TRƯỜNG ĐẠI HỌC KINH TẾ - KỸ THUẬT CÔNG NGHIỆP
KHOA: KHOA HỌC ỨNG DỤNG.

CHƯƠNG 7

LẬP TRÌNH GIAO DIỆN GUI

Giảng viên: TS.Cao Diệp Thắng

Khoa: Khoa học Ứng dụng

NỘI DUNG BÀI HỌC

7.1 **LẬP TRÌNH GUI VỚI Tkinter**

7.2 **GIỚI THIỆU LẬP TRÌNH GUI VỚI PyQt**

MỤC TIÊU BÀI HỌC

Sau khi học xong bài này, người học sẽ nắm được các vấn đề sau:

+ Cú pháp và biết sử dụng các tính năng cơ bản của tkinter, bao gồm tạo cửa sổ, widget, xử lý sự kiện, và trình bày giao diện người dùng.
+ Hiểu về các nguyên lý thiết kế giao diện người dùng, bố cục, màu sắc, font chữ và các yếu tố giao diện khác để tạo giao diện hấp dẫn và dễ sử dụng.
+ Biết cách xử lý sự kiện từ người dùng, bao gồm nhấn nút, thay đổi giá trị trong các trường nhập liệu, hoặc chọn từ các menu.
+ Hiểu về trạng thái của các widget và cách điều khiển và cập nhật chúng theo yêu cầu.
+ Nắm vững các phương thức layout của tkinter để điều chỉnh vị trí và kích thước của các widget trong giao diện.
+ Biết cách hiển thị hình ảnh, âm thanh và video trong ứng dụng tkinter và làm việc với tệp tin đa phương tiện.
+ Tích hợp với các thư viện và chức năng bên ngoài: Hiểu cách tích hợp tkinter với các thư viện và chức năng bên ngoài khác của Python để làm việc với các tính năng nâng cao.

7.1. LẬP TRÌNH GUI VỚI TKINTER

7.1.1 Giới thiệu

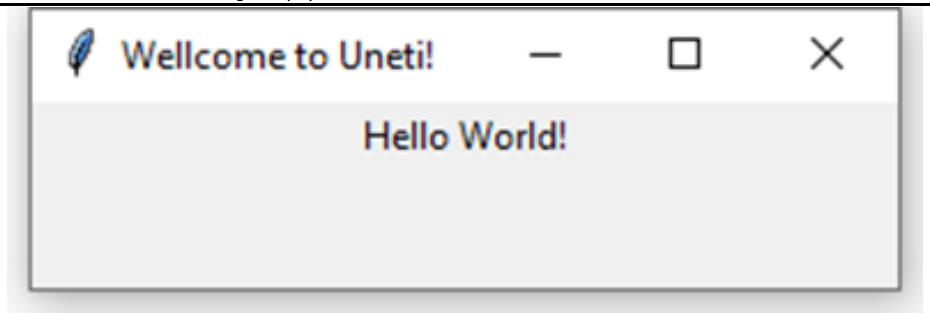
Tkinter là một thư viện đồ họa GUI mặc định của Python. Nó cung cấp các công cụ và widget để xây dựng các ứng dụng GUI. Tkinter cung cấp giao diện hướng đối tượng cho bộ công cụ Tk GUI.[14, 15, 16]

Các bước cơ bản để tạo một ứng dụng Tkinter:

1. import Tkinter module.
2. Tạo cửa sổ chính của ứng dụng GUI.
3. Thêm một hoặc nhiều widget nói trên vào ứng dụng GUI.
4. Gọi vòng lặp chạy ứng dụng để các hành động có thể diễn ra trên màn hình máy tính của người dùng.

Ví dụ 7.1.1. Chương trình giao diện GUI đầu tiên.

```
1  import tkinter as tk
2
3  # Tạo một cửa sổ chính của ứng dụng
4  window = tk.Tk()
5
6  # Thêm các thành phần widget  vào cửa sổ
7  label = tk.Label(window, text="Hello World!")
8  label.pack()
9
10 #Thêm tiêu đề cho cửa  sổ
11 window.title("Wellcome to Uneti!")
12 #Đặt kích thước cửa sổ theo pixels
13 window.geometry('280x60')
14
15 # Bắt đầu vòng lặp chạy ứng dụng (lặp vô tận để hiển thị cửa sổ trên màn hình máy tính)
16 window.mainloop()
```



7.1.2. Các thành phần cơ bản (widgets) của Tkinter python.

a. **Khái niệm widgets:** Widget (còn được gọi là thành phần hoặc điều khiển) là một thành phần cơ bản trong việc xây dựng giao diện người dùng trong một ứng dụng. **Widgets** là một đối tượng đồ họa được sử dụng để hiển thị thông tin và tương tác với người dùng.[15,16]

Trong Tkinter có 2 loại widget:

- **Loại 1**: Các Widget đại diện cho các phần tử như nút (**button**), ô nhập liệu (**entry**), nhãn (**label**), danh sách (**list**), v.v.

  Mỗi Widget có tính năng, ngoại hình và hành vi riêng, và chúng có thể được sắp xếp và tương tác với nhau để tạo thành giao diện người dùng hoàn chỉnh. Các Widget thường có các thuộc tính để điều chỉnh ngoại hình, như màu sắc, kích thước, vị trí và kiểu chữ. Ngoài ra, chúng cũng có các phương thức để xử lý sự kiện, như khi người dùng nhấn nút hoặc thay đổi giá trị trong ô nhập liệu.

- **Loại 2**: **container widget**. Đây là các widget có khả năng chứa các widget khác bên trong chúng. Container widget giúp tổ chức và quản lý các widget con trong một cấu trúc hỗn hợp.

  Các Container widget phổ biến bao gồm **Frame** (khung), **LabelFrame** (khung nhãn), **Toplevel** (cửa sổ con), **Canvas** (bảng vẽ), ….

Các container Widget còn được gọi là layout.

Trong tkinter có 03 loại layout là:

- **Pack Layout**: Phương thức pack() cho phép người dùng tự động xếp chồng các widget dọc hoặc ngang, phụ thuộc vào hướng họ chọn. Widget sẽ được thêm vào trong một container và có thể căn chỉnh, lấp đầy, hoặc mở rộng kích thước của container.
- **Grid Layout**: Phương thức grid() cho phép người dùng xây dựng giao diện bằng cách đặt các widget vào các ô lưới của một bảng. Chúng ta có thể quy định vị trí của mỗi widget bằng cách chỉ định số hàng và số cột của ô lưới mà nó nằm trong.
- **Place Layout**: Phương thức place() cho phép người dùng đặt các widget tại các tọa độ tùy ý trên cửa sổ hoặc khung chứa. Chúng ta có thể chỉ định vị trí và kích thước của mỗi widget bằng cách sử dụng các tham số x và y.

Widget đóng vai trò quan trọng trong việc tạo ra giao diện người dùng tương tác trong các ứng dụng và cho phép người dùng tương tác với các chức năng và dữ liệu trong ứng dụng.

b. Các Widget trong Tkinter [14]

Bảng 7.1. Một số widget trong Tkinter

TT	Tên (widgets)	Mô tả
1	Label	Label là một văn bản được sử dụng để hiển thị một số thông báo hoặc thông tin cho các widget khác.
2	Button	Button được sử dụng để thêm nhiều nút khác nhau vào ứng dụng python.
3	Canvas	Canvas được sử dụng để vẽ các hình trên cửa sổ.
4	Checkbutton	Checkbutton được sử dụng để hiển thị CheckButton trên cửa sổ.
5	Entry	Entry được sử dụng để hiển thị trường văn bản một dòng cho người dùng. Nó thường được sử dụng để nhập các giá trị của người dùng.
6	Frame	Frame có thể được định nghĩa là một vùng chứa mà có thể chứa một hoặc nhiều widget khác.
7	ListBox	ListBox được sử dụng để hiển thị danh sách các tùy chọn cho người dùng.
8	Menubutton	Menubutton được sử dụng để hiển thị các mục menu cho người dùng.
9	Menu	Menu được sử dụng để thêm các mục menu cho người dùng.
10	Message	Message được sử dụng để hiển thị hộp tin nhắn cho người dùng.
11	Radiobutton	Người dùng được cung cấp các tùy chọn khác nhau và người dùng chỉ có thể chọn một tùy chọn trong số đó.
12	Scale	Nó được sử dụng để cung cấp thanh trượt cho người dùng.
13	Scrollbar	Nó cung cấp thanh cuộn cho người dùng để người dùng có thể cuộn cửa sổ lên và xuống.
14	Text	Nó khác với Entry vì nó cung cấp một trường văn bản nhiều dòng cho người dùng để người dùng có thể viết văn bản và chỉnh sửa văn bản bên trong nó.
15	Toplevel	Nó được sử dụng để tạo một vùng chứa cửa sổ riêng biệt.
16	Spinbox	Một widget mục nhập được sử dụng để chọn từ các tùy chọn của các giá trị.
17	PanedWindow	Nó giống như một widget container chứa các ô ngang hoặc dọc.
18	LabelFrame	LabelFrame là một widget vùng chứa hoạt động như một container.
19	MessageBox	Được sử dụng để hiển thị hộp thông báo trong các ứng dụng desktop.

c. Làm việc với các thành phần trong Tkinter

Label Đối tượng này dùng để hiển thị văn bản hoặc hình ảnh

Cú pháp `w = Label( master, option, … )`

danh sách một số tùy chọn

Tên của cửa sổ chứa nhãn.

Bảng 7.2. Danh sách một số tùy chọn khi sử dụng Label (xem TLHT)

Ví dụ 7.1.2.1. Tạo một cửa sổ gồm một nhãn (label) có nội dung là "Hello" và font chữ "Arial Bold" cỡ chữ 50, trong giao diện người dùng.

```
1   from tkinter import *
2   window = Tk()
3   window.title("Welcome to Uneti.")
4
5   #Thêm label có nội dung Hello, font chữ "Arial Bold, cỡ chữ 50"
6   lbl = Label(window, text="Hello", font=("Arial Bold", 50))
7
8   #Đặt kích thước cửa sổ theo pixels
9   window.geometry('280x80')
10
11  #Xác định vị trí của label
12  lbl.grid(column=0, row=0)
13  # Bắt đầu vòng lặp chạy ứng dụng (lặp vô tận
14  để hiển thị cửa sổ trên màn hình máy tính)
15  window.mainloop()
16
```

Button : nút bấm để có thể hiển thị văn bản hoặc hình ảnh, lập trình viên có thể thiết lập sự kiện khi người dùng click vào nút bấm chẳng hạn như tự động gọi hàm.

Cú pháp:

```
w = Button( master, option = value, ... )
```

danh sách một số tùy chọn (Bảng 7.3)

Tên của cửa sổ chứa nhãn.

Phương thức

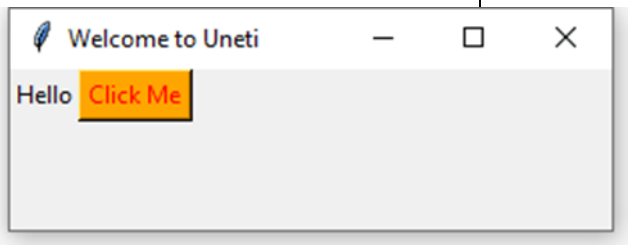
flash(): Phương thức flash() trong Tkinter được sử dụng để tạo hiệu ứng nhấp nháy trên một nút (button). Khi gọi phương thức flash() trên một đối tượng nút, nút sẽ nhấp nháy một cách nhanh chóng giữa hai trạng thái màu sắc, tạo hiệu ứng nhấp nháy. Để nút ở trạng thái ban đầu. Bỏ qua nếu nút bị tắt.

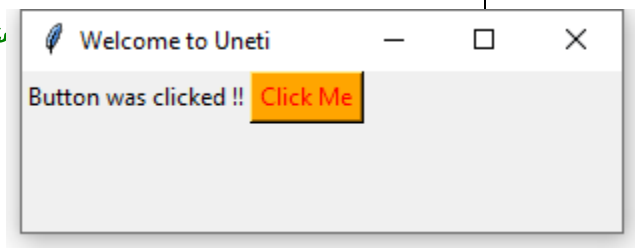
invoke(): Phương thức invoke() được sử dụng để gọi một hàm hoặc phương thức liên kết với một nút (button) khi nút được nhấn. Khi người dùng nhấn vào nút, phương thức invoke() sẽ tự động gọi hàm hoặc phương thức được liên kết, thực thi các lệnh tương ứng.. Không có tác dụng nếu nút bị tắt hoặc không có lệnh gọi lại.

Ví dụ 7.1.2.2. Tạo một cửa sổ gồm một nhãn (label) và một nút nhấn (button). Khi nút nhấn được nhấn, nhãn (label) sẽ được cập nhật với nội dung mới.

```
1  from tkinter import *
2
3  window = Tk()
4  window.title("Welcome to Uneti")
5  window.geometry('300x80')
6  lbl = Label(window, text="Hello")
7  lbl.grid(column=0, row=0)
8
9  #Hàm khi nút được nhấn
10 def clicked():
11     lbl.configure(text="Button was clicked !!")
12 #Thêm một nút nhấn Click Me
13 btn = Button(window, text="Click Me", bg="orange",
14                         fg="red",command=clicked)
15
16 #Thiết lập vị trí của nút nhấn có màu nền và màu chữ
17 btn.grid(column=5, row=0)
18
19 window.mainloop()
```

Kết quả thực hiện





Textbox

Textbox: (hay còn gọi là Text widget) là một widget dùng để hiển thị và cho phép người dùng nhập liệu văn bản dạng nhiều dòng. Nó là một hộp văn bản lớn, thích hợp để hiển thị và chỉnh sửa văn bản dài hoặc dạng định dạng.

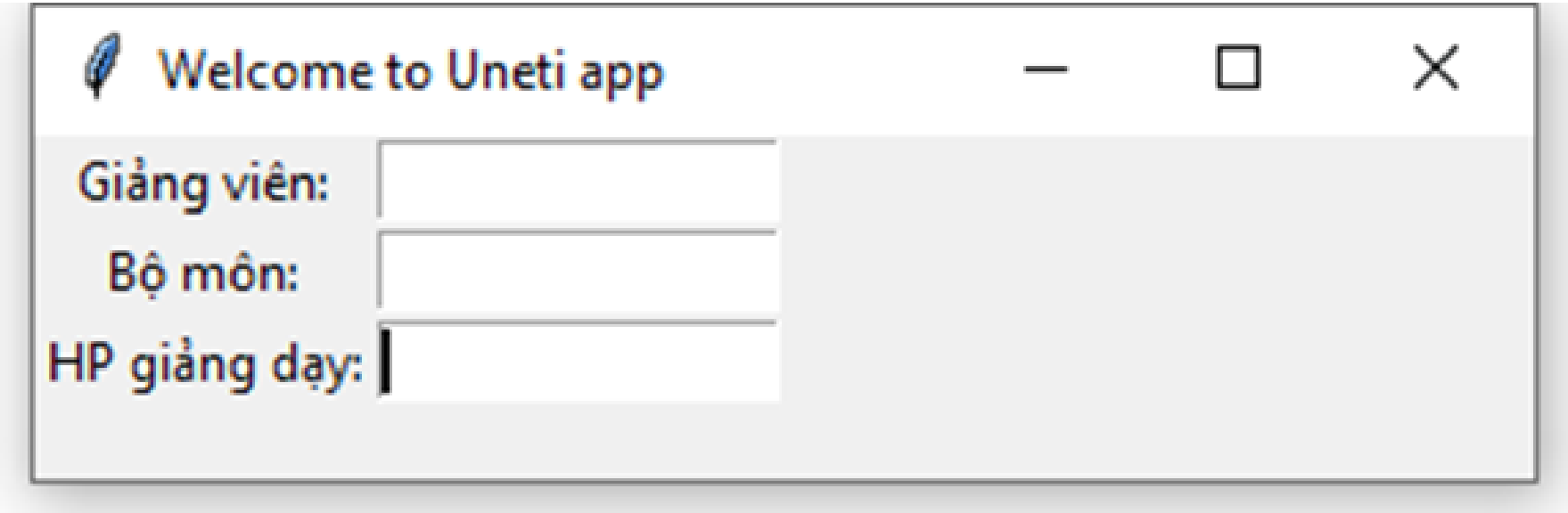
Textbox cung cấp các tính năng cho phép người dùng:

- **Hiển thị và chỉnh sửa văn bản dạng nhiều dòng**: chỉnh sửa, xóa và chèn văn bản vào Textbox. Văn bản có thể chia thành nhiều dòng, tùy thuộc vào kích thước của Textbox.
- **Cuộn nội dung:** Nếu văn bản vượt quá kích thước của Textbox, một thanh cuộn sẽ xuất hiện cho phép người dùng cuộn lên xuống để xem và chỉnh sửa phần nội dung không nhìn thấy.
- **Định dạng văn bản**: Textbox hỗ trợ định dạng văn bản, cho phép người dùng chọn *phông chữ*, *kích thước*, *màu sắc*, *in đậm, in nghiêng, gạch chân, gạch ngang* và nhiều thuộc tính văn bản khác.
- **Xử lý sự kiện**: Người dùng có thể gắn các hàm xử lý sự kiện vào Textbox, cho phép thực hiện các hành động khi người dùng thực hiện các tương tác như nhấp chuột, nhấn phím, chọn văn bản, vv.

Ví dụ 7.1.2.3. Tạo Textbox trong tkinter:

```
import tkinter as tk

class UnetiApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Welcome to Uneti app")
        self.root.geometry('350x80')

        self.lbl1 = tk.Label(root, text="Giảng viên:")
        self.lbl1.grid(column=0, row=0)

        self.txt1 = tk.Entry(root, width=15)
        self.txt1.grid(column=1, row=0)
        self.txt1.focus()

        self.lbl2 = tk.Label(root, text="Bộ môn:")
        self.lbl2.grid(column=0, row=2)

        self.txt2 = tk.Entry(root, width=15)
        self.txt2.grid(column=1, row=2)
        self.txt2.focus()

        self.lbl3 = tk.Label(root, text="HP giảng dạy:")
        self.lbl3.grid(column=0, row=3)

        self.txt3 = tk.Entry(root, width=15)
        self.txt3.grid(column=1, row=3)
        self.txt3.focus()

if __name__ == "__main__":
    root = tk.Tk()
    app = UnetiApp(root)
    root.mainloop()
```

Kết quả thực hiện chương trình



Combobox

Combobox: là một widget cho phép người dùng chọn giá trị từ một danh sách các tùy chọn. Nó kết hợp giữa tính năng của Entry (một ô nhập liệu văn bản) và Listbox (một danh sách các tùy chọn) để tạo ra một hộp chọn tùy chọn mở rộng.

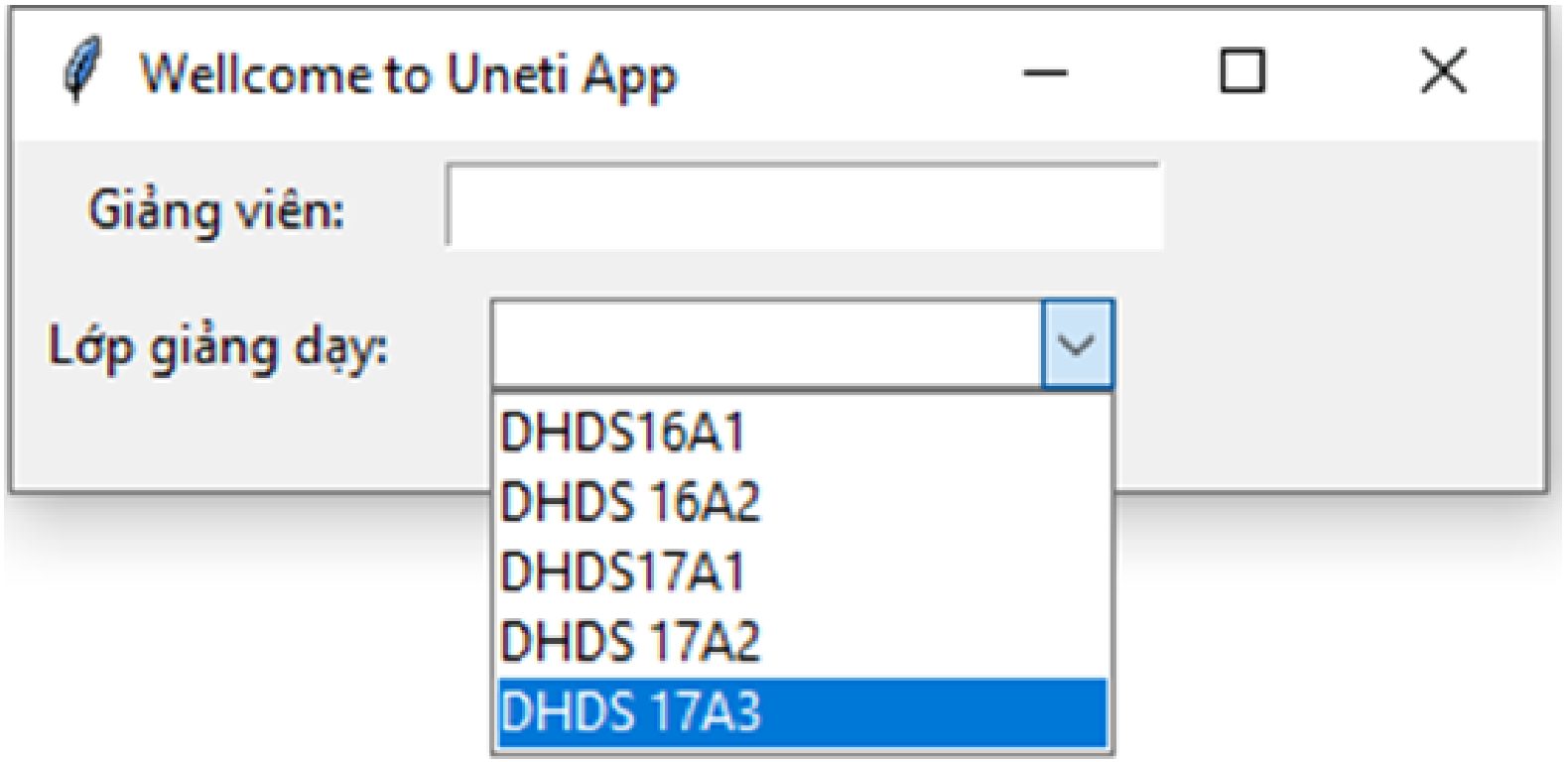
Tính năng

- Hiển thị tùy chọn: Một giá trị mặc định được hiển thị trong Combobox. Khi người dùng nhấp vào Combobox, danh sách tùy chọn sẽ xuất hiện để người dùng có thể chọn từ các tùy chọn khác.
- Chọn giá trị: Người dùng có thể chọn một giá trị từ danh sách tùy chọn bằng cách nhấp vào nó.
- Tùy chỉnh danh sách tùy chọn: Bạn có thể thiết lập danh sách các tùy chọn mà Combobox hiển thị bằng cách cung cấp dữ liệu là một danh sách hoặc tuple.
- Xử lý sự kiện: Người dùng có thể gắn các hàm xử lý sự kiện vào Combobox, cho phép thực hiện các hành động khi người dùng thực hiện các tương tác như chọn tùy chọn khác nhau.

Ví dụ 7.1.2.4. Sử dụng lập trình hướng đối tượng để tạo Textbox và combobox.

```
import tkinter as tk
from tkinter import ttk

class ComboboxApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Wellcome to Uneti App")
        self.root.geometry('350x80')

        self.options=["DHDS16A1", "DHDS 16A2", "DHDS17A1", "DHDS 17A2", "DHDS 17A3"]

        self.teacher_label = tk.Label(root, text="Giảng viên:")
        self.teacher_label.grid(row=0, column=0, padx=5, pady=5)

        self.teacher_textbox = tk.Text(root, height=1, width=20)
        self.teacher_textbox.grid(row=0, column=1, padx=5, pady=5)

        self.label = tk.Label(root, text="Lớp giảng dạy:")
        self.label.grid(row=1, column=0, padx=5, pady=5)

        self.combobox = ttk.Combobox(root, values=self.options)
        self.combobox.grid(row=1, column=1, padx=5, pady=5)

        self.combobox.bind("<<ComboboxSelected>>", self.on_select)

    def on_select(self, event):
        selected_value = self.combobox.get()
        print("Selected:", selected_value)

root = tk.Tk()
app = ComboboxApp(root)
root.mainloop()
```

Kết quả thực hiện chương trình:



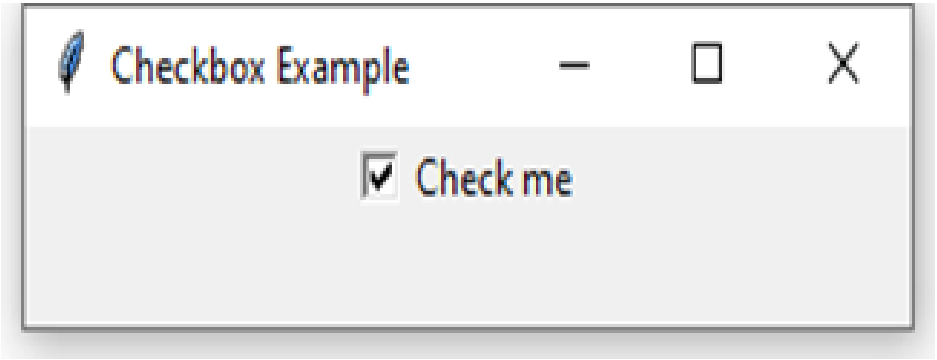
Checkbox

Checkbox (hộp kiểm) là một loại widget trong tkinter được sử dụng để cho phép người dùng chọn hoặc bỏ chọn một tùy chọn. Dạng hình vuông nhỏ có thể được đánh dấu (checked) hoặc không được đánh dấu (unchecked). Giá trị của nó được đặt thành True; khi không được chọn, giá trị của nó được đặt thành False.

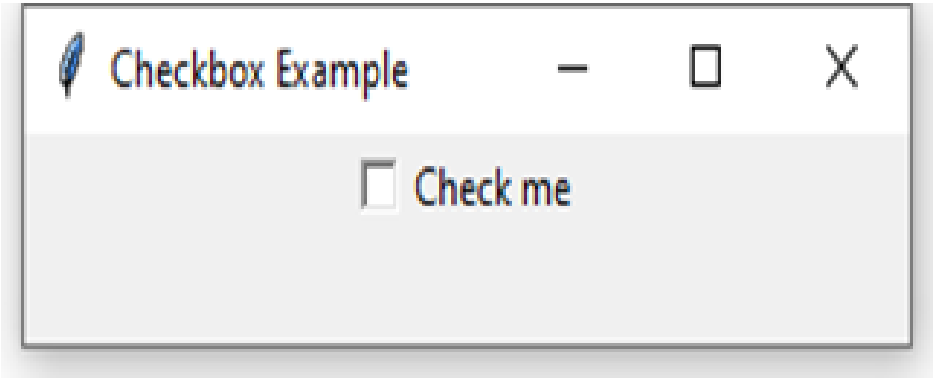
Ví dụ 7.1.2.5. Sử dụng Checkbox.

```
1  import tkinter as tk
2
3  class CheckboxApp:
4      def __init__(self, root):
5          self.root = root
6          self.root.title("Checkbox Example")
7          self.root.geometry('300x50')
8
9          self.var = tk.BooleanVar()
10         self.checkbox = tk.Checkbutton(root, text="Check me",
11                                        variable=self.var,
12                                        command=self.on_checkbox_click)
13         self.checkbox.pack()
14
15     def on_checkbox_click(self):
16         if self.var.get():
17             print("Checkbox is checked.")
18         else:
19             print("Checkbox is unchecked.")
20
21 if __name__ == "__main__":
22     root = tk.Tk()
23     app = CheckboxApp(root)
24     root.mainloop()
```

Checkbox is checked.



Checkbox is unchecked.



Radio button

Radio button là một loại widget trong tkinter cho phép người dùng chọn một trong nhiều tùy chọn đơn từ một danh sách. Khi một radio button được chọn, các radio button khác trong cùng nhóm sẽ tự động bỏ chọn

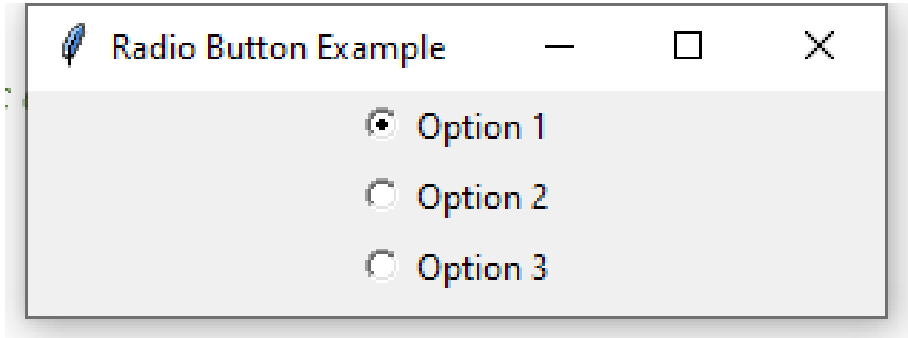
Ví dụ 7.1.2.6. Sử dụng Radio button

```
1   import tkter as tk
2
3   class RadioButtonExample:
4       def __init__(self, root):
5           self.root = root
6           self.root.title("Radio Button Example")
7           self.root.geometry('300x80')
8
9           # Biến kiểu IntVar để lưu giá trị radio button được chọn
            self.selected_option = tk.IntVar()
```

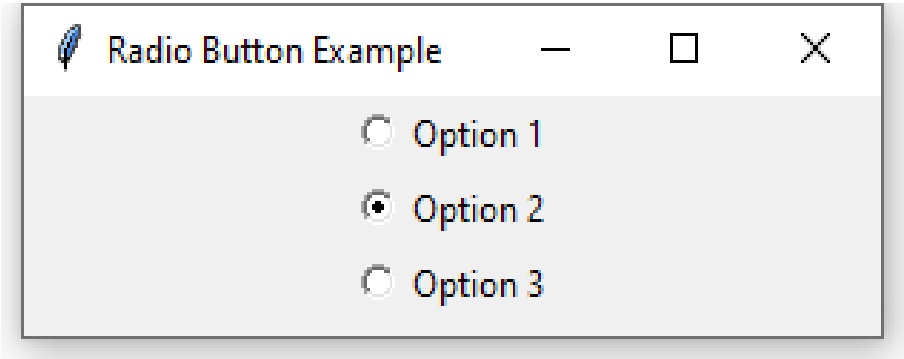
```
    # Tạo các radio button và liên kết với biến selected_option
            self.radio1 = tk.Radiobutton(self.root, text="Option 1",
                                         variable=self.selected_option,value=1,
                                         command=self.on_radio_select)
            self.radio2 = tk.Radiobutton(self.root, text="Option 2",
                                         variable=self.selected_option, value=2,
                                         command=self.on_radio_select)
            self.radio3 = tk.Radiobutton(self.root, text="Option 3",
                                         variable=self.selected_option, value=3,
                                         command=self.on_radio_select)

            # Đặt các radio button vào cửa sổ giao diện
            self.radio1.pack()
            self.radio2.pack()
            self.radio3.pack()

        def on_radio_select(self):
            selected_value = self.selected_option.get()
            print(f"Option {selected_value} is selected.")

    if __name__ == "__main__":
        root = tk.Tk()
        app = RadioButtonExample(root)
        root.mainloop()
```

Kết quả thực hiện chương trình:

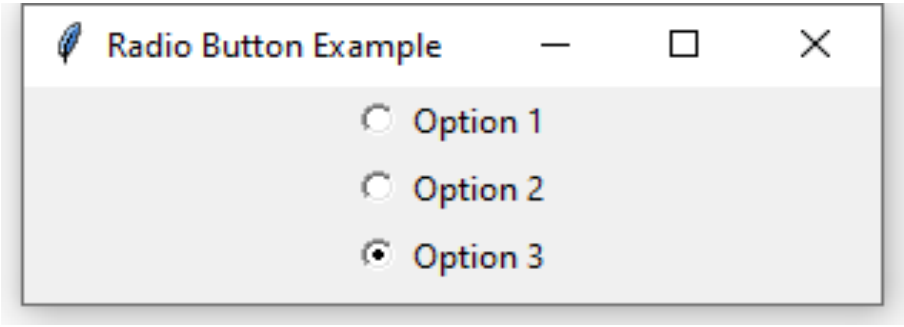
```
Option 1 is selected.
```




```
Option 2 is selected.
```




```
Option 3 is selected.
```



ScrolledText

ScrolledText là một widget được sử dụng để hiển thị văn bản có khả năng cuộn. Nó kết hợp giữa một widget văn bản thông thường và một thanh cuộn, cho phép người dùng xem và chỉnh sửa văn bản dài hơn mà không cần phải điều chỉnh kích thước của cửa sổ hay trình cuộn.

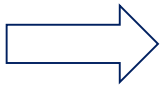
Điểm nổi bật của ScrolledText là khả năng cuộn văn bản tự động khi nội dung vượt quá kích thước hiển thị. Người dùng có thể cuộn nội dung bằng cách sử dụng thanh cuộn hoặc các phím mũi tên.

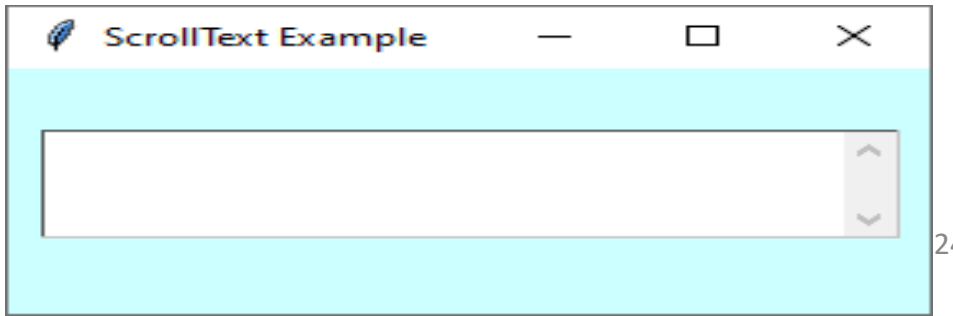
Ví dụ 7.2.1.7.

```
1  import tkinter as tk
2
3  class TextEditorApp:
4      def __init__(self, win):
5          self.win = win #Tạo một cửa sổ chính (main window) và lưu vào biến win.
6           self.win.title("ScrollText Example")
7          self.win.geometry('280x120')
8           #Đặt màu nền của cửa sổ là màu xanh lam (#CCFFFF).
9           self.win.configure(background="#CCFFFF")
10
11         self.frame1 = tk.Frame(self.win, width=100, height=40,
12                                bg='#BB0000', borderwidth=1, relief='sunken')
13         #Tạo một thanh cuộn (scrollbar) và đặt nó trong khung frame1.
14         self.scrollbar = tk.Scrollbar(self.frame1)
```

```
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
```

```
        #Tạo một vùng văn bản (Text) có chiều rộng là 30, chiều cao là 3 và có thể tự động
        # xuống dòng (wrap="word").
            self.editArea = tk.Text(self.frame1, width=30, height=3,
            wrap="word", yscrollcommand=self.scrollbar.set,
                                        borderwidth=0, highlightthickness=0)

    #Thiết lập thanh cuộn để cuộn theo dọc khi vùng văn bản được cuộn.
                self.scrollbar.config(command=self.editArea.yview)
                #Đặt thanh cuộn ở phía bên phải của khung và điền cả chiều dọc của khung.
                self.scrollbar.pack(side="right", fill="y")
                #Đặt vùng văn bản ở phía bên trái của khung và lấp đầy cả chiều ngang và
                # chiều dọc của khung.
               self.editArea.pack(side="left", fill="both", expand=True)
                #Đặt khung frame1 ở tọa độ (x=10, y=30) trong cửa sổ chính.
                self.frame1.place(x=10, y=30)

            def run(self):
                self.win.mainloop()#Vòng lặp chính của ứng dụng, giúp cửa sổ hiển thị &
                                            #duy trì liên tục cho đến khi đóng cửa sổ.

    if __name__ == "__main__":
        win=tk.Tk()
        app = TextEditorApp(win)
        app.run()
```





Listbox

ListBox trong tkinter là một loại widget cho phép hiển thị một danh sách các mục cho người dùng lựa chọn. Điều này rất hữu ích khi muốn cho phép người dùng chọn một hoặc nhiều mục từ danh sách được cung cấp.

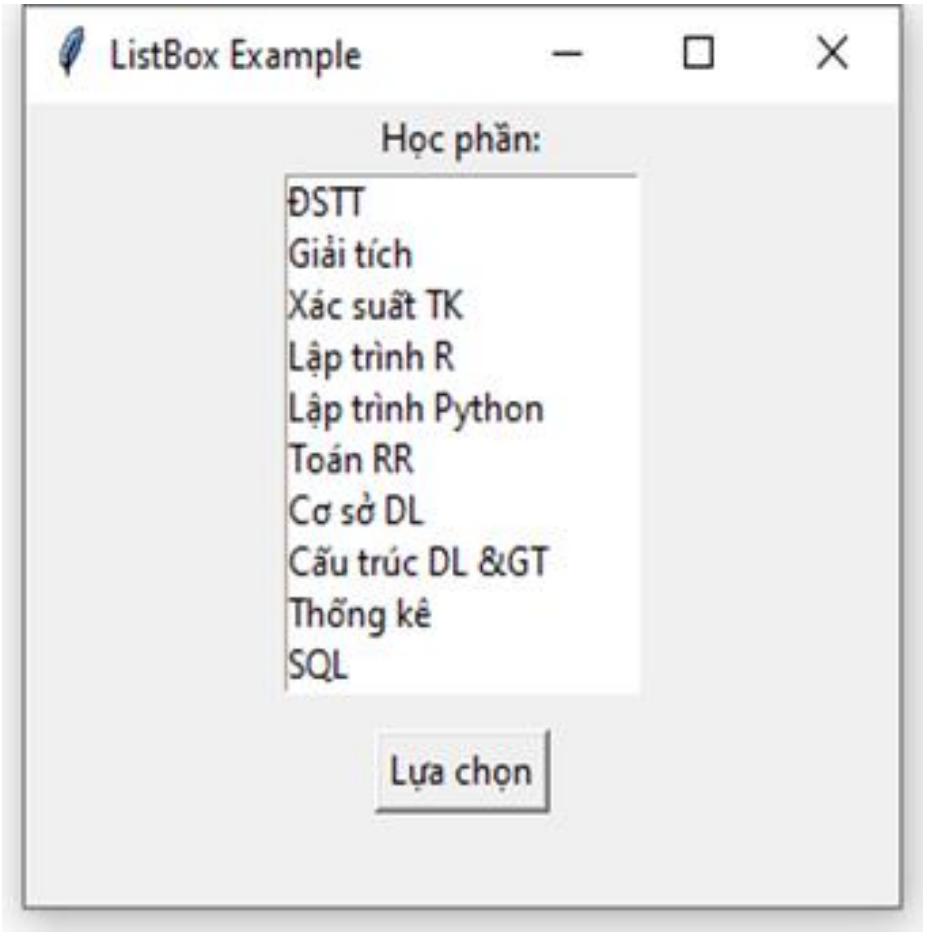
Một số phương thức quan trọng của ListBox:

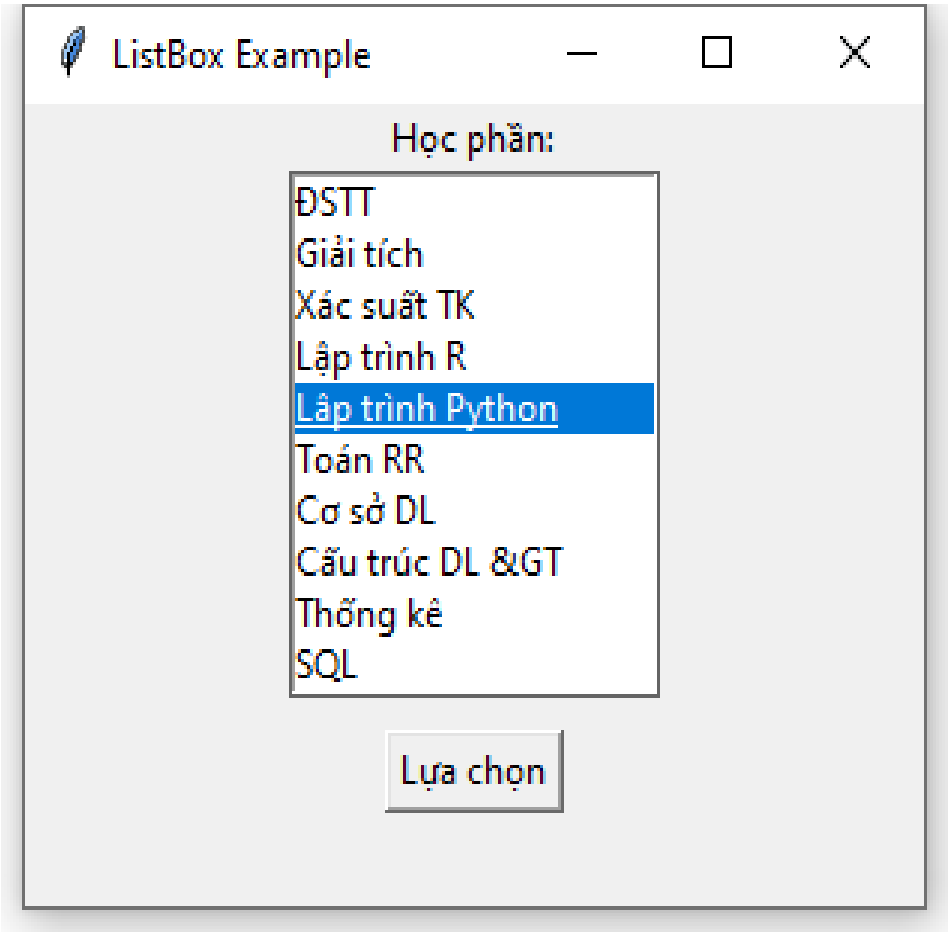
- **insert(index, *elements):** Thêm các mục vào ListBox tại vị trí chỉ định (index). Nếu không cung cấp index, các mục sẽ được thêm vào cuối danh sách.
- **delete(first, last=None):** Xóa các mục từ ListBox. Nếu chỉ cung cấp first, chỉ mục này sẽ bị xóa. Nếu cung cấp cả first và last, các mục từ first đến last sẽ bị xóa.
- **get(index):** Lấy giá trị của mục ở vị trí index.
- **curselection():** Trả về một tuple chứa chỉ mục của các mục đã chọn trong ListBox.
- **select_set(index):** Đánh dấu mục ở vị trí index là mục đã chọn.
- **select_clear(index):** Bỏ chọn mục ở vị trí index.

Ví dụ 7.1.2.8. Sử dụng ListBox.

```
1  import tkinter as tk
2  class ListBoxExample:
3      def __init__(self, root):
4          self.root = root
5          self.root.title("ListBox Example")
6          self.root.geometry('300x250')
7          self.create_widgets()
8
9      def create_widgets(self):
10         self.lbl = tk.Label(root, text="Học phần:")
11         self.lbl.pack()
12
13         hoc_phan=['ĐSTT','Giải tích','Xác suất TK', 'Lập trình R',
14                   'Lập trình Python', 'Toán RR', 'Cơ sở DL',
15                   'Cấu trúc DL &GT', 'Thống kê', 'SQL']
16          self.listbox = tk.Listbox(self.root, selectmode=tk.MULTIPL
17
```

```
        self.listbox.pack()

        for i in hoc_phan:
            self.listbox.insert(tk.END, f"{i}")

        self.select_button = tk.Button(self.root, text="Select",
command=self.on_select)
        self.select_button.pack(pady=10)

    def on_select(self):
        selected_items = [self.listbox.get(index) for index in
self.listbox.curselection()]
        print("Selected Items:", selected_items)

if __name__ == "__main__":
    root = tk.Tk()
    app = ListBoxExample(root)
    root.mainloop()
```





Khi người dùng click vào Select, trên màn hinh in ra dòng thông báo:

`Học phần được chọn là: ['Lập trình Python']`

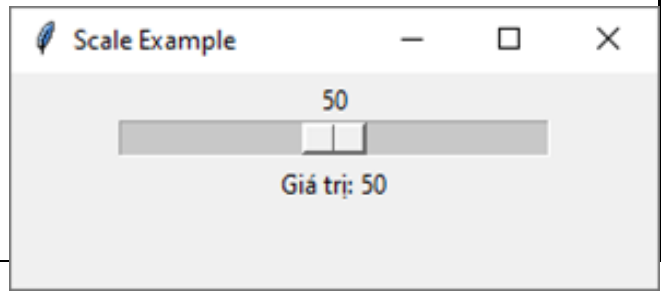
scale

Scale cho phép người dùng chọn giá trị trong một phạm vi số đã định. Nó thường được sử dụng để điều chỉnh giá trị số như điều chỉnh âm lượng, độ sáng, độ dài, và các tham số khác.

Để tạo một Scale trong tkinter, chúng ta cần sử dụng lớp Scale và cung cấp các thông số cần thiết như giá trị tối thiểu (from_), giá trị tối đa (to), bước nhảy (resolution), và giá trị mặc định (variable).

Ví dụ 7.1.2.9. Sử dụng Scale.

```
1  import tkinter as tk
2  from tkinter import ttk
3
4  def on_scale_change(value):
5      label.config(text=f"Giá trị: {value}")
6
7  root = tk.Tk()
8  root.title("Scale Example")
9  root.geometry("300x100")
10
11 scale = tk.Scale(root, from_=0, to=100, resolution=1,
12 orient=tk.HORIZONTAL, length=200, command=on_scale_change)
13 scale.set(50)  # Giá trị mặc định khi khởi tạo
14 scale.pack()
15 label = ttk.Label(root, text="Giá trị: 50")
16 label.pack()
17
18 root.mainloop()
```



Messagebox

Messagebox là một hộp thoại thông báo được sử dụng để hiển thị thông báo hoặc cảnh báo đến người dùng. Nó cung cấp các hộp thoại đơn giản để thông báo các thông tin quan trọng, hỏi người dùng xác nhận một hành động hoặc cảnh báo về lỗi trong ứng dụng.

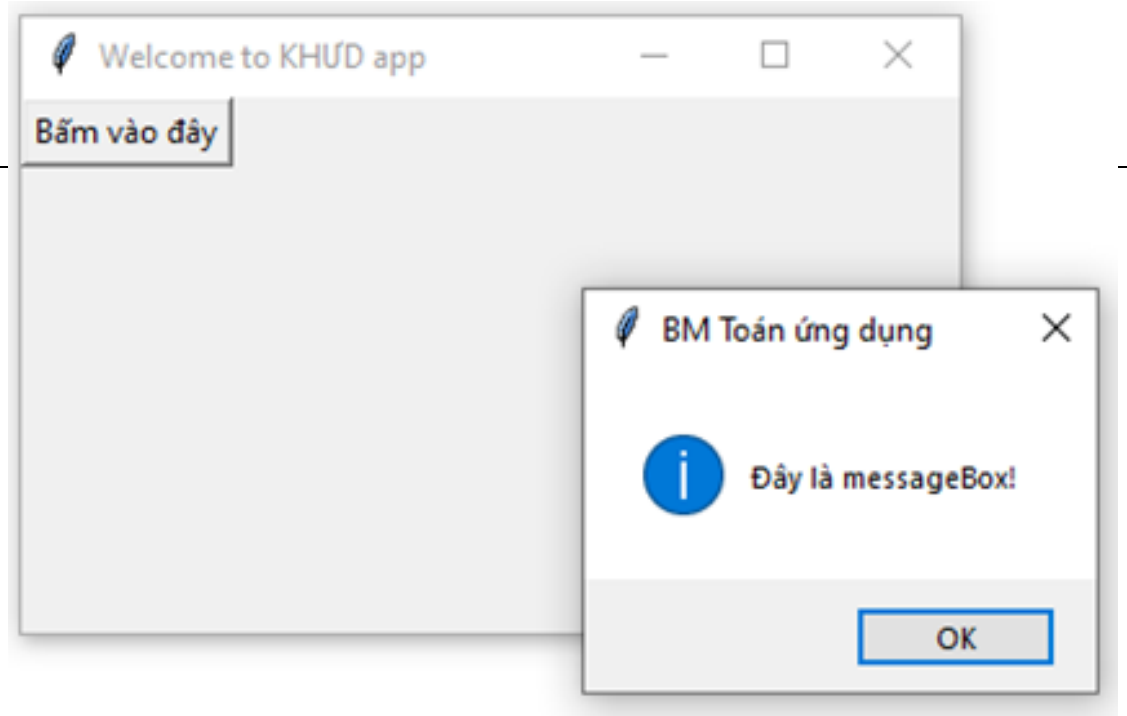
Thư viện 'Messagebox' trong tkinter bao gồm các hàm:

- showinfo(): Hiển thị hộp thoại thông báo thông tin.
- showwarning(): Hiển thị hộp thoại cảnh báo.
- showerror(): Hiển thị hộp thoại cảnh báo lỗi.
- askquestion(): Hiển thị hộp thoại hỏi người dùng có đồng ý hay không.
- askokcancel(): Hiển thị hộp thoại hỏi người dùng có đồng ý hay hủy bỏ.
- askyesno(): Hiển thị hộp thoại hỏi người dùng có đồng ý hay không (Yes/No).

Tùy vào yêu cầu cụ thể, chúng ta có thể sử dụng 'Messagebox' để tương tác với người dùng trong các trường hợp khác nhau trong ứng dụng của mình.

Ví dụ 7.1.2.10. Sử dụng Messagebox.

```
1  from tkinter import *
2  from tkinter import messagebox
3
4  window = Tk()
5  window.title("Welcome to KHƯD app")
6  window.geometry('350x200')
7
8  def clicked():
9      messagebox.showinfo('BM Toán ứng dụng', 'Đây là messageBox!')
10
11 btn = Button(window,text='Bấm vào đây', command=clicked)
12 btn.grid(column=0,row=0)
13
14 window.mainloop()
```



SpinBox

- SpinBox là một widget cho phép người dùng chọn một giá trị từ một danh sách các giá trị cho trước. Nó thường được sử dụng để chọn các giá trị số nguyên hoặc số thập phân trong một khoảng giá trị cụ thể.
- SpinBox cung cấp một hộp văn bản để hiển thị giá trị hiện tại và hai nút mũi tên lên xuống để thay đổi giá trị. Bằng cách nhấn vào các nút mũi tên, người dùng có thể tăng hoặc giảm giá trị của SpinBox.

Ví dụ 7.1.2.11.

Tạo một ứng dụng giao diện người dùng (GUI) đơn giản sử dụng thư viện Tkinter, bao gồm một Spinbox, hai Label và một Button. Mục đích chính của ứng dụng này là cho phép người dùng chọn một giá trị từ Spinbox và hiển thị giá trị đó khi nhấn nút "Kết quả".

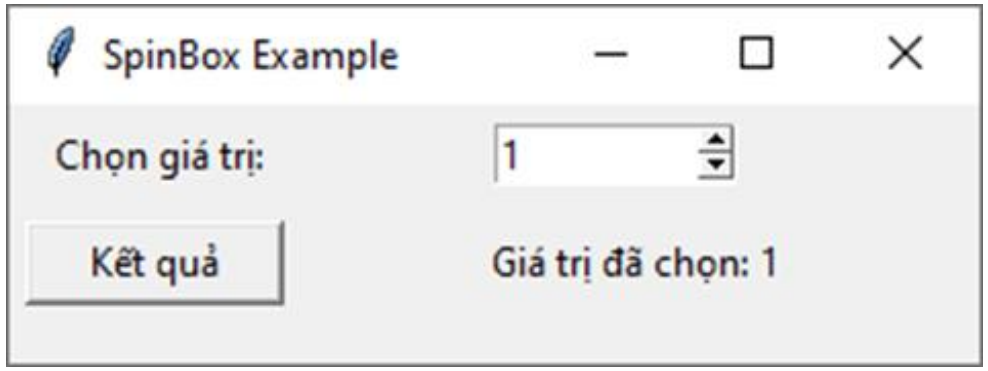
Ví dụ 7.1.2.11.

```
1   import tkinter as tk
2   from tkinter import Spinbox
3
4   def on_spinbox_changed():
5       value = spinbox.get()
6       label2.config(text="Giá trị đã chọn: " + value)
7
8   root = tk.Tk()
9   root.title("SpinBox Example")
10  root.geometry('300x80')
11
12  spinbox = Spinbox(root, from_=1, to=10, increment=1, width=10)
```

Ví dụ 7.1.2.11.

```
13    spinbox.grid(row=0, column=2)
14
15    label1 = tk.Label(root, text=" Chọn giá trị:", anchor="e")
16    label1.grid(row=0, column=0, padx=5, pady=5)
17
18    label2 = tk.Label(root, text="Giá trị đã chọn là:", width=20,
19    anchor="e")
20    label2.grid(row=1, column=0, padx=5, pady=5, columnspan=2)
21
22    button = tk.Button(root, text="Kết quả", width=10,
23    command=on_spinbox_changed)
24    button.grid(row=1, column=0, padx=5, pady=5, sticky="w")
25
26    root.mainloop()
```

Kết quả thực hiện chương trình:



Làm việc với trình chọn tệp và thư mục

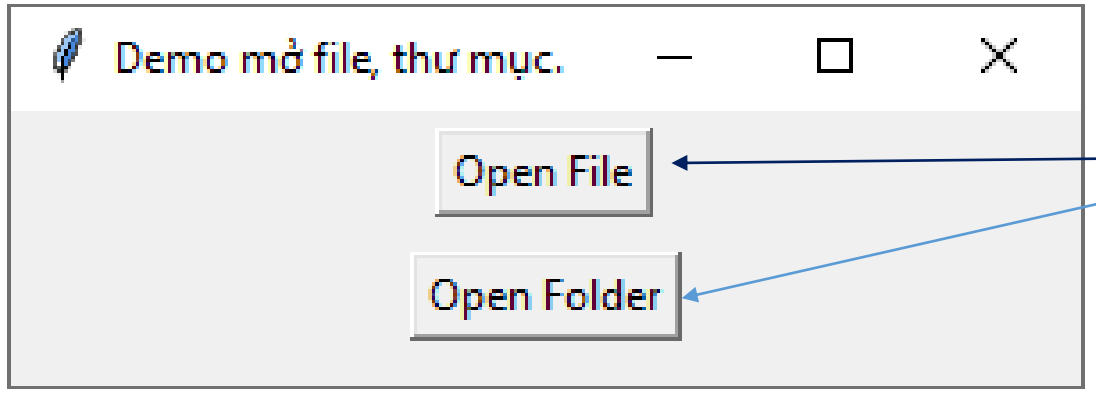
Trong Tkinter chúng ta có thể sử dụng thư viện filedialog để tạo một hộp thoại chọn tệp hoặc thư mục.

Thư viện filedialog cung cấp các hàm để hiển thị các hộp thoại cho phép người dùng duyệt và chọn tệp hoặc thư mục từ hệ thống tệp của mình.

Ví dụ 7.1.2.12.

```
1  import tkinter as tk
2  from tkinter import filedialog
3
4  def open_file():
5      file_path = filedialog.askopenfilename()
6      print("Selected file:", file_path)
7
8  def open_folder():
9      folder_path = filedialog.askdirectory()
10     print("Selected folder:", folder_path)
11
12 root = tk.Tk()
13 root.title('Demo mở file, thư mục.')
14 root.geometry('300x80')
15 open_file=tk.Button(root,text="Open File", command=open_file)
16 open_file.pack(padx=5,pady=5)
17
18 open_folder=tk.Button(root, text="Open Folder", command=open_folder)
19 open_folder.pack(padx=5, pady=5)
20
21 root.mainloop()
```

Kết quả thực hiện chương trình:



Click vào nút 'Open File'/'Open Folder' để mở hộp thoại cho phép chọn file/thư mục tùy mục đích sử dụng.

***Làm việc với Menu*:** Menu là một phần quan trọng trong việc tạo các giao diện người dùng tương tác. Menu trong tkinter chứa các tùy chọn hoặc lựa chọn để người dùng có thể thực hiện các hành động nhất định hoặc chọn các tùy chọn từ danh sách.

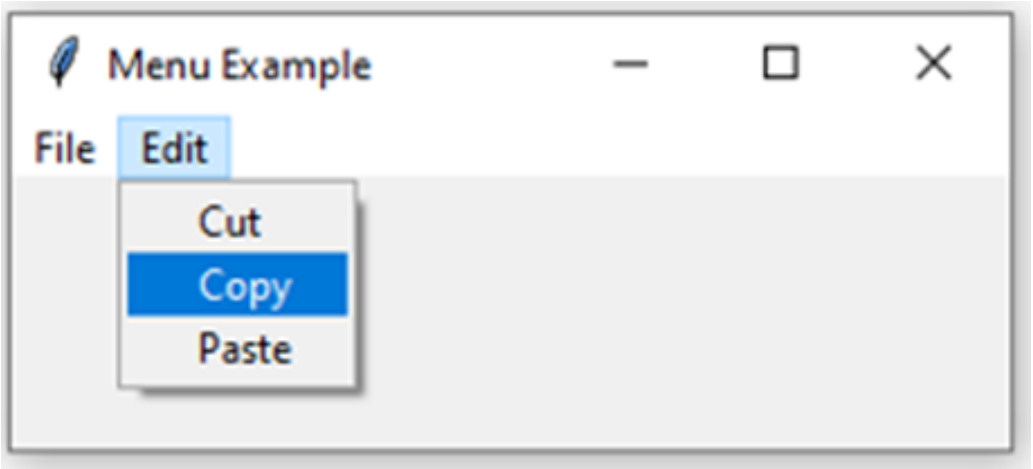
Menu trong tkinter thường bao gồm các loại sau:

- Menu chính (**Main Menu**): Đây là menu chính của chương trình, thường nằm ở đầu cửa sổ và chứa các tùy chọn chức năng chính của ứng dụng.
- Menu con (**Submenu**): Là các menu con được hiển thị khi người dùng chọn một tùy chọn trong menu chính.
- Menu ngữ cảnh (**Context Menu**): Là menu xuất hiện khi người dùng nhấp chuột phải vào một vùng hoặc phần tử cụ thể trong giao diện.
- Để tạo menu trong tkinter, người dùng cần sử dụng các đối tượng như **Menu**, **add_cascade**, **add_command**, **add_separator**, **add_checkbutton**, **add_radiobutton**, vv.

Ví dụ 7.1.2.13. Sử dụng Menu.

```
1  import tkinter as tk
2
3  def select1():
4      print("Bạn đã chọn menu con 'New'!")
5  def select2():
6      print("Bạn đã chọn menu con 'Open'!")
7  def select3():
8      print("Bạn đã chọn menu con 'Cut'!")
9  def select4():
10     print("Bạn đã chọn menu con 'Copy'!")
11 def select5():
12     print("Bạn đã chọn menu con 'Paste'!")
13
14 def quit_app():
15     root.quit()
16
17 root = tk.Tk()
18 root.title("Menu Example")
19 root.geometry('300x80')
20
21 menu_bar = tk.Menu(root)
22 root.config(menu=menu_bar)
23
24 file_menu = tk.Menu(menu_bar, tearoff=0)
25 menu_bar.add_cascade(label="File", menu=file_menu)
26 file_menu.add_command(label="New", command=select1)
27 file_menu.add_command(label="Open", command=select2)
28 file_menu.add_separator()
29 file_menu.add_command(label="Exit", command=quit_app)
30
31 edit_menu = tk.Menu(menu_bar, tearoff=0)
32 menu_bar.add_cascade(label="Edit", menu=edit_menu)
33 edit_menu.add_command(label="Cut", command=select3)
34 edit_menu.add_command(label="Copy", command=select4)
35 edit_menu.add_command(label="Paste", command=select5)
36
37 root.mainloop()
```

Kết quả thực hiện chương trình:



Làm việc với các Tabs

- Trong tkinter, "Tabs" là một loại tiện ích hiển thị nội dung trong các tabbed widget (widget có các tab). Tabs thường được sử dụng để chia nhóm các nội dung liên quan vào các tab riêng biệt.
- Các tabs thường được sử dụng trong các ứng dụng có nhiều chức năng hoặc tùy chọn khác nhau. Thay vì hiển thị tất cả các chức năng hoặc tùy chọn cùng một lúc, các tabs cho phép người dùng chọn tab tương ứng để xem nội dung liên quan đến chức năng hoặc tùy chọn đó.
- Tabs trong tkinter thường được thực hiện bằng cách sử dụng các widget như **ttk.Notebook** hoặc **ttk.TabbedFrame**. Các tabs này cho phép chúng ta thêm các tab con (tab items) và chứa các widget hoặc giao diện người dùng riêng biệt trong từng tab. Khi người dùng chọn một tab cụ thể, nội dung của tab đó sẽ được hiển thị trên giao diện.

Ví dụ 7.1.2.14.

```
1  import tkinter as tk
2  from tkinter import ttk
3
4  root = tk.Tk()
5  root.title("Tabs Example")
6  root.geometry('300x80')
7
8  notebook = ttk.Notebook(root)
9
10 tab1 = ttk.Frame(notebook)
11 tab2 = ttk.Frame(notebook)
12
13 notebook.add(tab1, text="Tab 1")
14 notebook.add(tab2, text="Tab 2")
15
16 label1 = tk.Label(tab1, text="This is Tab 1")
17 label1.pack(padx=5,pady=5)
18
19 label2 = tk.Label(tab2, text="This is Tab 2")
20 label2.pack(padx=5,pady=5)
21
22 notebook.pack(fill="both", expand=True)
23
24 root.mainloop()
```

Thêm Widgets vào Notebook

Sau khi tạo các tab, chúng ta có thể đặt các widget bên trong các tab này bằng cách gán thuộc tính cha cho tab mong muốn.

Thêm widget vào tab bằng cách gán "cha"

- Trong Tkinter, mọi widget đều có một cha (parent). Parent là widget chứa widget đó, và nó quyết định nơi widget con sẽ xuất hiện trong giao diện.
- Khi chúng ta tạo một widget (ví dụ: Label, Button, Entry), cần phải chỉ định parent để widget biết nó sẽ thuộc về container nào.

Cách gán cha (parent) cho widget

Cú pháp chung

```
widget = WidgetClass(parent, options...)
```

Trong đó:

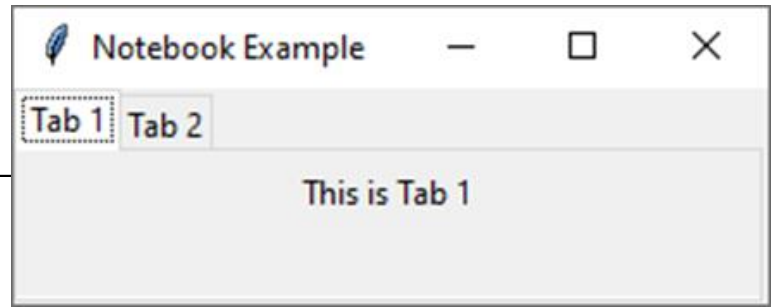
- WidgetClass: Loại widget muốn tạo (ví dụ: Label, Button, Frame, Entry,...).
- parent: Widget cha, tức là nơi widget con sẽ được hiển thị.
- options...: Các tùy chọn cấu hình widget.

Ví dụ: thêm một Button và một Entry vào "Tab số 1" và một Text vào "Tab số 2".

```
button1 = tk.Button(tab1, text="Button in Tab 1")
button1.pack(pady=5)

entry1 = tk.Entry(tab1)
entry1.pack(pady=5)

text2 = tk.Text(tab2, height=5, width=30)
text2.pack(padx=5, pady=5)
```

Ví dụ 7.1.2.15.

```
1  import tkinter as tk
2  from tkinter import ttk
3
4  root = tk.Tk()
5  root.title("Tabs Example")
6  root.geometry('300x80')
7
8  notebook = ttk.Notebook(root)
9
10 tab1 = ttk.Frame(notebook)
11 tab2 = ttk.Frame(notebook)
12
13 notebook.add(tab1, text="Tab 1")
14 notebook.add(tab2, text="Tab 2")
15
16 label1 = tk.Label(tab1, text="This is Tab 1")
17 label1.pack(padx=5,pady=5)
18
19 label2 = tk.Label(tab2, text="This is Tab 2")
20 label2.pack(padx=5,pady=5)
21
22 notebook.pack(fill="both", expand=True)
23
24 root.mainloop()
```



7.1.3. Bố cục (layout) trong Python Tkinter

Python Tkinter cung cấp các phương thức để bố cục (layout) các widget sau.

- Phương thức pack().
- Phương thức grid().
- Phương thức place().

a. Phương thức pack(): Phương thức pack() cho phép chúng ta xếp các widget theo chiều dọc hoặc chiều ngang. Nó tự động điều chỉnh kích thước của các widget và đẩy chúng lại với nhau để sắp xếp. Chúng ta có thể sử dụng các tham số như **side**, **fill** và **expand** để điều chỉnh cách các widget được xếp chồng và căn chỉnh trong cửa sổ.

Cú pháp: `widget.pack(options)`

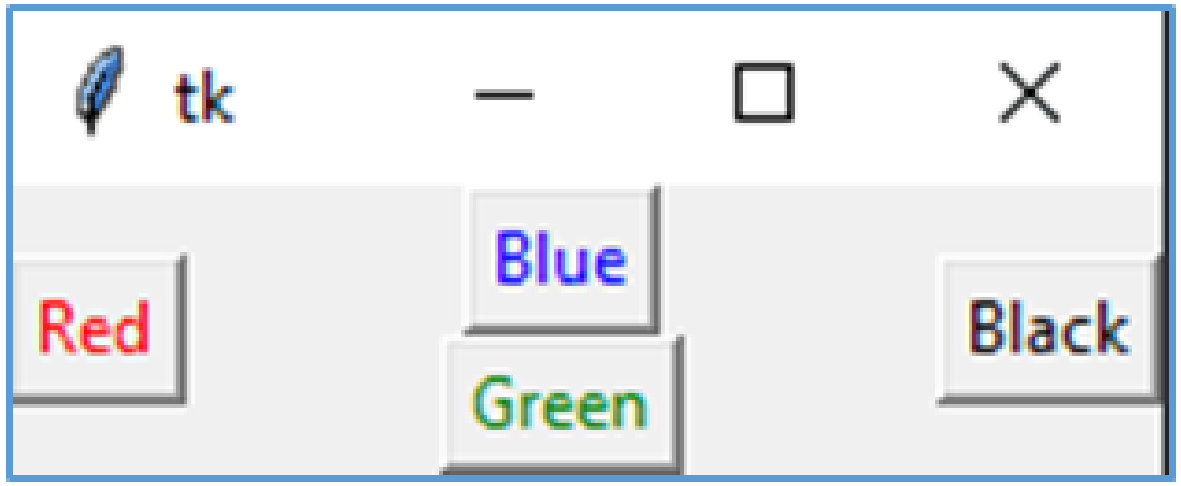
Các tùy chọn (options):

- **expand** : Nếu expand là **True** thì tiện ích con sẽ mở rộng để lấp đầy khoảng trống.
- **Fill** : Xác định xem widget con có lấp đầy bất kỳ không gian thừa nào do trình đóng gói phân bổ cho nó hay không hoặc giữ các kích thước tối thiểu của riêng nó: NONE(mặc định), X (chỉ điền theo chiều ngang), Y (chỉ điền theo chiều dọc) hay BOTH (điền theo cả chiều ngang và chiều dọc).
- **side** : Nó giúp xác định vị trí của widget so với widget cha.

Ví dụ 7.1.3.1. Ví dụ đơn giản về việc sử dụng Tkinter để tạo một giao diện người dùng đơn giản với bốn nút (button) có màu và được sắp xếp theo các hướng khác nhau.

```
from tkinter import * # import tất cả các lớp và hàm từ module tkinter vào
# không  gian tên hiện tại, cho phép sử dụng chúng mà không cần tiền tố tkinter.

parent = Tk()

# Đặt kích thước của cửa sổ theo pixels,  chiều rộng là 200 và chiều cao là 50
parent.geometry("200x50")

#Tạo một đối tượng nút (button) có chữ "Red" màu đỏ. Và gán cho biến redbutton.
redbutton = Button(parent, text="Red", fg="red")
redbutton.pack(side=LEFT) #sắp xếp nút redbutton vào cửa sổ gốc, đặt nút bên trái.
#Tiếp tục, tương tự như trên với các biến greenbutton, bluebutton và blackbutton.
greenbutton = Button(parent, text="Black", fg="black")
greenbutton.pack(side=RIGHT)

bluebutton = Button(parent, text="Blue", fg="blue")
bluebutton.pack(side=TOP)

blackbutton = Button(parent, text="Green", fg="green")
blackbutton.pack(side=BOTTOM)

#Bắt đầu vòng lặp chạy của ứng dụng, cho phép giao diện người dùng hiển thị và xử lý
#sự kiện cho đến khi cửa sổ được đóng.
parent.mainloop()
```

Kết quả thực hiện chương trình:



b. Phương thức grid(): Phương thức grid() cho phép người dùng sắp xếp các widget trong lưới ô vuông ở dạng bảng. Chúng ta có thể xác định vị trí của từng widget bằng cách chỉ định hàng (row) và cột (column) trong lưới. Bằng cách sử dụng các tham số như rowspan và columnspan, lập trình viên có thể điều chỉnh kích thước và vị trí của các ô trong lưới. Chúng ta cũng có thể chỉ định khoảng cột (chiều rộng) hoặc chiều dài hàng (chiều cao) của widget con.

Cú pháp: `widget.grid(options)`

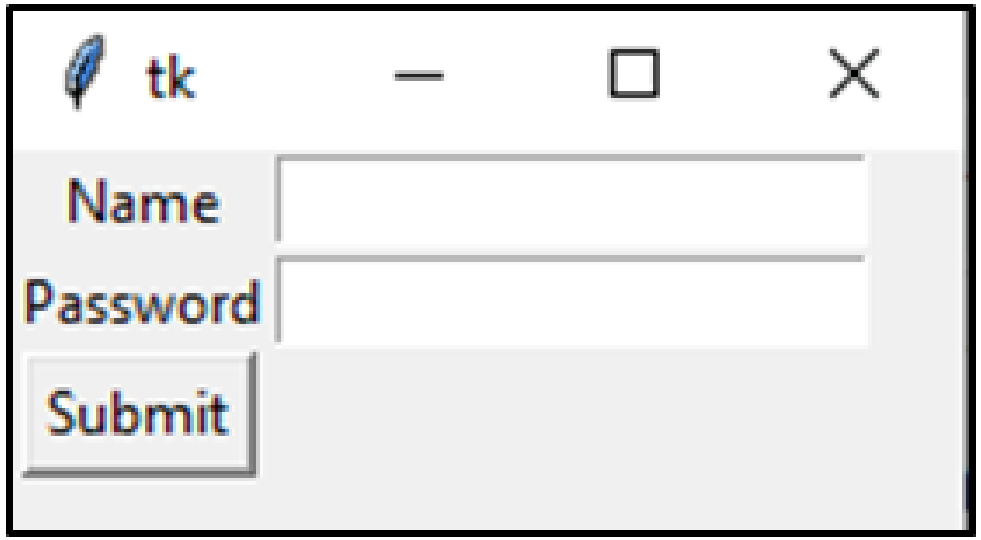
Các tùy chọn (options) của phương thức grid bao gồm:

- Column: Số cột mà widget con sẽ được đặt. Cột ngoài cùng bên trái được biểu thị bằng 0.
- Columnspan: Chiều rộng của widget con. Nó đại diện cho số cột mà đến đó, cột được mở rộng.
- ipadx, ipady: Nó đại diện cho số lượng pixel cho đường viền của gidget.
- padx, pady: Nó đại diện cho số lượng pixel bên ngoài đường viền của gidget.
- row: Số hàng mà widget con sẽ được đặt. Hàng trên cùng được biểu thị bằng 0.
- rowspan: Chiều cao của widget con, tức là số hàng mà tiện ích con được mở rộng.
- Sticky: Nếu ô lớn hơn một widget, thì Sticky được sử dụng để chỉ định vị trí của widget bên trong ô. Nó có thể là sự ghép nối của các chữ cái sticky đại diện cho vị trí của widget. Nó có thể là N, E, W, S, NE, NW, NS, EW, ES.

Ví dụ 7.1.3.2. Tạo một giao diện người dùng đơn giản với các nhãn (label), ô nhập liệu (entry) và một nút (button) "Submit".

```
1  from tkinter import * # import tất cả các lớp và hàm từ module tkinter vào
2  # không  gian tên hiện tại, cho phép sử dụng chúng mà không cần tiền tố tkinter.
3
4  parent = Tk()
5  # Đặt kích thước của cửa sổ theo pixels,  chiều rộng là 200 và chiều cao là 50
6  parent.geometry("200x50")
7
8  #tạo một đối tượng nhãn (label) với văn bản "Name" và gắn nó vào cửa sổ gốc.
9  #phương thức grid() được gọi để xác định vị trí của nhãn với row=0 và column=0.
10 name = Label(parent, text = "Name").grid(row = 0, column = 0)
11
12 #Tương tự, các dòng mã 14,15,16 tạo và định vị các nhãn và ô nhập liệu khác trong lưới.
13 e1 = Entry(parent).grid(row = 0, column = 1)
14 password = Label(parent, text = "Password").grid(row = 1, column = 0)
15 e2 = Entry(parent).grid(row = 1, column = 1)
16 submit = Button(parent, text = "Submit").grid(row = 4, column = 0)
17
18 #Bắt đầu vòng lặp chạy của ứng dụng, cho phép giao diện người dùng hiển thị và xử lý
19 #sự kiện cho đến khi cửa sổ được đóng.
20 parent.mainloop()
```

Kết quả thực hiện chương trình:



Trong ví dụ 7.1.3.2 ở trên, mọi widget (nhãn, ô nhập liệu, nút) được gắn vào cửa sổ gốc và được xác định vị trí trong lưới bằng phương thức grid().

Lưu ý: Trong Tkinter, khi gọi phương thức grid() trực tiếp sau khi tạo một widget như Label, Entry, hoặc Button,… nó sẽ trả về None thay vì trả về đối tượng widget đã được tạo.

Vì vậy mã lệnh dòng 10, 14, 16, nên tách riêng, cụ thể dòng 10 sửa lại là:

```
name = Label(parent, text = "Name")
name.grid(row = 0, column = 0)
```

Tương tự với các dòng 14, 16.

*c. Phương thức place():*Phương thức **palace()** cho phép người dùng định vị và điều chỉnh vị trí của các widget trong cửa sổ dựa trên tọa độ tuyệt đối hoặc tương đối. Chúng ta có thể xác định tọa độ x và y để đặt widget ở vị trí mong muốn trên cửa sổ. Tuy nhiên, place() cần sự thủ công và có thể khó khăn khi quản lý các widget phức tạp.

Cú pháp: `widget.place(options)`

Trong đó, widget là đối tượng widget muốn định vị, và options là một tập hợp các tham số và giá trị để điều chỉnh vị trí và cách hiển thị của widget.

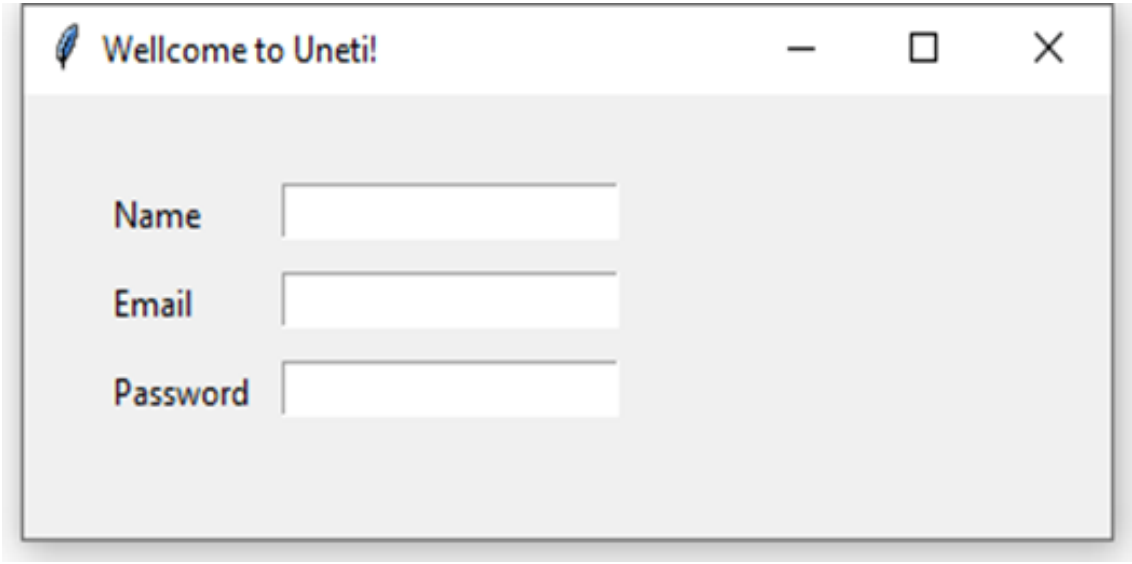
Danh sách tùy chọn options bao gồm:

- **Anchor**: thể hiện vị trí chính xác của widget con trong container. Giá trị mặc định là NW (góc trên bên trái).
- **bordermode**: Giá trị mặc định của kiểu đường viền là INSIDE đề cập đến việc bỏ qua giá trị gốc bên trong đường viền. Tùy chọn còn lại là OUTSIDE.
- **height, width**: Nó đề cập đến chiều cao và chiều rộng của widget tính bằng pixel.
- **relheight, relwidth**: Chiều cao và chiều rộng có giá trị trong khoảng 0,0 và 1,0.
- **relx, rely**: Có giá trị trong khoảng 0,0 và 1,0, là độ lệch theo hướng ngang và dọc.
- **x, y**: Nó đề cập đến độ lệch ngang và dọc theo pixel.

Ví dụ 7.1.3.3. Tạo một giao diện người dùng đơn giản sử dụng Tkinter để hiển thị các nhãn (label) và ô nhập liệu (entry) cho "Name", "Email" và "Password".

```
1  from tkinter import *
2
3  #tạo một đối tượng cửa sổ gốc (root window) sử dụng lớp Tk trong Tkinter
4  # và gán cho biến top. Cửa sổ gốc là cửa sổ chính của ứng dụng.
5  windows = Tk()
6  #Thêm tiêu đề cho cửa cửa sổ
7  windows.title("Wellcome to Uneti!")
8  windows.geometry("400x150") #Xác định kích thước của sổ tính theo pixels
9
10 # tạo một nhãn (label) có văn bản "Name" và gắn nó vào cửa sổ windows.
11 # Sau đó, gọi phương thức place() để đặt nhãn tại tọa độ tuyệt đối (30, 50).
12 name = Label(windows, text = "Name").place(x = 30, y = 30)
13
14 #Tương tự: Tạo các nhãn (label) "Email", "Password" với các ô nhập liệu (entry) tương ứng.
15 # Gọi phương thức place() để đặt các nhãn đó tại các tọa độ tuyệt đối x,y.
16 email = Label(windows, text = "Email").place(x = 30, y = 60)
17 password = Label(windows, text = "Password").place(x = 30, y = 90)
18
19 #Tương tự như trên, các dòng mã sau đó tạo và định vị các nhãn và ô nhập liệu khác.
20 e1 = Entry(windows).place(x = 95, y = 30)
21 e2 = Entry(windows).place(x = 95, y = 60)
22 e3 = Entry(windows).place(x = 95, y = 90)
23 windows.mainloop()
```

Kết quả thực hiện chương trình:

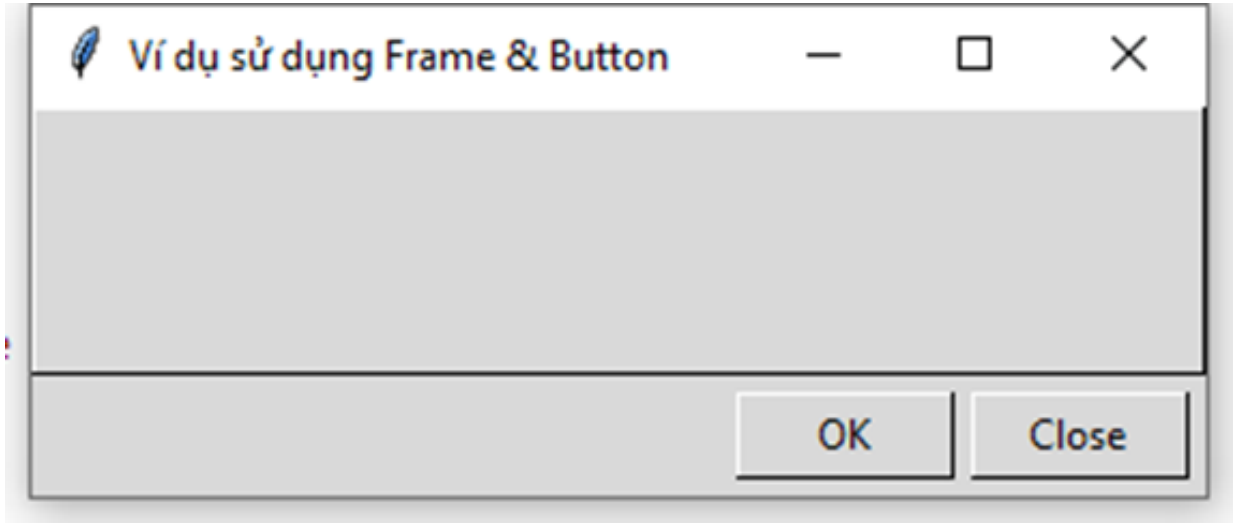


Lưu ý: Chúng ta có thể chọn bố cục (layout) phù hợp dựa trên yêu cầu của giao diện người dùng của mình. Một ứng dụng có thể sử dụng một hoặc kết hợp cả ba loại bố cục (layout) để đạt được sự linh hoạt và tổ chức tốt trong việc hiển thị các widget.

Ví dụ minh họa 1: tạo một ứng dụng sử dụng Tkinter để hiển thị hai nút nhấn "Close" và "OK" trong một khung (frame).

```
1  from tkinter import Tk, RIGHT, BOTH, RAISED
2  from tkinter.ttk import Frame, Button, Style
3  #định nghĩa một lớp con có tên là Example, kế thừa từ lớp Frame trong Tkinter.
4  # Lớp Example đại diện cho ứng dụng.
5  class Example(Frame):
6      def __init__(self, parent):
7          Frame.__init__(self, parent)
8          self.parent = parent
9          self.initUI()
10     def initUI(self):
11         self.parent.title("Ví dụ sử dụng Frame & Button")
12         self.style = Style()
13         self.style.theme_use("default")
14
15         self.pack(fill=BOTH, expand=True)
16         frame = Frame(self, relief=RAISED, borderwidth=1)
17         frame.pack(fill=BOTH, expand=True)
18
19
20         closeButton = Button(self, text="Close")
21         closeButton.pack(side=RIGHT, padx=5, pady=5)
22         okButton = Button(self, text="OK")
23         okButton.pack(side=RIGHT)
24 windows = Tk()
25 windows.geometry("300x200+300+300")
26 app = Example(windows)
   windows.mainloop()
```

Kết quả thực hiện chương trình



Ví dụ minh họa 2: Viết một ứng dụng tạo ra một cửa sổ gồm ba khung và ba phần nhập liệu, cho phép người dùng nhập thông tin về giảng viên, bộ môn và dự án giảng viên đó đang thực hiện.

Ví dụ minh họa 2

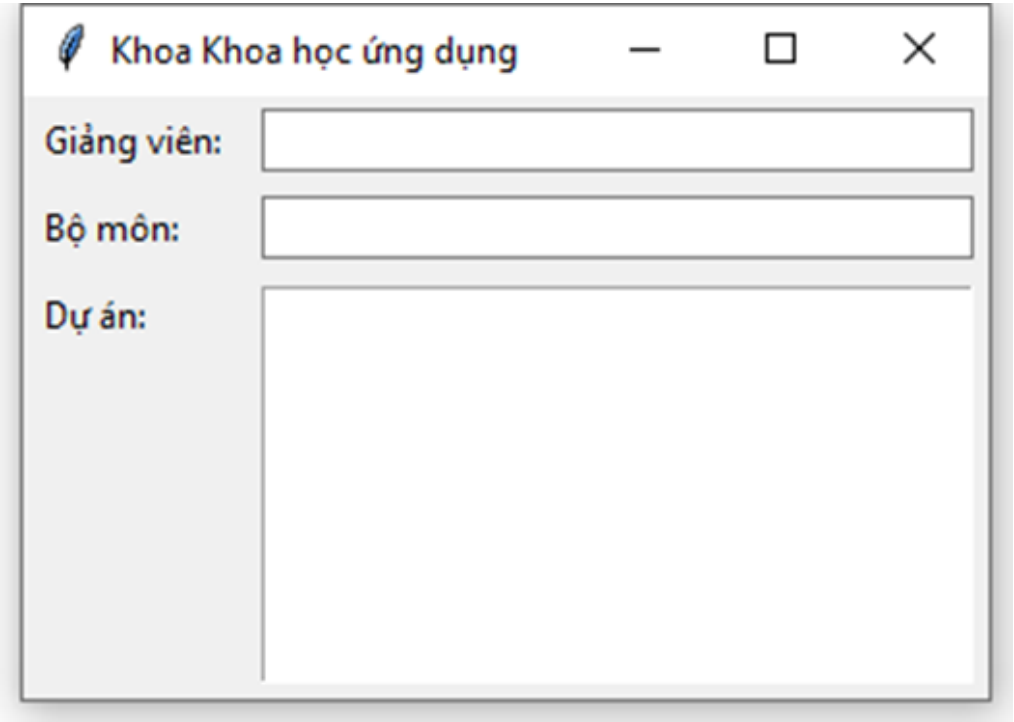
```
1  from tkinter import Tk, Text, TOP, BOTH, X, N, LEFT
2  from tkinter.ttk import Frame, Label, Entry
3
4  class Demo_2(Frame):
5      def __init__(self, parent):
6          Frame.__init__(self, parent)
7          self.parent = parent
8          self.initGUI()
9
10     def initGUI(self):
11         self.parent.title("Khoa Khoa học ứng dụng")
12         self.pack(fill=BOTH, expand=True)
13
14         frame1 = Frame(self)
15         frame1.pack(fill=X)
16
17         lbl1 = Label(frame1, text="Giảng viên:", width=10)
18         lbl1.pack(side=LEFT, padx=5, pady=5)
19
20         entry1 = Entry(frame1)
21         entry1.pack(fill=X, padx=5, expand=True)
22
23         frame2 = Frame(self)
           frame2.pack(fill=X)
```

Ví dụ minh họa 2

```
24          lbl2 = Label(frame2, text="Bộ môn:", width=10)
25          lbl2.pack(side=LEFT, padx=5, pady=5)
26
27          entry2 = Entry(frame2)
28          entry2.pack(fill=X, padx=5, expand=True)
29
30          frame3 = Frame(self)
31          frame3.pack(fill=BOTH, expand=True)
32
33          lbl3 = Label(frame3, text="Dự án:", width=10)
34          lbl3.pack(side=LEFT, anchor=N, padx=5, pady=5)
35
36          txt = Text(frame3)
37          txt.pack(fill=BOTH, pady=5, padx=5, expand=True)
38
39  windows_root = Tk()
40  windows_root.geometry("320x200+320+200")
41  app = Demo_2(windows_root)
42  windows_root.mainloop()
```

Ví dụ minh họa 2

Kết quả thực hiện chương trình



*Ví dụ minh họa 3:*Thiết kế giao diện một ứng dụng tính toán (máy tính cầm tay).

Ví dụ minh họa 3

```
1   from tkinter import Tk, W, E
2   from tkinter.ttk import Frame, Button, Style
3   from tkinter.ttk import Entry
4
5   class Caculator(Frame):
6       def __init__(self, parent):
7           Frame.__init__(self, parent)
8           self.parent = parent
9           self.initGUI()
10
11      def initGUI(self):
12          self.parent.title("Ứng dụng tính toán")
13
14          Style().configure("TButton", padding=(0, 5, 0, 5),
15  font='serif 10')
16
17          self.columnconfigure(0, pad=3)
18          self.columnconfigure(1, pad=3)
19          self.columnconfigure(2, pad=3)
20          self.columnconfigure(3, pad=3)
21
22          self.rowconfigure(0, pad=3)
23          self.rowconfigure(1, pad=3)
24          self.rowconfigure(2, pad=3)
25          self.rowconfigure(3, pad=3)
26          self.rowconfigure(4, pad=3)
27
```

Ví dụ minh họa 3

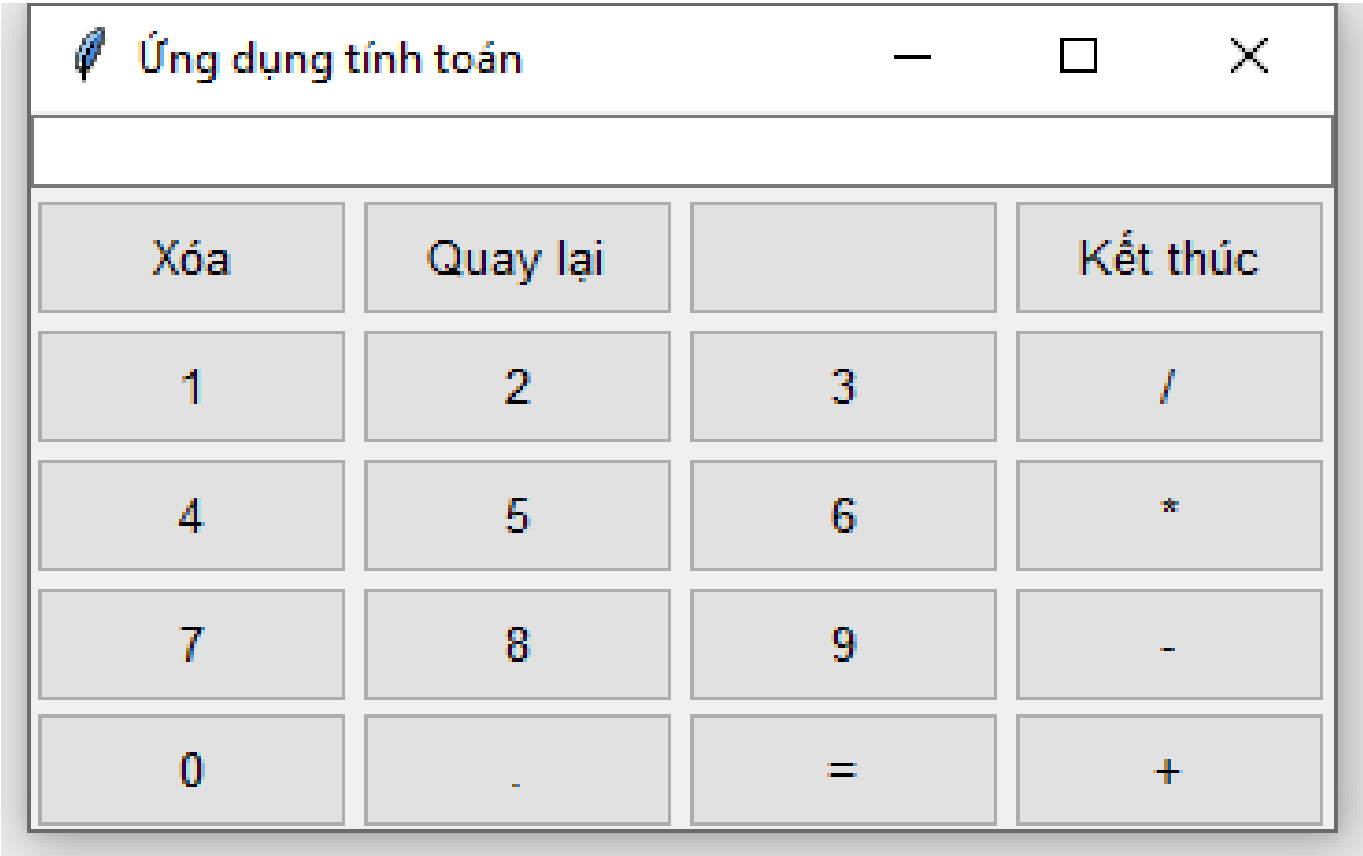
```
28  entry = Entry(self)
29          entry.grid(row=0, columnspan=4, sticky=W+E)
30          clear = Button(self, text="Xóa")
31          clear.grid(row=1, column=0)
32          back = Button(self, text="Quay lại")
33          back.grid(row=1, column=1)
34          lbl = Button(self)
35          lbl.grid(row=1, column=2)
36          close = Button(self, text="Kết thúc")
37          close.grid(row=1, column=3)
38          mot = Button(self, text="1")
39          mot.grid(row=2, column=0)
40          hai = Button(self, text="2")
41          hai.grid(row=2, column=1)
42          ba = Button(self, text="3")
43          ba.grid(row=2, column=2)
44          Chia = Button(self, text="/")
45          Chia.grid(row=2, column=3)
46
47          bon = Button(self, text="4")
48          bon.grid(row=3, column=0)
49          nam = Button(self, text="5")
50          nam.grid(row=3, column=1)
51          sau = Button(self, text="6")
52          sau.grid(row=3, column=2)
53          Nhan = Button(self, text="*")
54          Nhan.grid(row=3, column=3)
55
```

Ví dụ minh họa 3

```
56  bay = Button(self, text="7")
57          bay.grid(row=4, column=0)
58          tam = Button(self, text="8")
59          tam.grid(row=4, column=1)
60          chin = Button(self, text="9")
61          chin.grid(row=4, column=2)
62          Tru = Button(self, text="-")
63          Tru.grid(row=4, column=3)
64
65          zero = Button(self, text="0")
66          zero.grid(row=5, column=0)
67          dau_cham = Button(self, text=".")
68          dau_cham.grid(row=5, column=1)
69          bang = Button(self, text="=")
70          bang.grid(row=5, column=2)
71          Cong = Button(self, text="+")
72          Cong.grid(row=5, column=3)
73
74          self.pack()
75
76  windows_root = Tk()
77  app = Caculator(windows_root)
78  windows_root.mainloop()
```

Ví dụ minh họa 3

Kết quả thực hiện chương trình

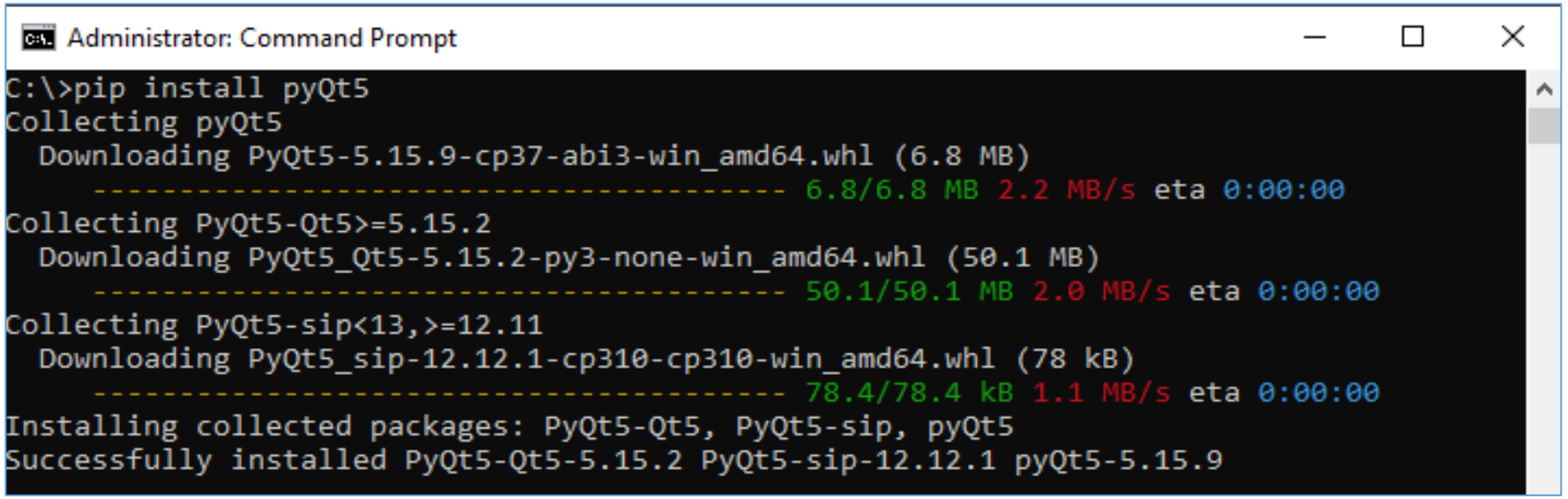


7.2. GIỚI THIỆU LẬP TRÌNH GUI VỚI PYQT

PyQt là một liên kết Python cho framework Qt, một công cụ phát triển ứng dụng đa nền tảng mạnh mẽ. PyQt cung cấp khả năng tùy chỉnh mạnh mẽ và khả năng tạo giao diện đồ họa chuyên nghiệp.

Trong chương này chúng ta sẽ làm việc với thư viện pyQt5, để có mô đun này chúng ta cần cài đặt pyQt5 trong môi trường Python. Có thể sử dụng pip để cài đặt pyQt5 như sau:

Từ giao diện của sổ dòng lệnh cmd ở chế độ Administrator gõ lệnh: `pip install pyqt5`



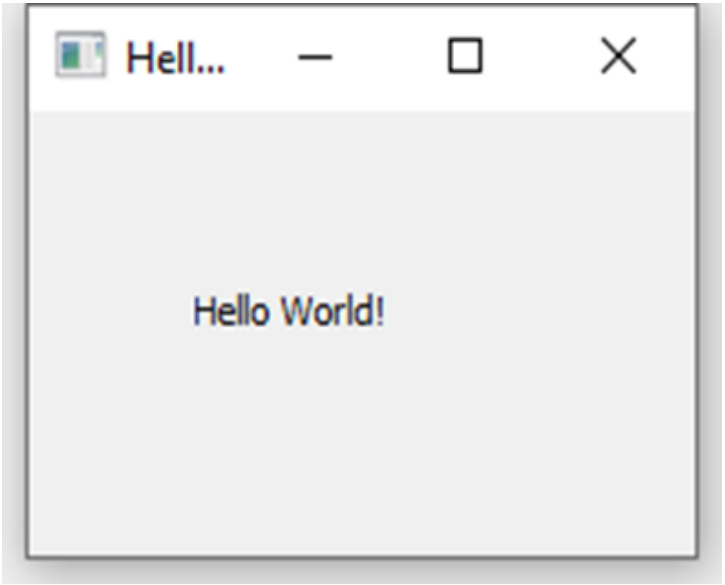
Hình 7.1. Cài đặt pyQt5

Sau khi cài PyQt5, chúng ta có thể viết chương trình đầu tiên như sau:

Ví dụ 7.2.1:

```
1  import sys
2  from PyQt5.QtWidgets import QApplication, QLabel, QWidget
3
4  # Khởi tạo ứng dụng
5  app = QApplication(sys.argv)
6
7  # Tạo một cửa sổ
8  window = QWidget()
9  window.setWindowTitle('Hello World!')
10
11 # Thêm các thành phần vào cửa sổ
12 label = QLabel(window)
13 label.setText('Hello World!')
14 label.move(50, 50)
15
16 # Hiển thị cửa sổ
17 window.show()
18
19 # Chạy vòng lặp ứng dụng
20 sys.exit(app.exec_())
```

Kết quả thực hiện chương trình:



CÂU HỎI THẢO LUẬN.

1. Python tkinter là gì?
2. Chức năng của Tkinter trong Python? Nêu các bước cơ bản để tạo một tkinter app?
3. Cho biết root tkinter là gì?
4. Chức năng phương thức Python Tkinter pack() là gì?
5. Chức năng phương thức grid()?
6. Chức năng phương thức place()?
7. Có bao nhiêu phương thức Python Tkinter pack()?
8. Có bao nhiêu phương thức Python Tkinter grid()?
9. Có bao nhiêu phương thức Python Tkinter palce()?
10. Làm cách nào để nhận được phản hồi API mới trong hộp văn bản Tkinter (Tkinter Textbox)?
11. Làm cách nào để lấy chỉ mục của tùy chọn option trong Combobox Tkinter?
12. Nêu các bước nhúng hình ảnh vào tiện ích Tkinter Canvas bằng PIL.
13. Làm thế nào để nhận được ngày hiện thời hiển thị trong cửa sổ tkinter?
14. Nêu cách tạo một thư mục mới sử dụng hộp thoại askdirectory trong tkinter?
15. Làm thế nào để sử dụng đối tượng StrinVar trong widget Entry trong tkinter?

BÀI TẬP VẬN DỤNG.

Làm các bài tập 7.1 đến 7.14 trang 335, 336 TLHT.