python GUI Application With Tkinter

Tkinter GUI ၏ အခြေခံကျသည့် foundational element တစ်ခုမှာ window ဖြစ်သည်။ window သည် တည်ဆောက်မည့် GUI element များထည့်သွင်းထားရန် အတွက် ကွန်တိန်နာ(container) လည်း ဖြစ်သည်။

GUI element များ ဖြစ်သည့် text boxes, labels, and buttons စသည်တို့ကို widget ဟုခေါ်သည်။ ထို widget များသည် window ဆိုသည့် ကွန်တိန်နာ(container) ထဲတွင် ရှိကြသည်။

Python GUI Tkinter module ကို အသုံးပြုရန်အတွက် import လုပ်ရန် လိုသည်။

```
import tkinter as tk
```

window သည် Tkinter's Tk class ၏ instance တစ်ခု ဖြစ်သည်။ window အသစ်တစ်ခုကို အောက်ပါအတိုင်း တည်ဆောက်နိုင်သည်။ ထို instance ကို root ဟု နာမည်ပေးသည်။

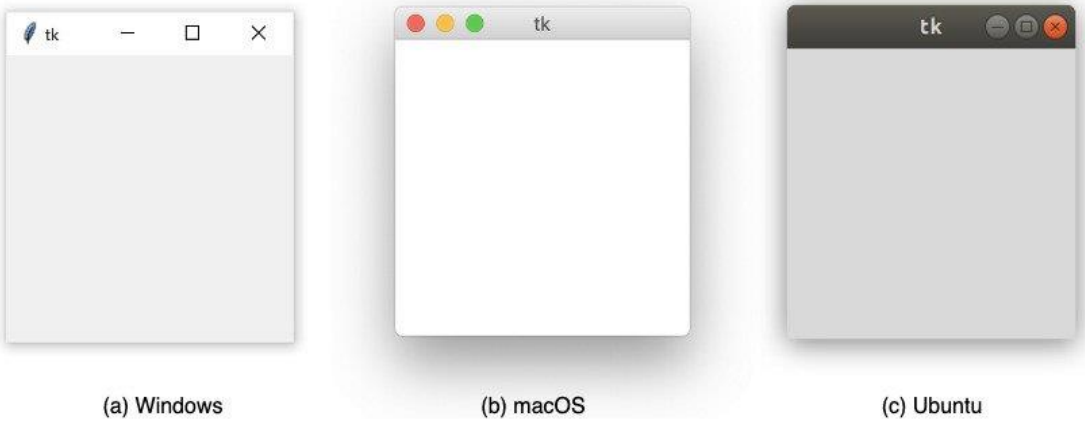
```
window = tk.Tk()
```

window ဟုလည်း နာမည်ပေးနိုင်သည်။

```
window = tk.Tk()


window.mainloop()
```

window.mainloop() အလုပ်မှာ Tkinter ၏ event loop ကို run ပေးရန် Python လှမ်းခိုင်းလိုက်ခြင်း ဖြစ်သည်။ button တစ်ခုကို click လုပ်ခြင်း သို့မဟုတ် key တစ်ခုခုကို နှိပ်လိုက်ခြင်း(key presses)ကို event ဟုခေါ်သည်။

Tkinter ဖြင့် တည်ဆောက်မည့် code များ၏ အောက်ဆုံး code အကြောင်းတွင် window.mainloop() မပါရှိပါက မည်သည့် အရာမျှ ပေါ်လာလိမ့်မည် မဟုတ်ပေ။
တည်ဆောက်လိုက်သည့်၊ သင်မြင်ရမည့် window အသွင်သည် operating system ကို လိုက်၍ ကွဲပြားသည်။



(a) Windows (b) macOS (c) Ubuntu

window တည်ဆောက်ပြီးလျှင် widget များ ထည့်သွင်းနိုင်ပြီ ဖြစ်သည်။

Adding a Widget

window တွင် စာတန်း စာသားများ ဖော်ပြလိုပါက tk.Label class ကို အသုံးပြုရသည်။ စာတန်း စာသားများကို Label ဟုခေါ်သည်။

Label widget တစ်ခု တည်ဆောက်မည်။ Label widget ကို greeting ဆိုသည့် variable name ပေး(assign လုပ်)မည်။ ထို Label widget ၏ အလုပ်မှာ "Hello, Tkinter" စာတန်း(text) ဖော်ပြပေးရန် ဖြစ်သည်။

```
greeting = tk.Label(text="Hello, Tkinter")
```

အပေါ်မှာ code တွင် Label widget ကို မည်သည့်နေရာတွင် ထားမည်။ မည်သည့် ကွန်တိန်နာ (container) ထဲတွင် ထည့်မည်ဆိုသည်ကို ဖော်ပြမထားပါ။

window ထဲသို့ Label widget ကို ထည့်သွင်းနိုင်သည် နည်းများစွာ ရှိသည်။ **.pack()** သည် Label widget ကို window ထဲသို့ ထည့်သွင်းနိုင်သည့် နည်း(method) တစ်ခု ဖြစ်သည်။

```
greeting.pack()
```

greeting.pack() ဟု ရေးလိုက်မှ Label widget တစ်ခု တည်ဆောက်ခြင်း ပြီးမြောက် သွားသည်။ Tkinter window သည် ထည့်ပေးသည့် widget နေရာ လုံလောက်ရုံသာ နေရာပေးသည်။

Working With Widgets

Widget များတည်ဆောက်ပုံ ကြောင်းသည် Python GUI framework ဖြစ်သည့် Tkinter တွင် အခြေခံ အကျဆုံးဖြစ်သည်။ Tkinter ၏ widget တိုင်းသည် class တစ်ခု ဖြစ်သည်။

widget တချို့မှာ

Widget Class	Description
Label	A widget used to display text on the screen
Button	A button that can contain text and can perform an action when clicked
Entry	A text entry widget that allows only a single line of text
Text	A text entry widget that allows multiline text entry
Frame	A rectangular region used to group related widgets or provide padding between widgets

Widget များအကြောင်းကို TkDocs tutorial တွင် အသေးစိတ် ဖတ်ရှုနိုင်သည်။
https://tkdocs.com/tutorial/index.html

Displaying Text and Images With Label Widgets

စာသားများနှင့်ပုံများကို ဖော်ပြ(display text or images)ရန်အတွက် Label widget ကို အသုံးပြုသည်။ Label widget ဖြင့် ဖော်ပြ(display လုပ်) ထားသည့် text များကို အသုံးပြုသူများ(user) တည်းဖြတ်(edit လုပ်)ခွင့် မရှိပါ။ ဖော်ပြရုံအတွက်သာ(display purposes) ဖြစ်သည်။

Label class ကို instantiating လုပ်၍ text parameter နေရာတွင် string တစ်ခုကို ထည့်ပေးခြင်းဖြင့် Label widget တစ်ခုကို တည်ဆောက်နိုင်သည်။ (create a Label widget by instantiating the Label class and passing a string to the text parameter)

```
label = tk.Label(text="Hello, Tkinter")
```

Label widget မှ text များကို ဖော်ပြသည့်အခါ system မှ default text color နှင့် default background color ကို သုံးသည်။ ပုံမှန်အားဖြင့် အဖြူရောင်နှင့် အနက်ရောင် ဖြစ်သည်။ perating system မှ setting ကို ပြောင်းလိုက်လျှင် default text color နှင့် default background ပြောင်းသွားလိမ့်မည်။

Label text ၏ foreground parameter နှင့် background parameter ကို အသုံးပြု၍ နောက်ခံအရောင်နှင့် စာသားအရောင်ကို မိမိအလိုရှိသလို ပြောင်းနိုင်သည်။ အောက်တွင် ဥပမာအဖြစ် စာသားအရောင်ကို အဖြူ နှင့် နောက်ခံအရောင်ကို အနက်အဖြစ် foreground parameter နှင့် back ground parameter ကို သုံး၍ ပြောင်းပြထားသည်။

```
label = tk.Label(
    text="Hello, Tkinter",
    foreground="white",  # Set the text color to white
    background="black"  # Set the background color to black)
```

အသုံးပြုနိုင်သည့် အရောင်အချို့မှာ "red", "orange","yellow","green","blue" နှင့် "purple" တို့ ဖြစ်သည်။

HTML တွင် အသုံးပြုနိုင်သည့် အရောင်(color) အားလုံးနီးပါး Tkinter တွင် ရနိုင်သည်။ Color chart ကို အောက်က link တွင် ရနိုင်သည်။

http://www.science.smith.edu/dftwiki/index.php/Color_Charts_for_TKinter

Colors manual page ကို အောက်က link တွင် မြင်မြင်နိုင်သည်။

http://www.tcl.tk/man/tcl8.5/TkCmd/colors.htm

အရောင် နာမည်များ အပြင် hexadecimal RGB value များကိုထည့်ပေးနိုင်သည်။

https://en.wikipedia.org/wiki/Web_colors#Hex_triplet

ဥပမာ-

```
label = tk.Label(text="Hello, Tkinter", background="#34A2FE")
```

တစ်ခါတစ်ရံ background ကို bg အတိုခေါက်ရေးသားကြသည်ကိုလည်းတွေ့ မြင်ရလိမ့်မည်။

```
label = tk.Label(text="Hello, Tkinter", fg="white", bg="black")
```

Label ၏ အမြင့်(height)နှင့် အရှည်(width)ကိုလည်း `width` and `height` parameter ကို သုံး၍ မိမိကြိုက်သလို ပြင်နိုင်သည်။ ဥပမာ-

```
label = tk.Label(
    text="Hello, Tkinter",
    fg="white",
    bg="black",
    width=10,
    height=10)
```

အပေါ်မှ ဥပမာ label ကို အောက်တွင် ဖော်ပြထားသည်။

width and height ကို 10 ဟု ထည့်ထားသော်လည်း window သည် square ပုံသဏ္ဌာန် ဖြစ်မနေရသည့် ကြောင်းမှာ width နှင့် height ကို text unit ဖြင့် တိုင်းတာသောကြောင့် ဖြစ်သည်။ horizontal text unit တစ်ခုစာသည် default system font ၏ character "0" သို့မဟုတ် number zero တစ်နေရာစာ ဖြစ်သည်။ ထို့အတူ vertical text unit တစ်နေရာစာသည် character "0" ၏ အမြင့်(height)နှင့်တူညီသည်။

Tkinter တွင် လက်မ(inche), စင်တီမီတာ(centimeter), pixels စသည်တို့ကို မသုံးဘဲ text unit ကို အသုံးပြုသည်။ across platform တွင် application များ အားလုံး တစ်ပုံစံတည်းထွက်စေရန်၊ consistent behavior ရရှိစေရန်အတွက် text unit ကို အသုံးပြုခြင်း ဖြစ်သည်။

character၏ width ဖြင့် တိုင်းတာသောကြောင့် widget ၏ အရွယ် အစား(size)သည် user's machine တွင် ရှိနေသည့် default font အတိုင်း ဖြစ်နေလိမ့်မည်။ ထို့ကြောင့် မည်သည့်အခြေအနေတွင်မဆို labels နှင့် button တို့တွင် text များကောင်းစွာ နေရာတကျ ရှိနေလိမ့်မည်။

Label များသည် text များကို ဖော်ပြရန်အတွက် ကောင်းသောလည်း user မှ ထည့်ပေးလိုက်သည့် input များကို Label အဖြစ် ဖော်ပြရန် မဖြစ်နိုင်ပါ။

User Input ကို လက်ခံနိုင်သည့် widget

User Input ကို လက်ခံနိုင်သည့် widget တချို့မှာ clickable button, Entry widget နှင့် Text widget တို့ ဖြစ်သည်။

Button Widgets

Button widget များသည် clickable button များကို ဖော်ပြရန်(display လုပ်ရန်) အတွက် ဖြစ်သည်။ button ကို click လုပ်လိုက်သည့်အခါ မိမိအလိုရှိသည့် function များကို လှမ်းခေါ် call နိုင်သည်။ မည်ကဲ့သို့ function ကို call လုပ်သည်ကို နောက်ပိုင်းတွင် ဖော်ပြထားသည်။

create and style a Button

Button widget နှင့် Label widget တို့တွင် တူညီမှုများ ရှိသည်။ Button ကို click လုပ်၍ ရသည့် Label အဖြစ်လည်း မှတ်ယူနိုင်သည်။ Label တည်ဆောက်(create)ရန် နှင့် စတိုင်(style)အတွက် သုံးသည့် keyword argument များကို Button widget တွင်လည်း အသုံးပြုနိုင်သည်။

ဥပမာ အဖြစ် အောက်တွင် blue background and yellow text ပါသည့် Button တစ်ခု ပြုလုပ် ပြထားသည်။ width=25 နှင့် height=5 ဖြစ်သည်။

```
button = tk.Button(
    text="Click me!",
    width=25,
    height=5,
    bg="blue",
    fg="yellow",)
```

Entry widget သည် user Input ကို လက်ခံနိုင်သည့် widget တစ်ခု ဖြစ်သည်။ user ၏ နာမည်(name), email address စသည်ကို လက်ခံယူရန်အတွက် entry widget ကို အသုံးပြု နိုင်သည်။ User က ထည့်ပေးမည့် input များကို လက်ခံရန်အတွက် text box အသေး တစ်ခု ဖော်ပြပေးသည်။

Label နှင့် button တည်ဆောက်(create)ရန် နှင့် စတိုင်(style) အတွက် သုံးသည့် keyword argument များကို Entry widget တွင်လည်း အသုံးပြုနိုင်သည်။

ဥပမာ အဖြစ် အောက်တွင် blue background, yellow text နှင့် 50 text unit အရှည် (width) ရှိသည့် ပါသည့် Entry widget တစ်ခု ပြုလုပ် ပြထားသည်။

```
entry = tk.Entry(fg="yellow", bg="blue", width=50)
```

Entry widget စတိုင် လုပ်ရန် parameter မရှိပါ။ get input from a user မှ မည်ကဲ့သို့ input လက်မည်ကို သိထားရန်လိုသည်။ (get input from a user.)

Entry widget က လုပ်ပေးနိုင်သည့် ကိစ္စ (၃)မျိုးမှာ

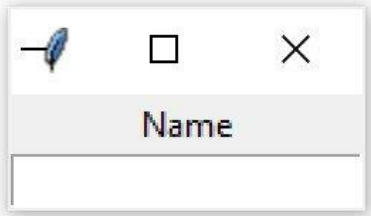
1. Retrieving text with **.get()**
2. Deleting text with **.delete() နှင့်**
3. Inserting text with **.insert() တို့ ဖြစ်သည်။**

Entry widget အလုပ်လုပ်ပုံကို နားလည်ရန် အကောင်းဆုံးနည်းမှာ လက်တွေ့ စမ်းကြည့်ခြင်း ဖြစ်သည်။

```
import tkinter as tk
window = tk.Tk()
# create a Label and an Entry widget
label = tk.Label(text="Name")
entry = tk.Entry()
```

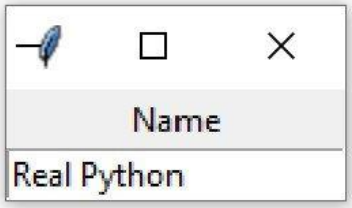
Entry widget ထဲသို့ မည်ကဲ့သို့ text အမျိုးအစားများ ထည့်ပေးမည်ကို Label က ဖော်ပြပေးသည်။

```
label.pack()
entry.pack()
```



Tkinter က အလိုအလျောက် Label widget ကို Entry widget ၏ အပေါ်တွင် နေရာ ချထားပေးသည်။ ထိုအချက်သည် `.pack()` ၏ feature တစ်ခု ဖြစ်သည်။

Entry widget ကွက်လပ် နေရာတွင် mouse ကို ထား၍ "Real Python" ဟု ရိုက်လိုက်ပါ။



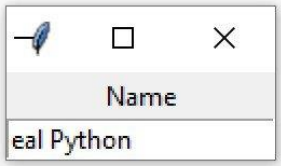
Entry widget အတွင်းသို့ text ရိုက်ထည့်ပြီး ဖြစ်သော်လည်း ထို text များကို program အတွင်းသို့ မထည့်ရသေးပါ။ **.get()** ကို သုံး၍ Entry widget အတွင်းသို့ ရိုက်ထည့်ထားသည့် text ကို retrieve လုပ်ပြီး name ဆိုသည့် variable ဖြင့် assign လုပ်သည်။

```
name = entry.get()
name
'Real Python'
```

ထည့်ပြီးသည့် text ကို .delete() ဖြင့် ဖျက်ပစ်နိုင်သည်။ .delete() method သည် integer argument ကို လက်ခံနိုင်သည်။ ဖျက်လိုသည့် အက္ခရာ(character)ကို indexing လုပ်၍ ဖျက်နိုင်သည်။ .delete(0) သည် ပထမဆုံးအက္ခရာ(first character) ကို ဖျက်ပေးသည်။

```
entry.delete(0)
```

"eal Python" ဆိုသည့် text သာ ကျန်နေလိမ့်မည်။

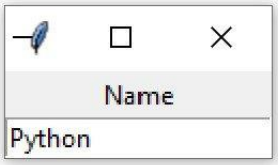


Python string object ကဲ့သို့ပင် Entry widget မှ text သည် zero indexing ဖြစ်သည်။ (indexed starting with 0)

Entry မှ အကွာရာများစွာကို ဖျက်လိုပါက slicing လုပ်နိုင်သည်။ ဥပမာ- first four letters in the Entry မှ ပထမဆုံး အကွာရာ (၄)လုံးကို ဖျက်လိုပါက အောက်ပါအတိုင်းရေးနိုင်သည်။

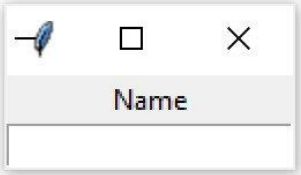
```
entry.delete(0, 4)
```

"Python" ဆိုသည့် စာလုံးသာ ကျန်နေလိမ့်မည်။



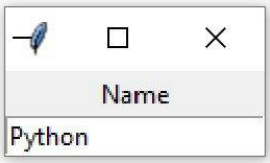
`Entry.delete()` အလုပ်လုပ်ပုံမှ Python မှ string slicing နှင့်တူညီသည်။ အစမှ အဆုံးအထိ ဖျက်လိုပါ **.delete(0, tk.END)** ဖြင့် ဖျက်နိုင်သည်။

```
entry.delete(0, tk.END)
```



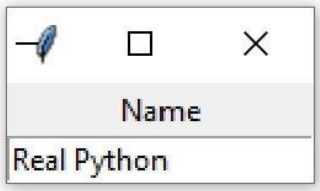
Entry widget ထဲသို့ .insert() method ဖြင့် text များ ထည့်သွင်းနိုင်သည်။

```
entry.insert(0, "Python")
```



.insert(0, "Python") တွင် 0 သည် text ကို ထည့်မည့်(insert လုပ်မည့်)နေရာဖြစ်သည်။ text ထည့်မည့်(insert လုပ်မည့်)နေရာကို မဖော်ပြထားပါက သို့မဟုတ် မထည့်ပေးပါ အစဆုံးနေရာတွင် ထည့်ပေးလိမ့်မည်။ text ထည့်ပေးနောက် ထည့်ပေးသည့်နေရာနောက်မှ စာများသည် နောက်ဆုတ်သွားလိမ့်မည်။

```
entry.insert(0, "Real ")
```



Entry widget သည် စာကြောင်း တစ်ကြောင်း (single line)အတွက်သာ ဖြစ်သည်။ အလွန်များသည့် text အတွက် မသင့်လျော်ပါ။ Text အလွန်များပါက text widget ကို အသုံးပြုသင့်သည်။

Getting Multiline User Input With Text Widgets

Text widget နှင့် Entry widget တို့ အသုံးပြုပုံတူညီသည်။ အဓိက ကွာခြားချက်မှာ Entry widget သည် စာကြောင်း တစ်ကြောင်း()အတွက်သာ ဖြစ်သည်။ Text widget သည် တစ်ကြောင်းထက်ပိုများသည့် စာကြောင်းများ စာပိုဒ်များ စာမျက်နှာများ (multiple lines of text) အတွက် ဖြစ်သည်။ Entry widget ကဲ့သို့ပင် Text widget တွင် method (၃) မျိုးရှိသည်။

Retrieve text with .get()

Delete text with .delete()

Insert text with .insert()

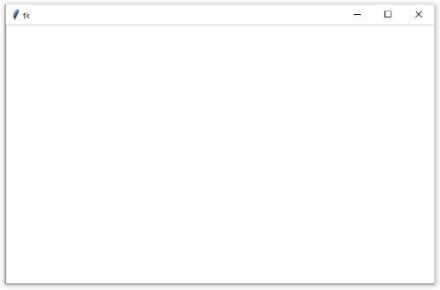
Text widget ၏ method နာမည်များသည် Entry method နာမည်များ နှင့် တူညီသော်လည်း အလုပ်လုပ်ပုံ အနည်းငယ် ကွဲပြားသည်။ window ပိတ်ရန်၊ ဖျက်ပစ်ရန် အတွက် **.destroy()** ကို သုံးသည်။

```
window.destroy()
```

Close button ကို click ၍ လည်း window ပိတ်နိုင်သည်။

```
window = tk.Tk()
text_box = tk.Text()
text_box.pack()
```

ပုံမှန်အားဖြင့်(by default)သည် Text boxe သည် Entry widget ထက် ပိုကြီးလေ့ ရှိသည်။

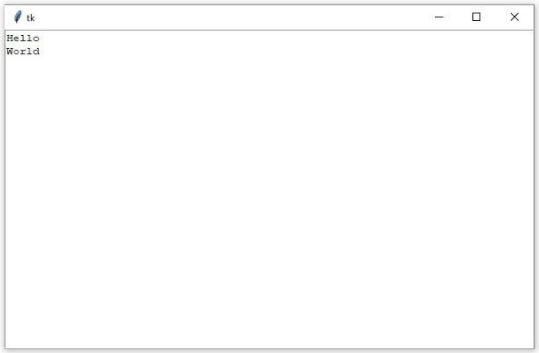


Window အတွင်းကို တစ်နေရာရာကို click လုပ်၍ text box ကို activate လုပ်နိုင်သည်။

Type in the word "Hello".

press Enter

type "World" on the second line လျှင် အောက်တွင် ဖော်ပြထားသည့် အတိုင်း ပေါ်လာလိမ့်မည်။



Entry widget ကဲ့သို့ပင် Text widget မှ text များကို .get() method ဖြင့် ထုတ်ယူ(retrieve လုပ်) နိုင်သည်။ .get() method ကို calling လုပ်သည့်အခါ လက်သည်းကွင်းထဲတွင် argument မထည့်ပေးပါက မည်သည့် text မှ return လုပ်လိမ့်မည် မဟုတ်ပေ။ (raises an exception)

```
text_box.get()
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    text_box.get()
TypeError: get() missing 1 required positional argument: 'index1'
```

Text.get() ကို အသုံးပြုရန်အတွက် အနည်းဆုံး argument တစ်ခု ထည့်ပေးရမည်။ .get() method ကို calling လုပ်သည့်အခါ single index တည်းသာထည့်ပေးပါက character တစ်ခုသာ return လုပ်လိမ့်မည်။

Text widget တွင် စာကြောင်းများစွာပါသောကြောင့် မိမိအလိုရှိသည့် အက္ခရာ() တစ်ခုခုကို အလိုရှိပါက ထို character ရှိသည့် line number နှင့် position ကို ထည့်ပေးရန် လိုသည်။ ဤအချက်သည် Entry widget နှင့် မတူညီသည့်အချက် ဖြစ်သည်။

လိုင်းနံပါတ်ကို ရေတွက်သည့်အခါ ၁ မှ စတင်ရေတွက်သည်။ character position ကို ရေတွက်သည့်အခါ ၀ မှ စတင်ရေတွက်သည်။(Line numbers start with 1, and character positions start with 0.)

"<line>.<char>" ပုံစံ မျိုး ဖြစ်သည်။ <line> နေရာတွင် လိုင်းနံပါတ်ကိုလည်းကောင်း နေရာတွင် <char> နေရာတွင် character position ကို ထည့်ပေးရသည်။ ဥပမာ- ("1.0")သည် ပထမလိုင်းမှ ပထမဆုံး အက္ခရာကို ရည်ညွှန်းသည်။ ("2.3") သည် ဒုတိယလိုင်းမှ စတုတ္ထအက္ခရာကို ရည်ညွှန်းသည်။

"Hello" text မှ .get("1.0")သည်'H' ဖြစ်သည်။

```
text_box.get("1.0")
'H'
```

"Hello" text မှ .get("1.0", "1.5") သည် 'Hello'ဖြစ်သည်။

```
text_box.get("2.0", "2.5")
'World'
```

text box မှ text အားလုံးကို ထုတ်ယူလိုပါက အစ(starting index in "1.0")မှ အဆုံး(tk.END) ဟုရေးနိုင်သည်။

```
text_box.get("1.0", tk.END)
```

```
'Hello\nWorld\n'
```

.get() မှ text များကို return ပြန်ထုတ်ပေးသည့်အခါ newline character(\n') ပါဝင်နေသည်ကို သတိပြုပါ။

text box မှ character များကို ဖျက်ပစ်ရန်အတွက် .delete() method ကို အသုံးပြုနိုင်သည်။ Entry widget မှ .delete() method ကို အသုံးပြုပုံနှင့် တူညီသည်။

.delete() ကို အသုံးပြုရ text box မှ character များကို ဖျက်ပစ်နိုင်သည့်နည်း (၂) ရှိသည်။

(က) argument တစ်ခုတည်း(single)ပေး၍ ဖျက်သည့်နည်းနှင့်

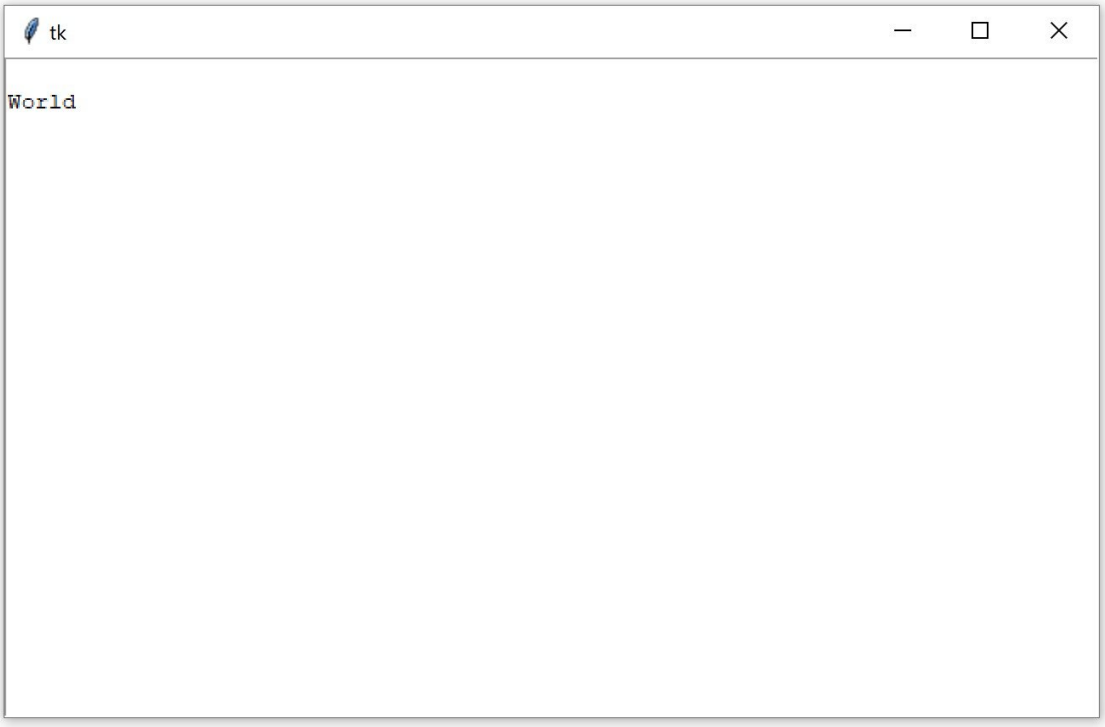
(ခ) argument နှစ်ခု (အစနှင့် အဆုံး) ပေး၍ ဖျက်သည့်နည်း တို့ ဖြစ်သည်။

argument တစ်ခုတည်း(single)ပေး၍ ဖျက်သည့်အခါ character တစ်ခုတည်း(single) ကို သာ ဖျက်နိုင်သည်။ ဥပမာ- "Hello" text မှ H အက္ခရာကို ဖျက်လိုပါက

```
text_box.delete("1.0")
```

အောက်တွင် argument နှစ်ခု (အစနှင့် အဆုံး) ပေး၍ ဖျက်သည့်နည်းကို ဥပမာ အဖြစ် ပြထားသည်။

```
text_box.delete("1.0", "1.4")
```



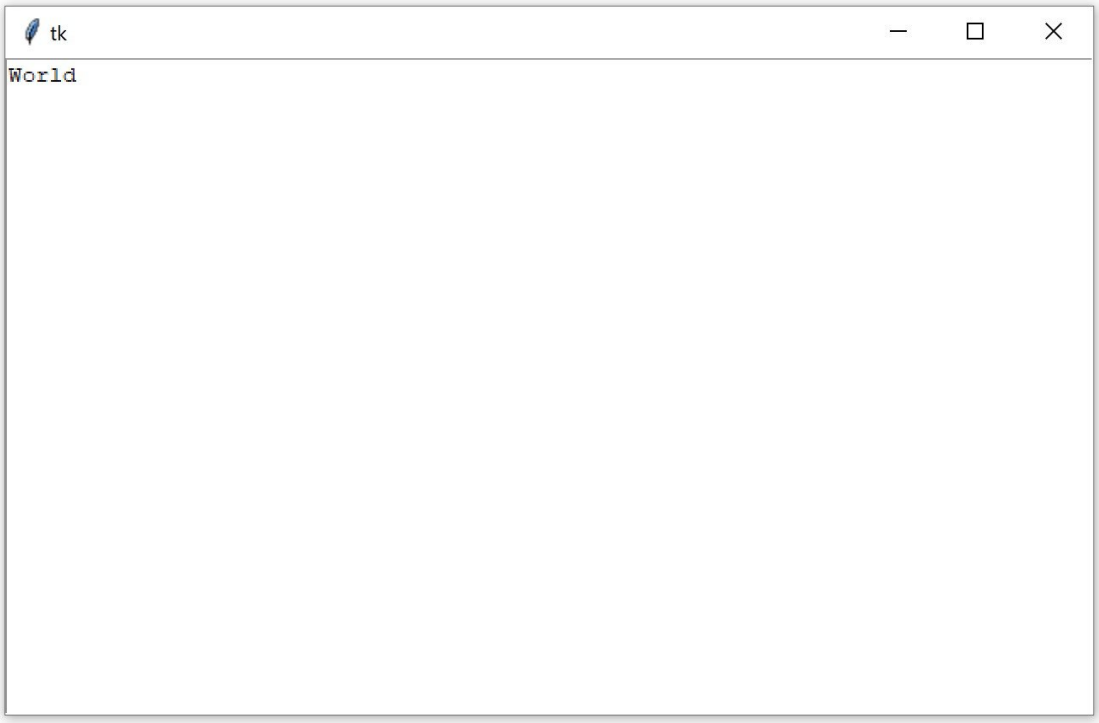
ပထမ စာကြောင်းမှ "Hello" ဆိုသည့် text ဖျက်သွားသော်လည်း စာကြောင်းနှင့် newline character ('\n') ကျန်ခဲ့သည်။ .get("1.0") စစ်ကြည့်ပါ newline character ('\n') ရှိသေးသည့်ကို တွေ့ရမည်။

```
text_box.get("1.0")
```

```
'\n'
```

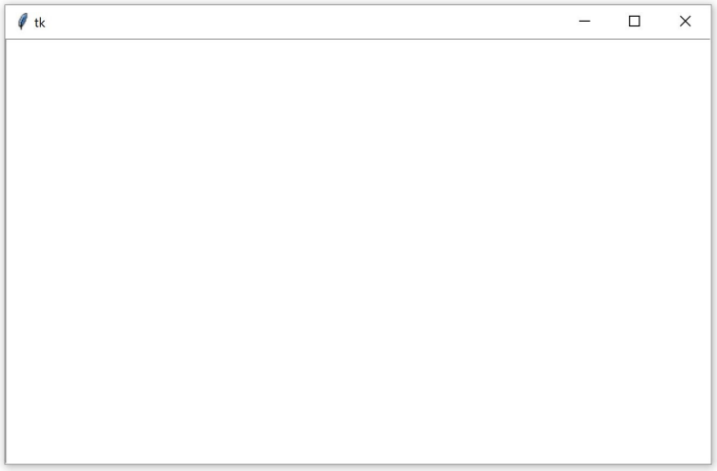
newline character ('\n') ကို ဖျက်လိုက်အခါမှသာ စာကြောင်းနေရာတစ်ခုလုံး ပျက်သွားပြီး ဒုတိယစာကြောင်း ပထမလိုင်းနေရာသို့ ရောက်လာလိမ့်မည်။

```
text_box.delete("1.0")
```



text box မှ text များ အားလုံ ပျက်သွားလိုပါက .delete("1.0", tk.END) ကို အသုံးပြုနိုင်သည်။

```
text_box.delete("1.0", tk.END)
```

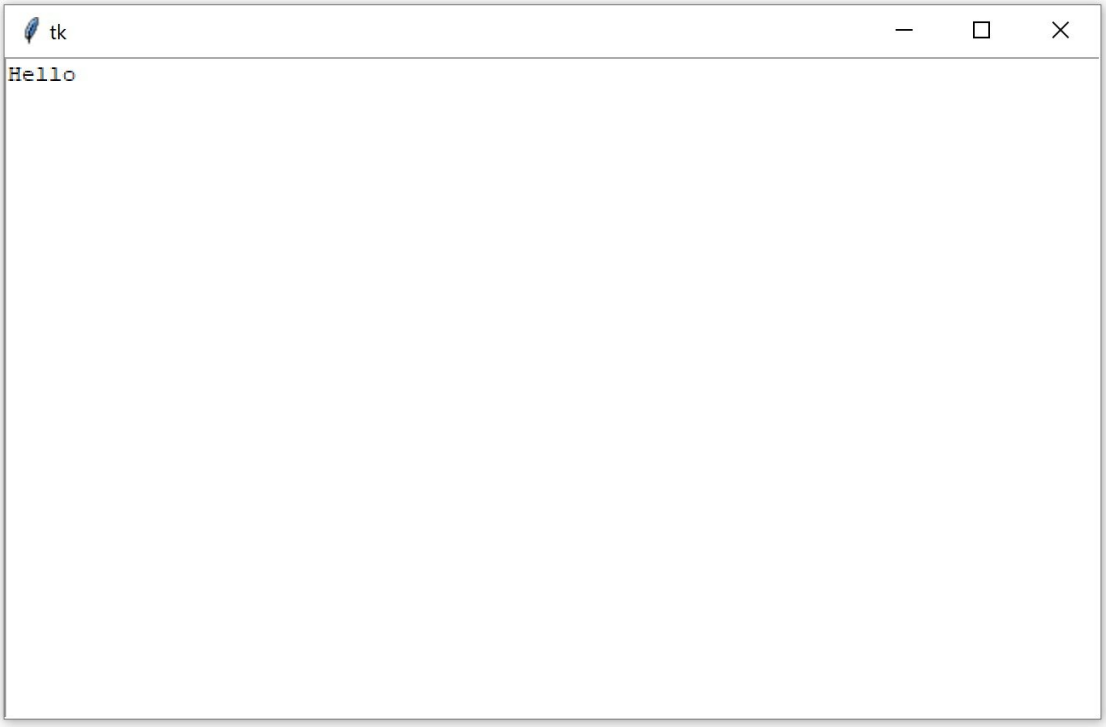


.insert() ကို အသုံးပြု၍ text_box အတွင်းသို့ text များ ထည့်ခြင်း

text box ၏ အစတွင် "Hello" ဆိုသည့် စာလုံးကို ထည့်လိုပါက

```
text_box.insert("1.0", "Hello")
```

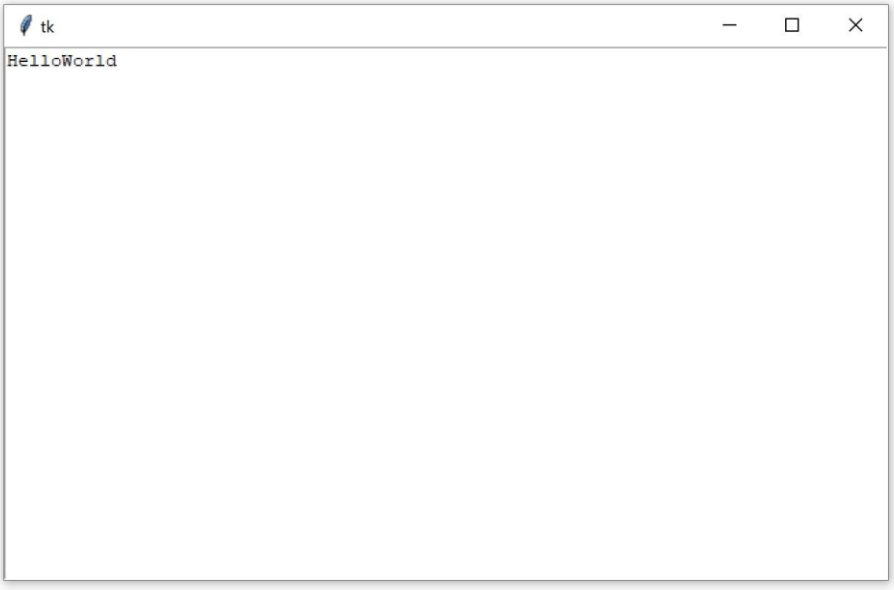
.get() တွင် သုံးသကဲ့ ထည့်မည့်နေရာ(insertion position) "<line>.<column>" ကို ပေး၍ ထည့်နိုင်သည်။



"World" ကို အောက်ပါအတိုင်း ထည့်သည့်အခါ

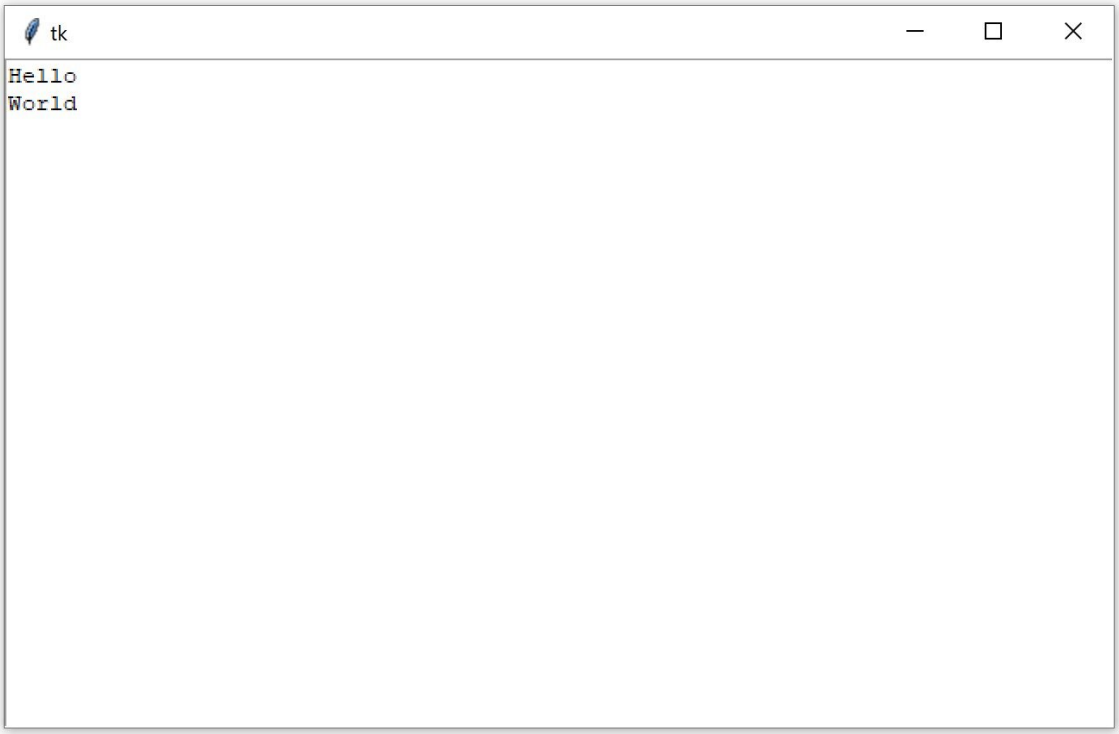
```
text_box.insert("2.0", "World")
```

ထည်ပေးလိုက်သည့် text သည် ဒုတိယစာ အကြောင်းတွင် ရှိမနေဘဲ ပထမစာကြောင်း၏ အဆုံးနေရာတွင် ရောက်နေသည်။



ထို့ကြောင့် လိုင်းအသစ်တစ်ကြောင်းတွင် text ကို ထည့်ပေးသည့်အခါ newline character ကို ထည့်ပေးရန်လိုသည်။

```
text_box.insert("2.0", "\nWorld")
```



text ကို မိမိအလို ရှိသည့်နေရာတွင် ထည့်သွင်းလိုလျှင်(Insert text at the specified position) သော်လည်းကောင်း၊ မိမိ အလိုရှိသည့် စာကြောင်းတွင် ထည့်သွင်းလိုလျှင်(Append text to the specified line) သော် လည်းကောင်း .insert() ကို အသုံးပြုနိုင်သည်။

နောက်ဆုံးစာလုံး(character)၏ နေရာကိုသိရန် တွက်ရန်အတွက် မဖြစ်နိုင်သောကြောင့် tk.END ကို အသုံးပြုနိုင်သည်။

```
text_box.insert(tk.END, "Put me at the end!")
```

ထည့်ပေးမည့် text ကို နောက်စာကြောင်း တစ်ခု၏ အစ(put it on a new line)တွင် ထည့်လိုပါက newline character (\n) ထည့်ပေးရန် လိုအပ်သည်။

```
text_box.insert(tk.END, "\nPut me on a new line!")
```

Assigning Widgets to Frames With Frame Widgets

Frame widget

widget များကို စုစည်းပြီး နေရာချသည့်အခါ organizing လုပ်သည့်အခါတွင် Frame widget သည် အလွန်အရေးကြီးသည့် widget တစ်ခု ဖြစ်သည်။ Frame widget မည်ကဲ့သို့ အလုပ်လုပ်သည်ကို ပထမဦးစွာ လေ့လာရန် လိုအပ်သည်။ ထို့နောက် Frame widget ထဲသို့ တခြားသော widget များ မည်ကဲ့သို့ assign လုပ်သည်ကို လေ့လာရန် လိုသည်။

main application window ဖြစ်သည့် root window တွင် Frame widget အလွတ်(blank)တစ်ခုကို တည်ဆောက်သည်။

```
import tkinter as tk

window = tk.Tk()
frame = tk.Frame()
frame.pack()

window.mainloop()
```

frame.pack() သည် frame ကို window ထဲတွင်ထည့်၍ pack လုပ်ခြင်း ဖြစ်သည်။ window သည် frame ကို ထည့်ပြီးသော်လည်း အရွယ်အစား သေးသောကြောင့် အောက်ပါအတိုင်းသာ မြင်ရလိမ့်မည်။

Frame widget သည် empty ဖြစ်လျှင် ဘာမျှ ပေါ်လာလိမ့်မည် မဟုတ်ပေ။ Frame များသည် တခြား widget များ ထည့်ရန်အတွက် သင့်လျော်သည့် ကွန်တိန်နာ(container)များ ဖြစ်ကြသည်။

widget များ၏ master attribute ကို အသုံးပြု၍ widget များကို frame ထဲသို့ ထည့်နိုင်သည်။

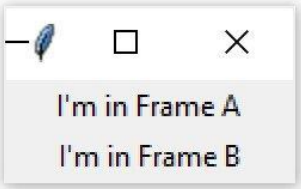
```
frame = tk.Frame()
label = tk.Label(master=frame)
```

ဥပမာအဖြစ် ဖော်ပြရန် frame_a နှင့် frame_b ကို တည်ဆောက်သည်။ frame_a ထဲတွင် "I'm in Frame A" ဆိုသည့် text ပါရှိသည်။ frame_b ထဲတွင် "I'm in Frame B" ဆိုသည့် text ပါရှိသည်။

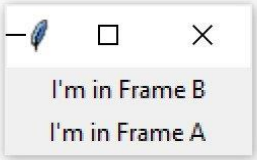
```
import tkinter as tk

window = tk.Tk()

frame_a = tk.Frame()
frame_b = tk.Frame()

label_a = tk.Label(master=frame_a, text="I'm in Frame A")
label_a.pack()

label_b = tk.Label(master=frame_b, text="I'm in Frame B")
label_b.pack()

frame_a.pack()
frame_b.pack()
```

```
window.mainloop()
```

frame_a ကို window ထဲသို့ အရင်ထည့်သည်။ (အရင် pack လုပ်သည်။) ထိုနောက် frame_b ကို ထည့်သည်။ (နောက်မှ pack လုပ်သည်။)။ ထို့ကြောင့် frame_a ၏ label သည် frame_b label ၏ အပေါ်ဘက်တွင် ရှိနေသည်။



frame_b ကို အရင် pack လုပ်လျှင် အောက်ပါအတိုင်း မြင်ရလိမ့်မည်။



တစ်နည်းအားဖြင့် label_b ကို frame_b နှင့် assign လုပ်ထားသောကြောင့် frame_b ကို ထားသည့် နေရာတွင်သာ label_b ရှိနေလိမ့်မည်။ display လုပ်လိမ့်မည်။

Adjusting Frame Appearance With Reliefs

relief attribute ကို အသုံးပြု၍ Frame widget များကို မိမိလိုချင်သည့် ဘောင်(border around the frame)ပုံစံကို ရွေးနိုင်သည်။

tk.FLAT: Has no border effect (the default value).

tk.SUNKEN: Creates a sunken effect.

tk.RAISED: Creates a raised effect.

tk.GROOVE: Creates a grooved border effect.

tk.RIDGE: Creates a ridged effect.

border effect ကိုသုံးရန်အတွက် borderwidth ကို ၁ ထက်ပိုများသည့်တန်းဖိုးထားပေး(value greater than 1) ရမည်။ relief argument:

```
1import tkinter as tk
2
```

```
 3border_effects = {
 4    "flat": tk.FLAT,
 5    "sunken": tk.SUNKEN,
 6    "raised": tk.RAISED,
 7    "groove": tk.GROOVE,
 8    "ridge": tk.RIDGE,
 9}
10
11window = tk.Tk()
12
13for relief_name, relief in border_effects.items():
14    frame = tk.Frame(master=window, relief=relief, borderwidth=5)
15    frame.pack(side=tk.LEFT)
16    label = tk.Label(master=frame, text=relief_name)
17    label.pack()
18
19window.mainloop()
```

Lines 3 to 9 - create a dictionary whose keys are the names of the different relief effects available in Tkinter. The values are the corresponding Tkinter objects. This dictionary is assigned to the border_effects variable.

Line 13 - starts a for loop to loop over each item in the border_effects dictionary.

Line 14 - creates a new Frame widget and assigns it to the window object. The relief attribute is set to the corresponding relief in the border_effects dictionary, and the border attribute is set to 5 so that the effect is visible.

Line 15 - packs the Frame into the window using .pack(). The side keyword argument tells Tkinter in which direction to pack the frame objects. You'll see more about how this works in the next section.

Lines 16 and 17 - create a Label widget to display the name of the relief and pack it into the frame object you just created.

tk.FLAT creates a frame that appears to be flat.
tk.SUNKEN adds a border that gives the frame the appearance of being sunken into the window.
tk.RAISED gives the frame a border that makes it appear to protrude from the screen.
tk.GROOVE adds a border that appears as a sunken groove around an otherwise flat frame.
tk.RIDGE gives the appearance of a raised lip around the edge of the frame.

Understanding Widget Naming Conventions

widget များကို နာမည်ပေးသည့်အခါ သင့်ကြိုက်နှစ်သက်သည့် နာမည်ပေးနိုင်သည်။ Python က ခွင့်ပြုထားသည့် identifier များ ဖြစ်ရန် လိုအပ်သည်။ သို့သော်

widget ၏ နာမည်တွင် widget class များကို ထည့်သွင်း၍ နာမည်ပေးခြင်းသည် အလွန်ကောင်းသည့် နည်း တစ်ခု ဖြစ်သည်။ widget နာမည် ဖတ်လိုက်သည်နှင့် တစ်ပြိုက်နက် မည်သည့် widget class ကို သိနိုင်သည်။ ဥပမာ- label_user_name သည် Label widget ဖြစ်သည်။ entry_age သည် Entry widget ဖြစ်သည်။

widget class များကို ထည့်ရေးသောကြောင့် variable name သည် အလွန်ရှည်လျား လိမ့်မည်။ ထို့ကြောင့် widget class များကို အတိုခေါက် ထည့်သွင်း ရေးသားရန်အတွက် Widget Naming Convention ကို အသုံးပြုသည်။

Widget Class	Variable Name Prefix	Example
`Label`	`lbl`	`lbl_name`
`Button`	`btn`	`btn_submit`
`Entry`	`ent`	`ent_age`
`Text`	`txt`	`txt_notes`
`Frame`	`frm`	`frm_address`

layout ချရန်အတွက် geometry manager (၃) မျိုးရှိသည်။

Controlling Layout With Geometry Managers

geometry manager တွင် နည်း(၃)နည်း ရှိသည်။

(က) .pack()

(ခ) .place() နှင့်

(ဂ) .grid() တို့ ဖြစ်သည်။

နည်း(၃)မျိုးအနက် application တစ်ခုတွင် နည်းတစ်နည်းကိုသာ အသုံးပြုနိုင်သည်။

frame မတူညီလျှင် သုံးထားသည့် geometry manager မတူညီသည်ကိုလည်း သတိပြုပါ။

The .pack() Geometry Manager

.pack() တွင် widget များကို Frame သို့မဟုတ် window အတွင်း၌ နေရာချထားရန်အတွက် packing algorithm ကို အသုံးပြုထားသည်။

packing algorithm has two primary steps:

Compute a rectangular area called a **parcel** that's just tall (or wide) enough to hold the widget and fills the remaining width (or height) in the window with blank space.

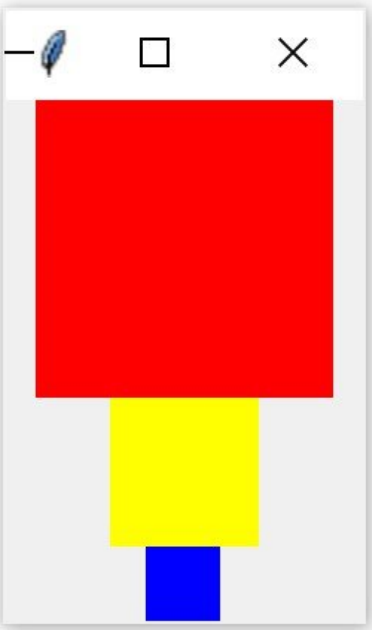
Center the widget in the parcel unless a different location is specified.

.pack() သည် အစွမ်းထက်သည့်နည်းဖြစ်သော်လည်း ပုံဖော်ကြည့်ရန်(visualize လုပ်ရန်) ခက်ခဲသည်။

The best way to get a feel for .pack() is to look at some examples. See what happens when you .pack() three Label widgets into a Frame:

```
import tkinter as tk
window = tk.Tk()

frame1 = tk.Frame(master=window, width=100, height=100, bg="red")
frame1.pack()

frame2 = tk.Frame(master=window, width=50, height=50, bg="yellow")
frame2.pack()

frame3 = tk.Frame(master=window, width=25, height=25, bg="blue")
frame3.pack()

window.mainloop()
```

.pack() ကို အသုံးပြုသည့်အခါ ပထမဆုံးရေးသည့် Frame သည် အပေါ်ဆုံးတွင် ရှိသည်။ (by default)



Frame တိုင်းကို top-most available position နေရာတွင် ထားရမည်။

.pack() ကို သုံးသည့်အခါ **anchor point** ထည့်မပေးလျှင် window ၏ အလယ်တည့်တည့် နေရာ(center) တွင်ထားပေးသည်။

was specified when `.pack()` was called for each `Frame`, they're all centered inside of

.pack() ထဲသို့ keyword argument တချို့ကို ထည့်ပေးနိုင်သည်။ fill keyword argument သည် X အတိုင်း(tk.X) သို့မဟုတ် Y အတိုင်း(tk.Y) သို့မဟုတ် နှစ်ဘက်စလုံး(tk.BOTH) ကို fill လုပ်ပေးသည်။

Optional parameters

- root = root window(optional)
- bg = background colour
- fg = foreground colour
- bd = border
- height = height of the widget.
- width = width of the widget.
- font = Font type of the text.
- cursor = cursor that appears on the widget which can be an arrow, a dot etc.

Common methods

- iconify turns the windows into icon.
- deiconify turns back the icon into window.
- state returns the current state of window.
- withdraw removes the window from the screen.
- title defines title for window.
- frame returns a window identifier which is system specific.

-End- updated on 03/01/2021