Lecture 10 - Python Graphical User Interface(GUI)

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

CSC-1004: Computational Laboratory Using Java
Course Page: [Click]

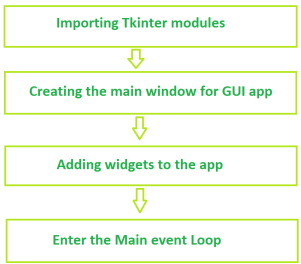
Introduction to Tkinter

Tkinter is the inbuilt Python module that is used to create GUI applications. It is one of the most commonly used modules for creating GUI applications in Python.

- Simple and easy to work with.
- No installation.
- Object-oriented interface.

Introduction to Tkinter

Fundamental structure of tkinter program:



Widgets in Tkinter are the elements of the GUI application that provide various controls to users to interact with the application.

Introduction to Tkinter

An example project of tkinter:

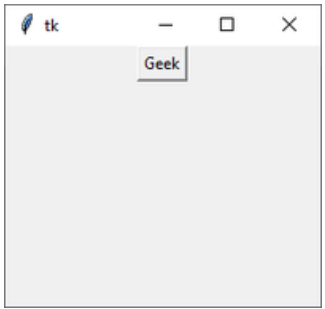
```
from tkinter import *
root = Tk()
frame = Frame(root)
frame.pack()
button = Button(frame, text ='Geek')
button.pack()
root.mainloop()
```

1. Create root window.
2. Create frame inside root window and call geometry method.
3. Create button inside frame which is inside root.
4. Call Tkinter event loop.

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Introduction to Tkinter

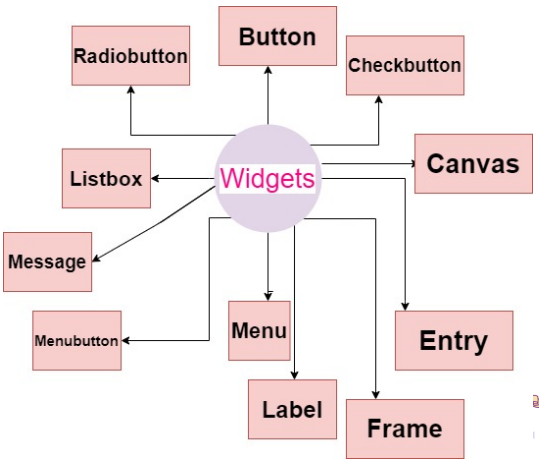
An example project of tkinter:



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Widgets in Tkinter

Widgets in Tkinter are the elements of the GUI application that provide various controls to users to interact with the application. The core widget classes are:



Widgets in Tkinter

Geometry Management. Creating a new widget doesn't mean that it will appear on the screen. To display it, we need to call a special method:

- pack(): The Pack geometry manager packs widgets in rows or columns.
- grid(): The Grid geometry manager puts the widgets in a 2-dimensional table.
- place(): The Place geometry manager allows you explicitly set the position and size of a window, either in absolute terms, or relative to another window.

Canvas Widget in Tkinter

The Canvas widget lets us display various graphics on the application. It can be used to draw simple shapes to complicated graphs. We can also display various kinds of custom widgets according to our needs.

```
C = Canvas(root, height, width, bd, bg, ..)
```

- root = root window.
- height = height of the canvas widget.
- width = width of the canvas widget.
- bg = background colour for canvas.
- bd = border of the canvas window.

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

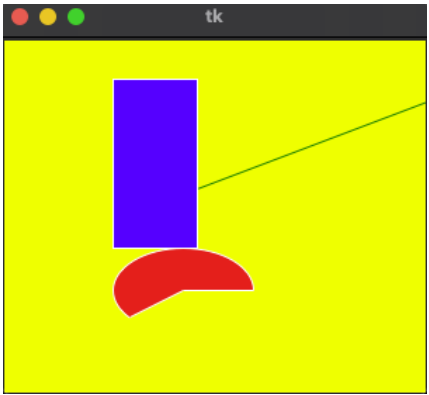
Canvas Widget in Tkinter

Some common drawing methods:

```
from tkinter import *
root = Tk()
C = Canvas(root, bg="yellow", height=250, width=300)
line = C.create_line(108, 120, 320, 40, fill="green")
arc = C.create_arc(180, 150, 80, 210, start=0, extent=220, fill="red")
oval = C.create_rectangle(80, 30, 140, 150,fill="blue")
C.pack()
mainloop()
```

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Canvas Widget in Tkinter



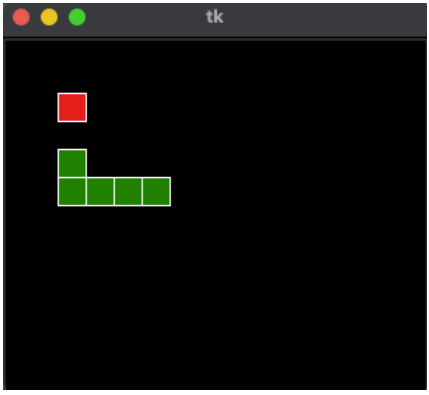
香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Canvas Widget in Tkinter

```
from tkinter import *
root = Tk()
canvas = Canvas(root, bg="black", height=250, width=300)
cell_size = 20
snake = [(100, 100), (80, 100), (60, 100), (40, 100), (40, 80)]
food = (40, 40)
for x, y in snake:
    canvas.create_rectangle(x, y, x + cell_size, y + cell_size, fill='green')
canvas.create_rectangle(food[0], food[1], food[0] + cell_size, food[1] + cell_size,
fill='red')
canvas.pack()
mainloop()
```

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Canvas Widget in Tkinter



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Binding function in Tkinter

The binding function is used to deal with the events. We can:

- bind Python's functions and methods to an event.
- bind Python's functions to any particular widget.

Binding function in Tkinter

Binding mouse movement with tkinter Frame.

```
from tkinter import *
from tkinter.ttk import *
root = Tk()
root.geometry('200x100')
def enter(event):
    print('Button-2 pressed at x = % d, y = % d'%(event.x, event.y))
frame1 = Frame(root, height = 100, width = 200)
frame1.bind('<Enter>', enter)
frame1.pack()
mainloop()
```

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Binding function in Tkinter

Binding Mouse buttons with Tkinter Frame.

```
from tkinter import *
from tkinter.ttk import *
root = Tk()
root.geometry('200x100')
def double_click(event):
    print('Double clicked at x = % d, y = % d'%(event.x, event.y))
frame1 = Frame(root, height = 100, width = 200)
frame1.bind('<Double 1>', double_click)
frame1.pack()
mainloop()
```

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Binding function in Tkinter

Binding keyboard buttons with the root window.

```
from tkinter import *
from tkinter.ttk import *
def key_press(event):
    if event.keysym in ['Left', 'Right', 'Up', 'Down']:
        direction = event.keysym
        print(direction, 'is pressed')
root = Tk()
root.geometry('200x100')
root.bind('<Key>', key_press)
mainloop()
```

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Tkinter Advance

Tkinter provides a variety of built-in functions to develop interactive and featured GUI.

- The after() function is also a universal function that can be used directly on the root as well as with other widgets. The function will be run after ms milliseconds.

```
after(parent, ms, function = None, *args)
```

- The destroy() function is a universal widget method i.e we can use this method with any of the available widgets as well as with the main tkinter window.

```
widget_object = Widget(parent, command = widget_class_object.destroy)
```

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Tkinter Advance

```
from tkinter import Tk, mainloop, TOP
from tkinter.ttk import Button
from time import time
root = Tk()
button = Button(root, text = 'Geeks')
button.pack(side = TOP, pady = 5)
print('Running...')
start = time()
root.after(5000, root.destroy)
mainloop()
end = time()
print('Destroyed after % d seconds' % (end-start))
```

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Tkinter Advance

What's the role of .after() and .destroy() functions in the Snanke Game?

Check the example code named "after_snake_canvas_example.py"!

Question and Answering (Q&A)