



Thư viện lập trình Tkinter Phần 1

Khoá học: Python căn bản

Kiểm tra bài trước

Hỏi và trao đổi về các khó khăn gặp phải trong bài “Lớp và đối tượng”
Tóm tắt lại các phần đã học từ bài “Lớp và đối tượng”

Mục tiêu



- Giới thiệu thư viện Tkinter
- Khái niệm GUI, các loại widget trong Tkinter
- Các trình quản lý hình học trong Tkinter
- Sự kiện trong Tkinter

Giới thiệu Tkinter



- GUI (Graphic user interface) hay còn gọi là giao diện người dùng đồ họa được sử dụng vô cùng phổ biến trong máy tính, các thiết bị đa phương tiện v.v
- Sự hiện diện của GUI ngày càng gia tăng trong kỷ nguyên số, khi các thiết bị số trở thành một phần không thể thay thế trong đời sống con người.
- Python cung cấp rất nhiều lựa chọn cho việc phát triển GUI. Tkinter, wxPython hay JPython là những công cụ quan trọng để tạo GUI.

Thảo luận

GUI là gì
Tkinter là gì
Các loại Widget

- GUI (Graphic user interface) là một dạng giao diện người dùng cho phép người dùng tương tác với máy tính thông qua những hình ảnh được vẽ trên màn hình
- Những đối tượng đồ họa đó sẽ mang thông tin, đại diện cho các hành động mà người dùng có thể làm.



Tkinter là gì

- Tkinter là thư viện GUI tiêu chuẩn cho Python. Khi kết hợp với Tkinter, Python sẽ được cung cấp các công cụ một cách nhanh chóng và dễ dàng để tạo các ứng dụng GUI.
- Tkinter cung cấp giao diện hướng đối tượng mạnh mẽ đến các bộ công cụ Tk GUI.

Các bước tạo ứng dụng Tkinter



Bước 1: Import Tkinter module

Bước 2: Tạo ra cửa sổ chính cho ứng dụng của GUI

Bước 3: Thêm một số widget cho ứng dụng

Bước 4: Tạo vòng lặp sự kiện chính để các hành động có thể diễn ra trên màn hình máy tính của người dùng



Ví dụ: Tạo text “Hello World !”

```
#Khai báo module tkinter
from tkinter import *
from tkinter.ttk import *

# Tạo cửa sổ chính cho giao diện
root = Tk()

# Đặt tiêu đề cho cửa sổ chính
root.title("First_Program")

# Tạo text "Hello World !" , đây là kết quả sẽ in ra
label = Label(root, text ="Hello World !").pack()

# Sử dụng phương thức để cửa sổ chính giao diện hiển thị ra màn hình
root.mainloop()
```

Các loại Widget



Widget	Mô tả
label	Dùng để hiển thị text hoặc hình ảnh trên màn hình
Button	Dùng để thêm các button vào ứng dụng
Canvas	Dùng để vẽ hình ảnh và các bố cục khác như text, đồ họa ...
ComboBox	Dùng chứa một mũi tên hướng xuống để chọn từ danh sách các tùy chọn có sẵn
CheckBox	Hiển thị một số tùy chọn cho người dùng dưới dạng các nút chuyển đổi mà từ đó người dùng có thể chọn
Entry	Sử dụng để nhập văn bản từ người dùng

Các loại Widget



Widget	Mô tả
Frame	Được sử dụng như một nơi để chứa và sắp xếp các widget
Message	Hiển thị hộp tin nhắn
Scale	Được sử dụng để cung cấp thanh trượt cho người dùng.
Scrollbar	Được sử dụng để cuộn lên xuống nội dung.
SpinBox	Cho phép người dùng lựa chọn các tùy chọn của giá trị
Menu	Tạo ra nhiều menu cho ứng dụng
Text	Cho phép người dùng chỉnh sửa văn bản nhiều dòng

Quản lý hình học (Geometry Management)



- Để tổ chức, sắp xếp hoặc đặt tất cả các widget trong cửa sổ chính, Tkinter cung cấp cho chúng ta cấu trúc hình học của các widget.
- Bố cục ứng dụng GUI chủ yếu được kiểm soát bởi quản lý hình học của Tkinter.
- Tất cả các widget Tkinter đều có quyền truy cập vào các phương pháp quản lý hình học cụ thể.
- Các phương pháp này được nhóm thành ba phương thức quản lý hình học, dùng để quản lý, sắp xếp các widget con trong các widget cha của chúng.

Phương thức trong quản lý hình học



Tên phương thức	Mô tả
pack()	Đặt các widget trong một khung hoặc cửa sổ theo một thứ tự nhất định.
grid()	Quản lý theo cấu trúc giống table. Thuận lợi cho việc thiết kế các hộp thoại
place()	Quản lý các widget bằng cách đặt chúng vào một vị trí cụ thể trong widget chính

Phương thức place()



- Sử dụng để sắp xếp các widget trong cửa sổ chính, đặt chúng vào một vị trí cụ thể theo chỉ dẫn của người lập trình
- Sắp xếp các widget theo các tọa độ x và y của nó. Cả hai tọa độ x và y đều tính bằng pixel
- Điểm gốc (0, 0) là góc trên cùng bên trái của khung hoặc cửa sổ.
- Do đó đối số y chỉ định khoảng cách tính theo pixel từ trên cùng của cửa sổ, đối số x chỉ định khoảng cách theo pixel từ bên trái của cửa sổ
- Kích thước và vị trí của widget không thay đổi nếu chúng ta thay đổi kích thước cửa sổ

Phương thức place(): Cú pháp



```
widget.place(options)
```

Trong đó:

- options là các tùy chọn như: **anchor**, **bordermode**, **height**, **width**, **relheight**, **relwidth**, **relx**, **rely**, **x**, **y**
- **x,y** : Nó được sử dụng để xác định độ lệch ngang và dọc theo pixel
- **height, width**: Dùng chỉ định chiều cao, chiều rộng của widget theo pixel
- **anchor**: Tùy chọn này chủ yếu thể hiện vị trí chính xác của widget trong vùng chứa. Giá trị mặc định là NW là góc trên bên trái. Anchor có thể có các giá trị như N,E,S,W,NE,NW,SE & SW.
- **bordermode**: Tùy chọn này cho biết giá trị mặc định của đường viền , nó sẽ bỏ qua giá trị chính bên trong đường viền
- **relx, rely**: Nó được sử dụng để xác định độ lệch ngang và dọc, dưới dạng giá trị nổi giữa 0,0 và 1,0
- **relheight, relwidth**: Nó được sử dụng để xác định chiều cao và chiều rộng dưới dạng giá trị thả nổi giữa 0,0 và 1,0

Phương thức place(): Ví dụ



Tạo ra một khung cửa sổ có các input Username, email, password

```
from tkinter import *
```

```
login_window = Tk()
```

```
login_window.geometry("400x250")
```

```
login_window.title("Login")
```

```
username = Label(login_window, text = "Username:"). place(x = 30, y = 50)
```

```
email = Label(login_window, text = "Email:"). place(x = 30, y = 90)
```

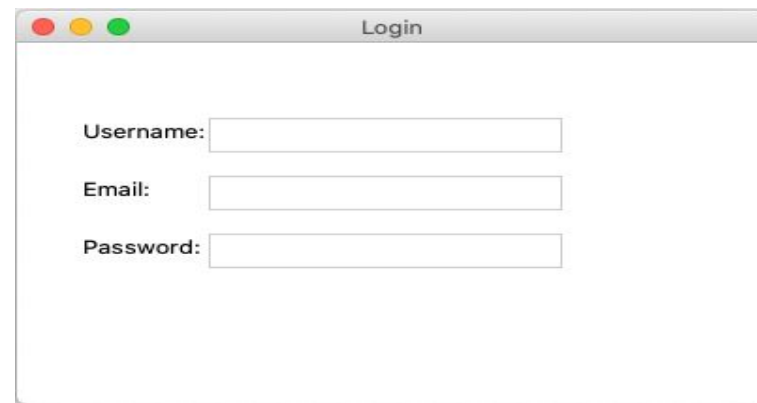
```
password = Label(login_window, text = "Password:"). place(x = 30, y = 130)
```

```
entry1 = Entry(login_window). place(x = 100, y = 50)
```

```
entry2 = Entry(login_window). place(x = 100, y = 90)
```

```
entry3 = Entry(login_window). place(x = 100, y = 130)
```

```
login_window.mainloop()
```



Phương thức grid()



- Grid() sử dụng các khái niệm về hàng và cột để sắp xếp các widget con
- Trình quản lý này sắp xếp các widget theo cấu trúc giống như một bảng
- Phương thức grid() đặt các widget con trong một bảng 2 chiều. Widget chính được chia thành một số hàng và cột và mỗi “ô” trong bảng kết quả có thể chứa một widget con
- Vị trí của cả hàng và cột sẽ bắt đầu từ 0

Phương thức grid(): Cú pháp



```
widget.grid(options)
```

Trong đó:

- options là các tùy chọn như: column, colspan, row, rowspan, padx, pady, ipadx, ipady và sticky
- **column** : Tùy chọn này chỉ định số cột mà widget sẽ được đặt. Giá trị mặc định là 0
- **colspan**: Dùng chỉ định số lượng cột mà widget sẽ được mở rộng
- **row**: Tùy chọn này chỉ định số hàng cho widget sẽ được đặt. Giá trị mặc định là 0
- **rowspan**: Dùng chỉ định số lượng hàng mà widget sẽ được mở rộng

Phương thức grid()

- **padx, pady:** Dùng thể hiện pixel của padding được thêm vào bên ngoài đường viền widget theo chiều ngang và dọc
- **ipadx, ipady:** Dùng thể hiện pixel của padding được thêm vào bên trong đường viền widget theo chiều ngang và dọc
- **sticky:** Sử dụng trong điều kiện ô lớn hơn widget, được sử dụng chỉ định vị trí của widget bên trong ô
 - Nếu không sử dụng **sticky**, thì widget mặc định đặt ở giữa ô.
 - **Sticky** có thể có các giá trị như N, E, S, W, NE, NW, SE & SW.
 - Nếu sticky = NE , thì vị trí của widget sẽ là trên cùng bên phải
 - Nếu sticky = SE , thì vị trí của widget sẽ là dưới cùng bên phải
 - Nếu sticky = NW , thì vị trí của widget sẽ là trên cùng bên trái
 - Nếu sticky = SW , thì vị trí của widget sẽ là dưới cùng bên trái



Phương thức grid(): Ví dụ

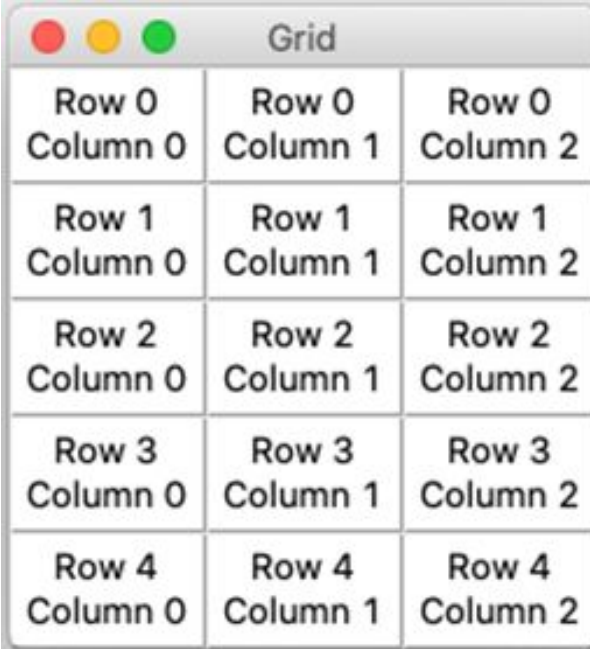
Tạo ra 1 khung có dạng grid theo tỉ lệ 5 hàng x 3 cột

```
import tkinter as tk

window = tk.Tk()
window.title("Grid")

for i in range(5):
    for j in range(3):
        frame = tk.Frame(
            master = window,
            relief = tk.RAISED,
            borderwidth = 1
        )
        frame.grid(row=i, column=j)
        label = tk.Label(master=frame, text=f"Row {i}\nColumn {j}")
        label.pack()

window.mainloop()
```



Row 0 Column 0	Row 0 Column 1	Row 0 Column 2
Row 1 Column 0	Row 1 Column 1	Row 1 Column 2
Row 2 Column 0	Row 2 Column 1	Row 2 Column 2
Row 3 Column 0	Row 3 Column 1	Row 3 Column 2
Row 4 Column 0	Row 4 Column 1	Row 4 Column 2



Phương thức pack()

- Sử dụng thuật toán đóng gói để đặt các widget trong một khung hoặc cửa sổ theo một thứ tự được chỉ định
- Sử dụng để quản lý các widget trong một khối
- Phương thức pack() quản lý các widget trong các hàng hoặc cột. Có các tùy chọn như **expand**, **fill** và **side** để kiểm soát phương thức quản lý hình học này.

Phương thức pack(): Cú pháp



```
widget.pack(options)
```

Trong đó:

- options là các tùy chọn như: **expand, fill và side**
- **expand**: Tùy chọn này sẽ quyết định các widget có được mở rộng để lấp đầy bất kỳ khoảng trống thừa trong tổng thể hình học hay không. Nó có các giá trị Boolean, True hoặc False, giá trị mặc định là False
- **fill**: Nó được sử dụng để lấp đầy các khoảng trống thừa. Giá trị mặc định là NONE, và có giá trị khác như X (lấp theo chiều ngang), Y (lấp theo chiều dọc) hoặc BOTH (lấp theo chiều ngang và dọc)
- **side**: Được sử dụng để chỉ hướng mặt của widget được đóng gói bên trong cửa sổ, có các giá trị TOP (mặc định), BOTTOM, LEFT hoặc RIGHT

Phương thức pack(): Ví dụ 1



```
import tkinter as tk

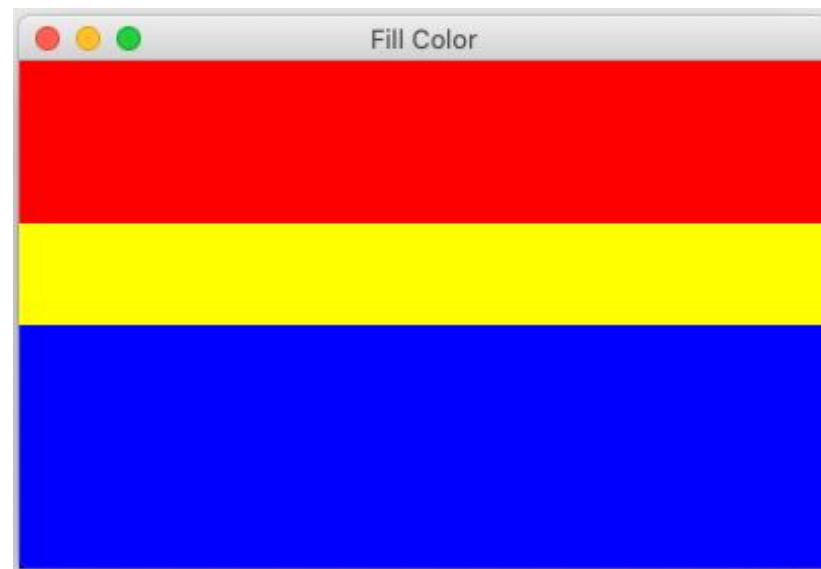
window= tk.Tk()
window.geometry("400x250")
window.title("Fill Color")

frame1 = tk.Frame(master=window,
height=80, bg="red")
frame1.pack(fill=tk.X)

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

frame3 = tk.Frame(master=window,
height=270, bg="blue")
frame3.pack(fill=tk.X)

window.mainloop()
```



Phương thức pack(): Ví dụ 2



Sử dụng pack() với các đối số fill, expand, side

```
import tkinter as tk

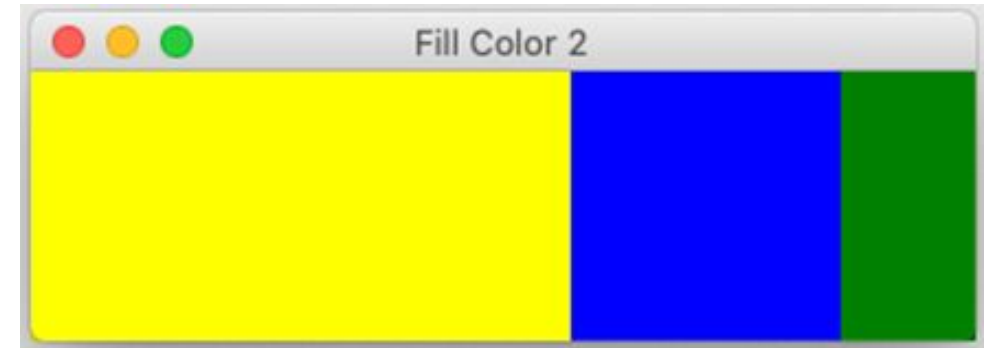
window = tk.Tk()
window.title("Fill Color 2")

frame1 = tk.Frame(master=window, width=200, height=100, bg="Yellow")
# setting fill, side and expand
frame1.pack(fill=tk.BOTH, side=tk.LEFT, expand=True)

frame2 = tk.Frame(master=window, width=100, bg="blue")
frame2.pack(fill=tk.BOTH, side=tk.LEFT, expand=True)

frame3 = tk.Frame(master=window, width=50, bg="green")
frame3.pack(fill=tk.BOTH, side=tk.LEFT, expand=True)

window.mainloop()
```



Sự kiện



- Sự kiện là bất kì hành động nào xảy ra khi kích hoạt một số hành vi trong ứng dụng, chẳng hạn như nhấn nút, nhấn chuột, chuyển động của chuột v.v
- Khi bạn tạo một ứng dụng Tkinter, bạn phải gọi `window.mainloop()` để bắt đầu vòng lặp sự kiện.
- Trong vòng lặp sự kiện, ứng dụng của bạn sẽ kiểm tra xem sự kiện đã xảy ra chưa. Nếu vậy, một số mã có thể được thực thi để đáp ứng.
- Sự kiện được cung cấp dưới dạng chuỗi

Sự kiện: Cú pháp



<modifier-type-detail>

Trong đó

- type là phần thiết yếu chỉ định sự kiện
- modifier và detail không bắt buộc và có thể bỏ qua trong một số trường hợp. Chúng là phần bổ sung cho type đã chọn

Bindings



Trong Tkinter cung cấp một cơ chế mạnh mẽ để cho phép tự giải quyết các sự kiện sinh ra, có thể liên kết các hàm và phương thức Python với các sự kiện

Cú pháp:

```
widget.bind(event, handler)
```

Nếu một sự kiện khớp với mô tả sự kiện xảy ra trong widget, thì hàm **handler** được gọi với một đối tượng mô tả sự kiện

Bindings: Ví dụ 1



Bắt sự kiện click trên cửa sổ

```
from tkinter import *

window = Tk()
window.geometry("400x250")
window.title("Click Event")

def callback(event):
    print("clicked at", event.x, event.y)

frame = Frame(window, width=400, height=250)
frame.bind("<Button-1>", callback)
frame.pack()

window.mainloop()
```

Bindings: Ví dụ 2



Minh họa cách liên kết hàm `return_pressed` với sự kiện nhấn phím `<Return>` của nút `Save`

```
import tkinter as tk
from tkinter import ttk

def return_pressed(event):
    print('Return key pressed.')

root = tk.Tk()

btn = ttk.Button(root, text='Save')
btn.bind('<Return>', return_pressed)

btn.focus()
btn.pack(expand=True)

root.mainloop()
```

Bindings: Ví dụ 3



Minh họa sử dụng bind để đăng kí nhiều trình xử lý cho cùng một sự kiện

```
import tkinter as tk
from tkinter import ttk

def return_pressed(event):
    print('Return key pressed.')

def log(event):
    print(event)
    root = tk.Tk()

btn = ttk.Button(root, text='Save')
btn.bind('<Return>', return_pressed)
btn.bind('<Return>', log, add='+')

btn.focus()
btn.pack(expand=True)

root.mainloop()
```

Phương thức **bind()**

- Phương thức **bind()** mà chúng ta sử dụng tạo ra một ràng buộc cá thể.
- Điều này có nghĩa là **bind** chỉ áp dụng cho một widget duy nhất; nếu bạn tạo các khung mới, chúng sẽ không kế thừa các ràng buộc.
- Nhưng Tkinter cũng cho phép bạn tạo các ràng buộc ở class và ứng dụng. Trên thực tế, bạn có thể tạo ràng buộc ở bốn cấp độ khác nhau.
- Trên widget, sử dụng **bind**
 - Ở cửa sổ cấp trên cùng của widget, cũng dùng **bind**
 - Ở lớp class của widget, sử dụng **bind_class**
 - Toàn bộ ứng dụng, sử dụng **bind_all**



Sự kiện: **<Button>**

<Button>: Một nút chuột được nhấn bằng trỏ chuột trên widget, có các kiểu **<Button>** sau:

- **<Button-1>**: Chỉ định nút chuột trái
 - **<Button-2>**: Chỉ định nút chuột giữa
 - **<Button-3>**: Chỉ định nút chuột phải
 - **<Button-4>**: Chỉ định sự kiện cuộn chuột hướng lên
 - **<Button-5>**: Chỉ định sự kiện cuộn chuột hướng xuống
- Bạn có thể dùng **<ButtonPress>** thay thế cho **<Button>**

Sự kiện: <Motion>

<Motion>: Khi chuột di chuyển với một nút chuột được giữ

- **<B1-Button>**: Tương ứng nút chuột trái
- **<B2-Button>**: Tương ứng nút chuột giữa
- **<B3-Button>**: Tương ứng nút chuột phải

<ButtonRelease>: Được kích hoạt khi một nút chuột được thả ra , để chỉ định nút chuột trái, giữa, phải hãy sử dụng

<ButtonRelease-1>, **<ButtonRelease-2>** và **<ButtonRelease-3>** tương ứng.



Sự kiện: <Double-Button>

<Double-Button>: Tương tự như sự kiện <Button>, nhưng nút được nhấp đúp thay vì nhấp một lần. Bạn có thể sử dụng Double hoặc Triple làm tiền tố.

Lưu ý nếu bạn liên kết cả <Button-1> và <Double-Button-1> thì cả hai liên kết sẽ được gọi

<Enter>: Được kích hoạt khi trỏ chuột vào Widget chỉ định, điều này không có nghĩa người dùng đã nhấn phím Enter

<Leave>: Được kích hoạt khi trỏ chuột rời khỏi Widget chỉ định

Các Sự kiện khác



<MouseWheel>: Kích hoạt khi con lăn chuột được cuộn

<FocusIn>: Được kích hoạt khi phần nhập văn bản từ bàn phím được chuyển đến widget hoặc widget con của widget đó.

<FocusOut>: Được kích hoạt khi phần nhập văn bản từ bàn phím chuyển từ widget này qua widget khác

<Return>: Khi người dùng nhấn phím Enter, bạn có thể liên kết với hầu như tất cả các phím trên bàn phím

<Key>: Khi người dùng nhấn phím bất kì, sẽ tạo ra một đối tượng sự kiện gọi lại (đây là một chuỗi trống cho các key đặc biệt)

<Shift-Up>: Khi người dùng nhấn phím mũi tên lên, trong khi nhấn giữ phím Shift, bạn có thể sử dụng các tiền tố Alt, Shift, Control

<Configure>: Kích hoạt khi cấu hình của tiện ích được chỉ định có những thay đổi chẳng hạn như chiều rộng, đường viền được điều chỉnh



Các thuộc tính của Sự kiện

widget : Đây là 1 phiên bản widget Tkinter hợp lệ, không phải là một cái tên. Thuộc tính này được đặt cho tất cả sự kiện

x,y : Vị trí của con chuột, tính theo pixel

x_root, y_root : Vị trí con chuột hiện tại so với góc bên trái của màn hình, tính theo pixel

char: Mã ký tự của bàn phím, dưới dạng một chuỗi(chỉ sự kiện từ bàn phím)

keysym: Biểu tượng phím(chỉ sự kiện từ bàn phím)

keycode: Mã phím(chỉ sự kiện từ bàn phím)

num: Số nút (chỉ sự kiện từ nút chuột)

width,height: Kích thước mới của widget, tính bằng pixel (chỉ sự kiện **Configure**)

type : Loại sự kiện



Tóm tắt bài học

- Tkinter là thư viện GUI tiêu chuẩn cho Python. Khi kết hợp với Tkinter, Python sẽ được cung cấp các công cụ một cách nhanh chóng và dễ dàng để tạo các ứng dụng GUI.
- Place() sử dụng để sắp xếp các widget trong cửa sổ chính, đặt chúng vào một vị trí cụ thể theo chỉ dẫn của người lập trình
- Grid() sử dụng các khái niệm về hàng và cột để sắp xếp các widget con
- Trong Tkinter cung cấp một cơ chế mạnh mẽ để cho phép tự giải quyết các sự kiện sinh ra, có thể liên kết các hàm và phương thức Python với các sự kiện
- Sự kiện là bất kì hành động nào xảy ra khi kích hoạt một số hành vi trong ứng dụng, chẳng hạn như nhấn nút, nhấn chuột, chuyển động của chuột v.v

Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo