


TRƯỜNG: ĐH Công thương TP HCM		
KHOA: CÔNG NGHỆ THÔNG TIN	BUỔI 10. Tkinter –	
BỘ MÔN: KHDL&TTNT	Các thành phần cơ	
MH: LẬP TRÌNH PYTHON	bản	

A. MỤC TIÊU

- Thiết kế giao diện với tkinter

B. DỤNG CỤ - THIẾT BỊ THÍ NGHIỆM CHO MỘT SV

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

C. NỘI DUNG THỰC HÀNH

1. Giới thiệu về tkinter và GUI

1.1. Định nghĩa và ý nghĩa của GUI

GUI (Graphical User Interface) là một loại giao diện mà người dùng có thể tương tác với hệ thống thông qua các thành phần đồ họa như cửa sổ, nút, ô nhập liệu, hình ảnh, và nhiều hơn nữa. Điểm đặc biệt của GUI là sự trực quan và dễ sử dụng, cho phép người dùng tương tác với máy tính một cách tự nhiên hơn so với việc sử dụng dòng lệnh.

GUI đóng vai trò quan trọng trong lập trình bởi nó mang lại nhiều lợi ích đối với cả người dùng và nhà phát triển. Đầu tiên, GUI tăng trải nghiệm người dùng bằng cách cung cấp một giao diện trực quan và dễ tiếp cận. Việc tương tác với ứng dụng thông qua các hình ảnh, nút, và cửa sổ giúp người dùng dễ dàng hiểu và sử dụng chương trình một cách thuận tiện.

Ngoài ra, GUI cũng tối ưu hóa hiệu suất làm việc. Thay vì phải ghi nhớ các lệnh hoặc cú pháp phức tạp, người dùng có thể thao tác và điều chỉnh các thao tác một cách nhanh chóng chỉ bằng cách click chuột và kéo thả. Điều này giúp tăng cường hiệu suất và giảm thời gian cần thiết cho việc thực hiện các tác vụ.

Một ứng dụng có GUI cũng dễ dàng phân phối và triển khai trên nhiều hệ thống. Điều này tăng tính linh hoạt và sự tiếp cận của ứng dụng, cho phép người dùng trải nghiệm và sử dụng chương trình trên nhiều thiết bị khác nhau mà không gặp khó khăn.

Cuối cùng, GUI không chỉ phù hợp cho ứng dụng desktop mà còn được sử dụng rộng rãi trong việc phát triển ứng dụng di động. Việc tạo ra giao diện người dùng trực quan và thân thiện trên cả hai nền tảng này giúp tối ưu hóa trải nghiệm người dùng và đáp ứng nhu cầu ngày càng cao về tính di động.

1.2. Giới thiệu về tkinter

Tkinter là một thư viện GUI (Graphical User Interface) tiêu chuẩn của Python, cung cấp các công cụ và thành phần để phát triển ứng dụng có giao diện người dùng trực quan. Tên "Tkinter" là viết tắt của "Tk interface" vì nó kết hợp sức mạnh của thư viện Tk của Tcl (Tool Command Language) với Python, tạo ra một cách dễ dàng và linh hoạt để tạo GUI đơn giản trong Python.

Đặc điểm chính của Tkinter:

- Dễ sử dụng: Tkinter được tích hợp sẵn trong Python, không cần cài đặt thêm, điều này giúp người mới bắt đầu có thể nhanh chóng tạo ra các ứng dụng GUI.
- Giao diện đồ họa chéo nền tảng (cross-platform GUI): Ứng dụng được viết bằng Tkinter có thể chạy trên nhiều hệ điều hành như Windows, macOS và Linux mà không cần sửa đổi nhiều mã nguồn.
- Cung cấp các thành phần GUI phổ biến: Tkinter cung cấp các thành phần như cửa sổ, nút, nhãn, ô nhập liệu, danh sách, v.v. để xây dựng giao diện người dùng đồ họa.
- Hỗ trợ sự kiện và xử lý sự kiện: Tkinter cho phép xử lý các sự kiện như click chuột, nhập liệu từ bàn phím, và các sự kiện khác để tương tác với người dùng.
- Được sử dụng rộng rãi trong cộng đồng Python: Tkinter là một thư viện phổ biến và được sử dụng rộng rãi trong việc phát triển ứng dụng desktop Python.

Ví dụ minh họa:

Dưới đây là một ví dụ đơn giản về cách tạo một cửa sổ sử dụng Tkinter trong Python:

```
import tkinter as tk

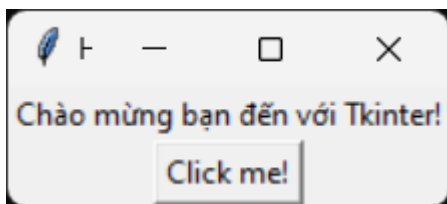
# Tạo cửa sổ chính
root = tk.Tk()
root.title("Hello Tkinter")
```

```
# Thêm nhãn vào cửa sổ
label = tk.Label(root, text="Chào mừng bạn đến với
Tkinter!")
label.pack()

# Thêm nút vào cửa sổ
button = tk.Button(root, text="Click me!",
command=root.quit)
button.pack()

# Loop chạy ứng dụng
root.mainloop()
```

Kết quả



1.3. Kiểm tra cài đặt tkinter

Thông thường, gói tkinter được cài đặt mặc định cùng với python. Để kiểm tra xem tkinter đã được cài đặt trên máy tính của bạn hay chưa, bạn có thể thực hiện các bước sau:

- Bước 1: Mở Python Interpreter

Mở terminal (hoặc Command Prompt trên Windows) và gõ lệnh sau để mở Python Interpreter:

ipython

- Bước 2: Kiểm tra cài đặt tkinter

Trong Python Interpreter, nhập các dòng lệnh sau:

```
import tkinter
print(tkinter.TkVersion)
```

Nếu tkinter đã được cài đặt và hoạt động, bạn sẽ nhìn thấy phiên bản của tkinter.

2. Xây dựng cửa sổ và các thành phần cơ bản

2.1. Tạo cửa sổ chính

Để tạo cửa sổ chính (main window) trong tkinter, bạn có thể sử dụng class Tk từ tkinter. Dưới đây là một ví dụ đơn giản về cách tạo cửa sổ chính:

```
import tkinter as tk

# Tạo cửa sổ chính
root = tk.Tk()
root.title("Cửa sổ chính")

# Để có thể xem được cửa sổ, chúng ta cần khởi động vòng
lặp mainloop
root.mainloop()
```

** Một số thuộc tính để thay đổi hiển thị cửa sổ chính*

- title: Đặt tiêu đề cho cửa sổ
- geometry: Thiết lập kích thước và vị trí của cửa sổ
- resizable: Cho phép/ Không cho phép thay đổi kích thước cửa sổ
- configure: Cấu hình các thuộc tính khác của cửa sổ

Ví dụ:

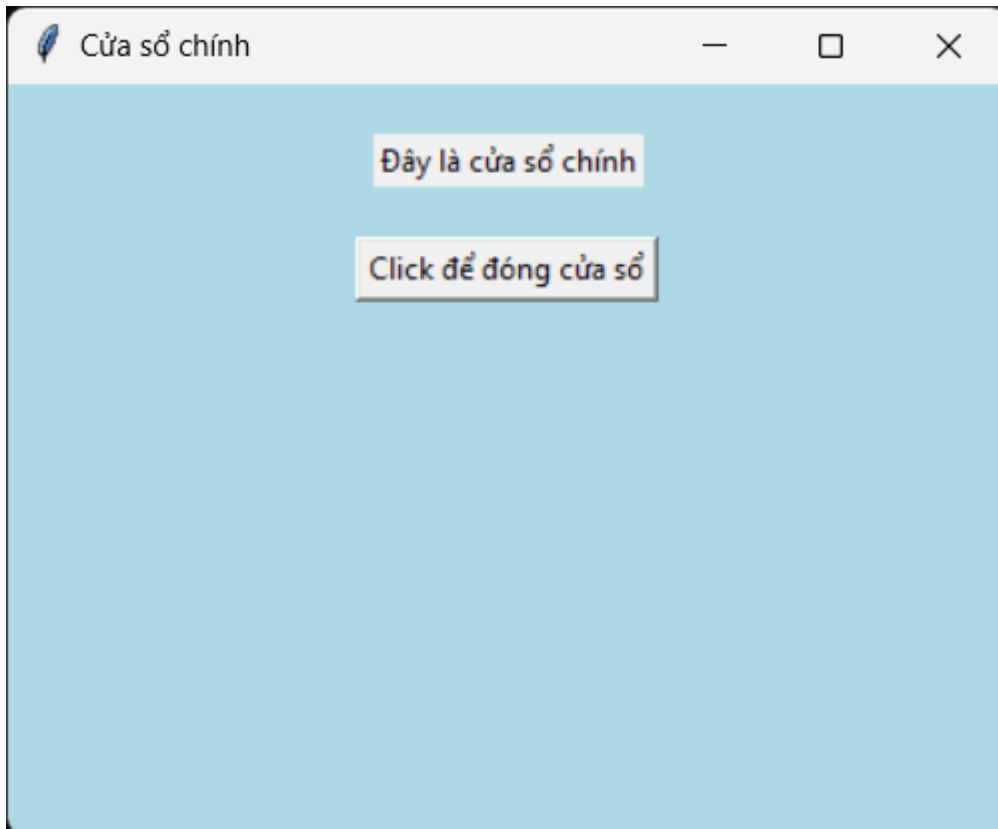
```
import tkinter as tk

root = tk.Tk()
root.title("Cửa sổ chính")
root.geometry("400x300+100+50") # Kích thước 400x300, vị
trí 100, 50 trên màn hình
root.resizable(width=True, height=True) # Cho phép thay
đổi kích thước cửa sổ
root.configure(bg='lightblue') # Đổi màu nền của cửa sổ

label = tk.Label(root, text="Đây là cửa sổ chính")
label.pack(pady=20)
```

```
button = tk.Button(root, text="Click để đóng cửa sổ",  
command=root.destroy)  
button.pack()  
  
root.mainloop()
```

Kết quả



2.2. Các thành phần cơ bản

2.2.1. Nhãn (label)

Dùng để hiển thị văn bản hoặc hình ảnh không thể chỉnh sửa.

* Các thuộc tính sử dụng trong nhãn

- text: Văn bản của nhãn
- font: Font chữ
- bg hoặc background: Màu nền
- fg hoặc foreground: Màu chữ

- width và height: Kích thước
- justify: Căn chỉnh văn bản

Ví dụ:

```
import tkinter as tk

root = tk.Tk()
root.title("Định dạng nhãn")

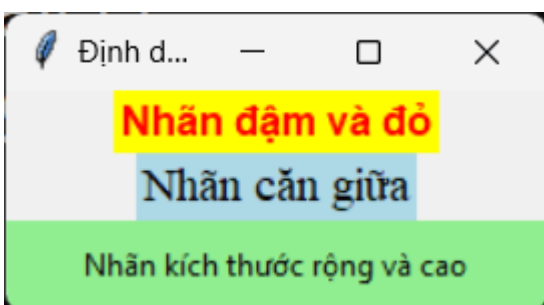
# Nhãn với văn bản đậm, màu đỏ, font Arial, và nền màu vàng
label1 = tk.Label(root, text="Nhãn đậm và đỏ",
font=("Arial", 12, "bold"), fg="red", bg="yellow")
label1.pack()

# Nhãn căn giữa với font chữ và màu nền khác
label2 = tk.Label(root, text="Nhãn căn giữa", font=("Times
New Roman", 14), justify="center", bg="lightblue")
label2.pack()

# Nhãn có kích thước rộng và cao
label3 = tk.Label(root, text="Nhãn kích thước rộng và
cao", width=30, height=2, bg="lightgreen")
label3.pack()

root.mainloop()
```

Kết quả



* **Hiển thị ảnh bằng label**

Có thể sử dụng PhotoImage từ thư viện tkinter để hiển thị hình ảnh trong nhãn (Label).

Ví dụ

```
import tkinter as tk
from tkinter import PhotoImage

root = tk.Tk()
root.title("Hiển thị Hình Ảnh")

# Tạo một đối tượng PhotoImage từ đường dẫn của hình ảnh
image_path = "tkinter.png"
photo = PhotoImage(file=image_path)

# Tạo nhãn và hiển thị hình ảnh
label = tk.Label(root, image=photo)
label.pack()

root.mainloop()
```

Kết quả



Lưu ý rằng PhotoImage chỉ hỗ trợ một số định dạng hình ảnh như GIF, PGM, PPM và PNG. Nếu bạn sử dụng định dạng hình ảnh khác như JPG, bạn cần sử dụng Pillow để chuyển đổi.

```
import tkinter as tk
from PIL import Image, ImageTk

root = tk.Tk()
root.title("Hiển thị Hình Ảnh")

# Load hình ảnh từ đường dẫn
image = Image.open("tkinter.jpg")
# Chuyển đổi hình ảnh thành định dạng có thể sử dụng trong
tkinter
photo = ImageTk.PhotoImage(image)

# Tạo nhãn và hiển thị hình ảnh
label = tk.Label(root, image=photo)
label.pack()

root.mainloop()
```

Chú ý rằng để chạy được đoạn code trên, python cần gói pillow. Bạn có thể cài đặt bằng lệnh *pip install pillow*

2.2.2. Nút (Button)

Trong tkinter, Button là một thành phần cho phép người dùng tương tác bằng cách nhấn vào đó để thực hiện một hành động.

* Tạo Button

```
button = tk.Button(root, text="Click me!",
command=your_function)
button.pack()
```

trong đó:

- root: Cửa sổ cha (ví dụ: Tk() instance).
- text: Văn bản được hiển thị trên nút.
- command: Hàm sẽ được gọi khi nút được nhấn.

* Các thuộc tính

- text: Văn bản hiển thị trên nút.
- command: Hàm sẽ được gọi khi nút được nhấn.
- width và height: Độ rộng và chiều cao của nút.
- bg hoặc background: Màu nền của nút.
- fg hoặc foreground: Màu của văn bản trên nút.
- font: Font chữ của văn bản trên nút.
- state: Trạng thái của nút (normal, disabled).
- relief: Loại relief (phong cách viền) của nút.

Ví dụ

```
import tkinter as tk

def say_hello():
    label.config(text="Hello Button!")

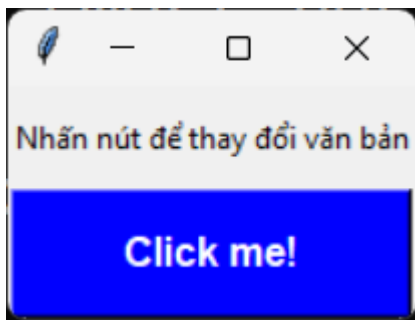
root = tk.Tk()
root.title("Button Demo")

# Tạo một nhãn
label = tk.Label(root, text="Nhấn nút để thay đổi văn bản")
label.pack(pady=10)

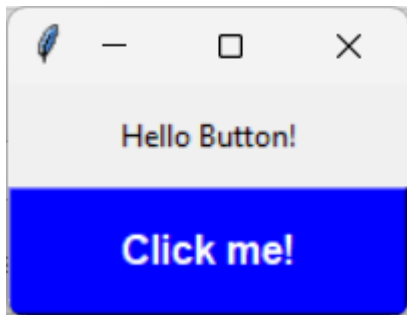
# Tạo nút để thực hiện hành động
button = tk.Button(root, text="Click me!",
                    command=say_hello, width=15, height=2,
                    bg="blue", fg="white", font=("Arial",
                    12, "bold"), relief="raised")
button.pack()

root.mainloop()
```

Kết quả



Sau khi nhấn vào nút



2.2.3. Ô nhập liệu (Entry)

Entry là một thành phần cho phép người dùng nhập dữ liệu từ bàn phím.

* Tạo Entry

```
entry = tk.Entry(root, width=30)
entry.pack()
```

trong đó:

- root: Cửa sổ cha (ví dụ: Tk() instance).
- width: Độ rộng của ô nhập liệu (số ký tự).

* Thuộc tính của Entry

- width: Độ rộng của ô nhập liệu (số ký tự).
- show: Hiển thị một ký tự thay thế khi nhập mật khẩu (ví dụ: "*").
- bg hoặc background: Màu nền của ô nhập liệu.
- fg hoặc foreground: Màu của văn bản trong ô nhập liệu.
- font: Font chữ của văn bản trong ô nhập liệu.
- state: Trạng thái của ô nhập liệu (normal, disabled).
- justify: Căn chỉnh văn bản bên trong ô nhập liệu (left, right, center).

- insertbackground: Màu của dấu nháy khi nhập liệu vào ô (cảm thấy một phần của highlightcolor).

Ví dụ:

```
import tkinter as tk

def show_text():
    text = entry.get()
    label.config(text=f"Hello: {text}")

root = tk.Tk()
root.title("Entry Demo")

# Tạo ô nhập liệu
entry = tk.Entry(root, width=30, font=("Arial", 12))
entry.pack(pady=10)

# Tạo nút để hiển thị văn bản trong ô nhập liệu
button = tk.Button(root, text="Hiển thị",
command=show_text)
button.pack()

# Nhấn để hiển thị văn bản từ ô nhập liệu
label = tk.Label(root, text="")
label.pack(pady=10)

root.mainloop()
```

2.2.4. Hộp chọn (Checkbutton)

Checkbutton là một thành phần cho phép người dùng chọn hoặc bỏ chọn một hoặc nhiều tùy chọn.

* Tạo Checkbutton

```
check_var1 = tk.BooleanVar()
check_var2 = tk.BooleanVar()
```

```
checkboxbutton1 = tk.Checkbutton(root, text="Tùy chọn 1",  
variable=check_var1)  
checkboxbutton2 = tk.Checkbutton(root, text="Tùy chọn 2",  
variable=check_var2)  
checkboxbutton1.pack()  
checkboxbutton2.pack()
```

trong đó:

- root: Cửa sổ cha (ví dụ: Tk() instance).
- text: Văn bản hiển thị trên Checkbutton.
- variable: Biến BooleanVar để lưu trạng thái (chọn hoặc không chọn).

* Thuộc tính của Checkbutton

- text: Văn bản hiển thị trên Checkbutton.
- variable: Biến BooleanVar để lưu trạng thái của Checkbutton.
- onvalue và offvalue: Giá trị khi Checkbutton được chọn và không chọn (mặc định là True và False).
- command: Hàm sẽ được gọi khi trạng thái của Checkbutton thay đổi.
- bg hoặc background: Màu nền của Checkbutton.
- fg hoặc foreground: Màu của văn bản trên Checkbutton.
- font: Font chữ của văn bản trên Checkbutton.

* Ví dụ

```
import tkinter as tk  
  
def show_selection():  
    selection = ""  
    if check_var1.get():  
        selection += "Tùy chọn 1 được chọn. "  
    if check_var2.get():  
        selection += "Tùy chọn 2 được chọn. "  
    label.config(text=selection)
```

```
root = tk.Tk()
root.title("Checkbutton Demo")

# Biến BooleanVar cho các Checkbutton
check_var1 = tk.BooleanVar()
check_var2 = tk.BooleanVar()

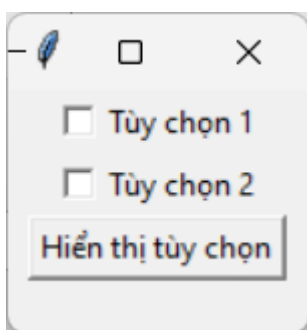
# Tạo các Checkbutton
checkbutton1 = tk.Checkbutton(root, text="Tùy chọn 1",
variable=check_var1)
checkbutton2 = tk.Checkbutton(root, text="Tùy chọn 2",
variable=check_var2)
checkbutton1.pack()
checkbutton2.pack()

# Tạo nút để hiển thị tùy chọn đã chọn
button = tk.Button(root, text="Hiển thị tùy chọn",
command=show_selection)
button.pack()

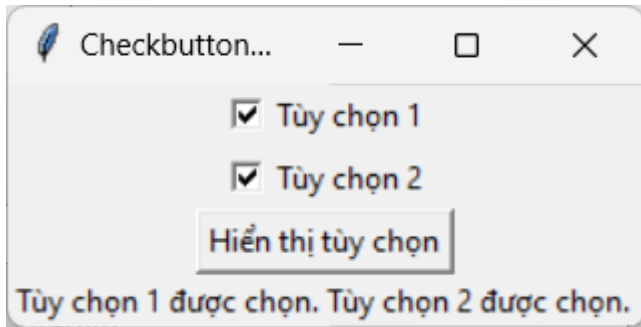
# Nhãn để hiển thị tùy chọn đã chọn
label = tk.Label(root, text="")
label.pack()

root.mainloop()
```

Kết quả



Nếu chọn và click vào button



2.2.5. Hộp chọn đa lựa chọn (Radiobutton)

Radiobutton là một thành phần cho phép người dùng chọn một trong số nhiều tùy chọn duy nhất. Khi một Radiobutton được chọn, tất cả các Radiobutton khác trong cùng nhóm sẽ tự động bị bỏ chọn.

* Tạo Radiobutton

```
radio_var = tk.IntVar()

radiobutton1 = tk.Radiobutton(root, text="Lựa chọn 1",
variable=radio_var, value=1)
radiobutton2 = tk.Radiobutton(root, text="Lựa chọn 2",
variable=radio_var, value=2)
radiobutton1.pack()
radiobutton2.pack()
```

trong đó:

- root: Cửa sổ cha (ví dụ: Tk() instance).
- text: Văn bản hiển thị trên Radiobutton.
- variable: Biến IntVar hoặc StringVar để lưu giá trị của Radiobutton.
- value: Giá trị tương ứng với Radiobutton (sẽ lưu vào biến variable khi được chọn).

* Thuộc tính của Radiobutton

- text: Văn bản hiển thị trên Radiobutton
- variable: Biến IntVar hoặc StringVar để lưu giá trị của Radiobutton.
- value: Giá trị tương ứng với Radiobutton.

- command: Hàm sẽ được gọi khi Radiobutton được chọn.
- bg hoặc background: Màu nền của Radiobutton.
- fg hoặc foreground: Màu của văn bản trên Radiobutton.
- font: Font chữ của văn bản trên Radiobutton.

*** Ví dụ**

```
import tkinter as tk

def show_selection():
    selected_option = radio_var.get()
    if selected_option == 1:
        label.config(text="Bạn chọn Lựa chọn 1")
    elif selected_option == 2:
        label.config(text="Bạn chọn Lựa chọn 2")

root = tk.Tk()
root.title("Radiobutton Demo")

# Biến IntVar để lưu trạng thái của Radiobutton
radio_var = tk.IntVar()

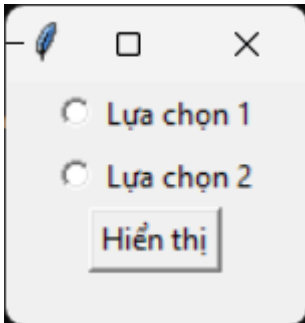
# Tạo các Radiobutton
radiobutton1 = tk.Radiobutton(root, text="Lựa chọn 1",
variable=radio_var, value=1)
radiobutton2 = tk.Radiobutton(root, text="Lựa chọn 2",
variable=radio_var, value=2)
radiobutton1.pack()
radiobutton2.pack()

# Tạo nút để hiển thị lựa chọn đã chọn
button = tk.Button(root, text="Hiển thị",
command=show_selection)
button.pack()
```

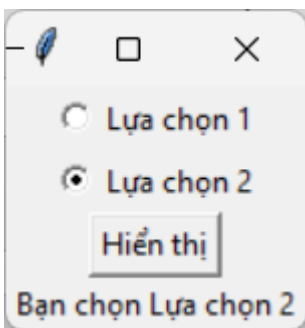
```
# Nhấn để hiển thị lựa chọn đã chọn
label = tk.Label(root, text="")
label.pack()

root.mainloop()
```

Kết quả:



Lựa chọn và click vào button



2.2.6. Danh sách chọn (Listbox)

Listbox là một thành phần cho phép người dùng chọn một hoặc nhiều mục từ một danh sách.

* Tạo Listbox

```
listbox = tk.Listbox(root, height=4,
selectmode=tk.MULTIPLE)
listbox.pack()
```

trong đó:

- root: Cửa sổ cha (ví dụ: Tk() instance).
- height: Chiều cao của Listbox (số mục hiển thị trên màn hình).

- selectmode: Chế độ chọn mục (tk.SINGLE - chọn một mục, tk.BROWSE - chọn một mục, tk.MULTIPLE - chọn nhiều mục).

* Các thuộc tính

- height: Chiều cao của Listbox (số mục hiển thị trên màn hình).
- selectmode: Chế độ chọn mục trong Listbox (tk.SINGLE, tk.BROWSE, tk.MULTIPLE, tk.EXTENDED).
- bg hoặc background: Màu nền của Listbox.
- fg hoặc foreground: Màu của văn bản trong Listbox.
- font: Font chữ của văn bản trong Listbox.
- width: Độ rộng của Listbox (số ký tự).
- activestyle: Kiểu hiển thị của mục được chọn (default, dotbox, underline, none).

* Thao tác với Listbox

- Thêm mục vào Listbox

```
listbox.insert(tk.END, "Mục 1")
listbox.insert(tk.END, "Mục 2")
```

- Lấy các mục đã chọn

```
selected_items = listbox.curselection()
for index in selected_items:
    print(listbox.get(index))
```

* Ví dụ

```
import tkinter as tk

def show_selected():
    selected_items = listbox.curselection()
    selection = ""
    for index in selected_items:
        selection += listbox.get(index) + "\n"
    label.config(text=selection)
```

```
root = tk.Tk()
root.title("Listbox Demo")

# Tạo Listbox với chiều cao 4 mục và chế độ chọn nhiều mục
listbox = tk.Listbox(root, height=4,
selectmode=tk.MULTIPLE)
listbox.pack(pady=10)

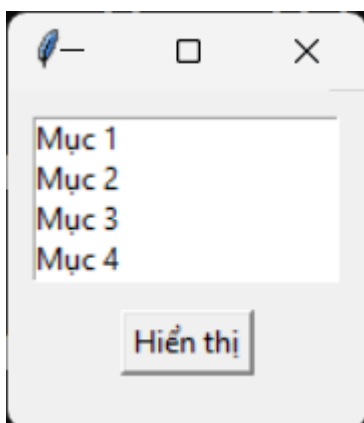
# Thêm mục vào Listbox
for item in ["Mục 1", "Mục 2", "Mục 3", "Mục 4"]:
    listbox.insert(tk.END, item)

# Tạo nút để hiển thị các mục đã chọn
button = tk.Button(root, text="Hiển thị",
command=show_selected)
button.pack()

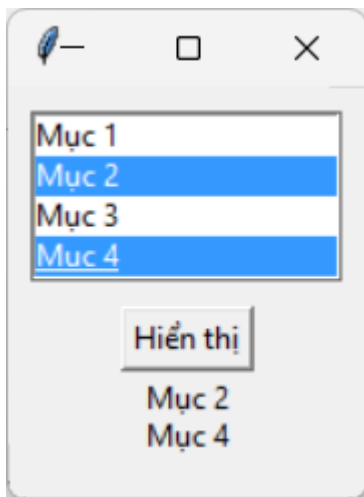
# Nhãn để hiển thị các mục đã chọn
label = tk.Label(root, text="")
label.pack()

root.mainloop()
```

Kết quả



Chọn vài mục vài click button



2.2.7. Ô văn bản (Text)

Text là một thành phần cho phép hiển thị và chỉnh sửa văn bản nhiều dòng. Nó cung cấp một vùng văn bản mà người dùng có thể nhập liệu, chỉnh sửa và xem nhiều dòng văn bản.

* Tạo Text:

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

trong đó:

- root: Cửa sổ cha (ví dụ: Tk() instance).
- height: Chiều cao của Text (số dòng).
- width: Độ rộng của Text (số ký tự).

* Các thuộc tính

- height và width: Chiều cao và độ rộng của Text.
- bg hoặc background: Màu nền của Text.
- fg hoặc foreground: Màu của văn bản trong Text.
- font: Font chữ của văn bản trong Text.
- wrap: Xác định cách Text xử lý khi văn bản vượt quá độ rộng của Text (none, char, word).
- insertbackground: Màu của dấu nháy khi nhập liệu vào Text.
- insertwidth và insertofftime: Độ rộng của dấu nháy khi nhập liệu vào Text và thời gian hiển thị/tắt của dấu nháy (đơn vị là milliseconds).

- state: Trạng thái của Text (normal, disabled).
- yscrollcommand và xscrollcommand: Sử dụng để kết nối Text với Scrollbar theo chiều dọc và chiều ngang.

* Ví dụ

```
import tkinter as tk

def get_text():
    text_content = text.get("1.0", tk.END)
    print("Nội dung văn bản:\n", text_content)

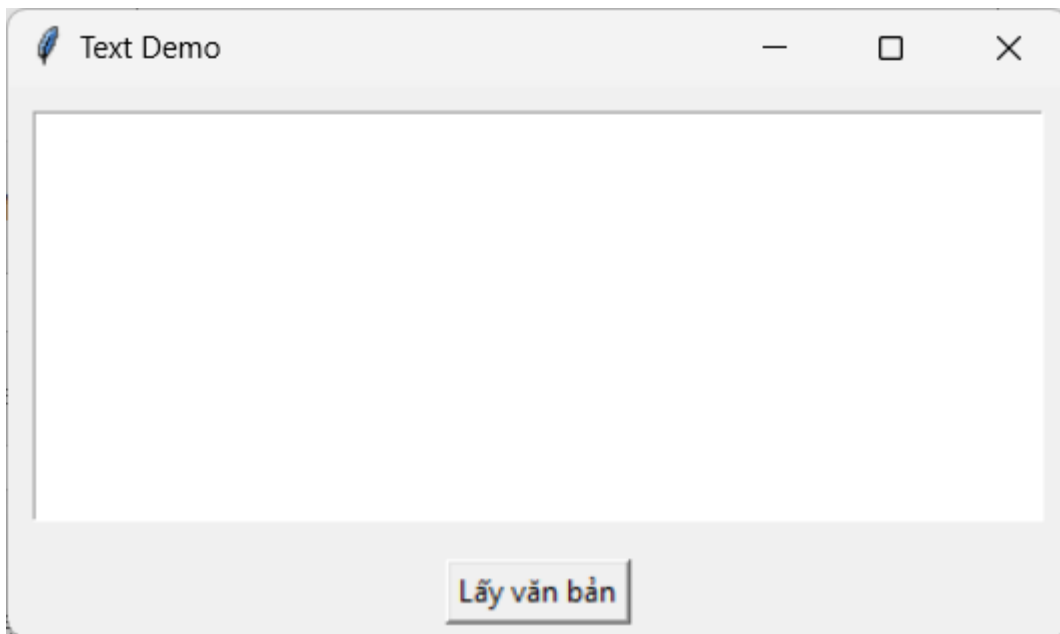
root = tk.Tk()
root.title("Text Demo")

# Tạo Text với chiều cao 10 dòng và chiều rộng 50 ký tự
text = tk.Text(root, height=10, width=50, wrap=tk.WORD)
text.pack(padx=10, pady=10)

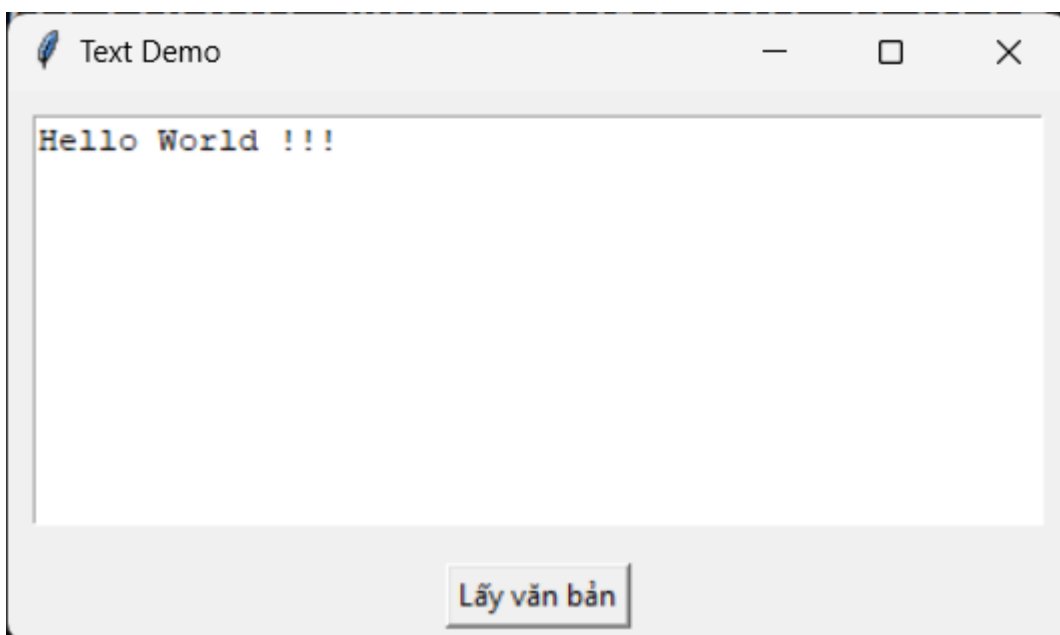
# Tạo nút để lấy nội dung văn bản từ Text
button = tk.Button(root, text="Lấy văn bản",
command=get_text)
button.pack(pady=5)

root.mainloop()
```

Kết quả



Khi nhập nội dung và click button



```
Nội dung văn bản:  
Hello World !!!
```

2.3. Các Message Box

Trong tkinter, để hiển thị các Message Box (hộp thoại thông báo) như thông báo cảnh báo, thông báo thông tin, xác nhận, bạn có thể sử dụng tkinter.messagebox module. Module này cung cấp các hàm để tạo và hiển thị các loại Message Box khác nhau.

2.3.1. *showinfo* - Hiển thị thông báo thông tin

```
import tkinter.messagebox as mb

mb.showinfo("Thông báo", "Đây là một thông báo thông tin!")
```

2.3.2. *showwarning* - Hiển thị thông báo cảnh báo

```
import tkinter.messagebox as mb

mb.showwarning("Cảnh báo", "Đây là một thông báo cảnh báo!")
```

2.3.3. *showerror* - Hiển thị thông báo lỗi

```
import tkinter.messagebox as mb

mb.showerror("Lỗi", "Đã xảy ra một lỗi!")
```

2.3.4. *askquestion* - Hiển thị thông báo xác nhận với lựa chọn Yes/No

```
import tkinter.messagebox as mb

response = mb.askquestion("Xác nhận", "Bạn có chắc chắn muốn tiếp tục?")
if response == "yes":
    print("Đã chọn Yes")
else:
    print("Đã chọn No")
```

2.3.5. *askyesno* - Hiển thị thông báo xác nhận với lựa chọn Yes/No

```
import tkinter.messagebox as mb

response = mb.askyesno("Xác nhận", "Bạn có chắc chắn muốn tiếp tục?")
if response:
    print("Đã chọn Yes")
else:
```

```
print("Đã chọn No")
```

2.3.6. askokcancel - Hiển thị thông báo xác nhận với lựa chọn OK/Cancel

```
import tkinter.messagebox as mb

response = mb.askokcancel("Xác nhận", "Bạn có chắc chắn  
muốn tiếp tục?")
if response:
    print("Đã chọn OK")
else:
    print("Đã chọn Cancel")
```

2.3.7. askretrycancel - Hiển thị thông báo xác nhận với lựa chọn Retry/Cancel

```
import tkinter.messagebox as mb

response = mb.askretrycancel("Xác nhận", "Đã xảy ra một  
lỗi. Bạn có muốn thử lại?")
if response:
    print("Đã chọn Retry")
else:
    print("Đã chọn Cancel")
```

3. Bài tập

Bài 1. Viết và cài đặt lại những đoạn code trong bài học này.

Bài 2. Viết chương trình python thực hiện chức năng sau:

- Tạo một cửa sổ chính với tiêu đề là "Hello Tkinter".
- Tạo một nhãn với nội dung "Chào mừng đến với tkinter!" và một nút với văn bản "Nhấn vào đây".
- Khi nhấn vào nút, hiển thị một Message Box thông báo với nội dung "Bạn đã nhấn vào nút".

Bài 3. Viết chương trình python thực hiện chức năng sau:

- Tạo một giao diện có chứa ô nhập liệu, nút "Submit", và một khu vực hiển thị kết quả. Khi người dùng nhập dữ liệu vào ô nhập liệu và nhấn "Submit", hiển thị thông báo "Bạn đã nhập: [nội dung nhập vào]" ở khu vực hiển thị kết quả.

Bài 4. Tạo một form đăng ký tài khoản người dùng bao gồm các trường:

- Họ và tên
- Ngày tháng năm sinh
- Email
- Password và Nhập lại password. Khi người dùng nhập vào

Tạo một nút Đăng ký. Khi người dùng click vào nút đăng ký. Thực hiện:

- Kiểm tra người dùng đã nhập đầy đủ thông tin. Nếu chưa, xuất thông báo yêu cầu nhập đầy đủ thông tin.
- Kiểm tra email đúng định dạng. Nếu không, xuất thông báo lỗi.
- Kiểm tra password và nhập lại password giống nhau. Nếu không, xuất thông báo lỗi.
- Kiểm tra người dùng từ 18 tuổi. Nếu không, xuất thông báo lỗi.
- Kiểm tra email đã có trong file user.json chưa. Nếu đã có, xuất thông báo lỗi Email đã tồn tại.
- Nếu mọi thông tin đều chính xác, thêm thông tin người dùng vào file user.json. Lưu ý, password phải được mã hóa trước khi ghi vào file.

Chú ý: Nếu file user.json không tồn tại, tạo file mới và ghi thông tin vào.

Bài 5. Tạo ứng dụng python với tkinter thực hiện chức năng:

- Hiện thị form đăng nhập gồm email và password.
- Người dùng nhập thông tin và nhấn Đăng nhập.

Kiểm tra thông tin đăng nhập bằng cách đọc và kiểm tra trong file user.json (của Bài 4). Nếu thành công, xuất hộp thông báo Đăng nhập thành công. Nếu không, xuất ra Lỗi: Đăng nhập thất bại.