BÁO CÁO PYTHON NÂNG CAO

1. Giải đoạn code

Dự án này xây dựng một ứng dụng GUI bằng Python sử dụng thư viện tkinter. Chương trình tạo giao diện người dùng với các thành phần tương tác như: label, combobox, spinbox, nút bấm và các tab. Chương trình cũng sử dụng các thành phần nâng cao như ToolTip để cung cấp hướng dẫn sử dụng khi người dùng di chuột qua các thành phần khác nhau.

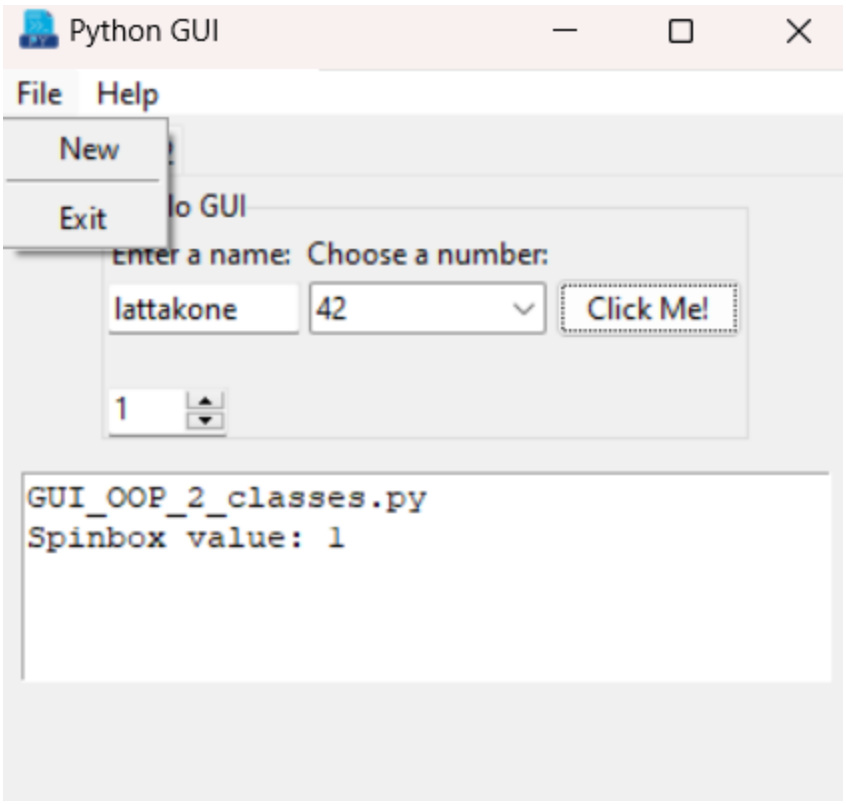
Các bước thực hiện:

- **Bước 1:** Thiết lập ứng dụng chính bằng cách khởi tạo class App để quản lý toàn bộ giao diện và logic.
- **Bước 2:** Tạo menu chính gồm hai menu con là "File" và "Help".
- **Bước 3:** Tạo hai tab chính trong giao diện người dùng, mỗi tab chứa các thành phần khác nhau để người dùng tương tác.
- **Bước 4:** Sử dụng ToolTip để hiển thị chú thích cho các thành phần khi người dùng di chuột qua chúng.
- **Bước 5:** Quản lý các sự kiện khi người dùng tương tác với các nút bấm và các điều khiển khác.
- **Bước 6:** Sử dụng class ToolTip để hiển thị thông tin khi người dùng di chuột vào các điều khiển tương ứng.

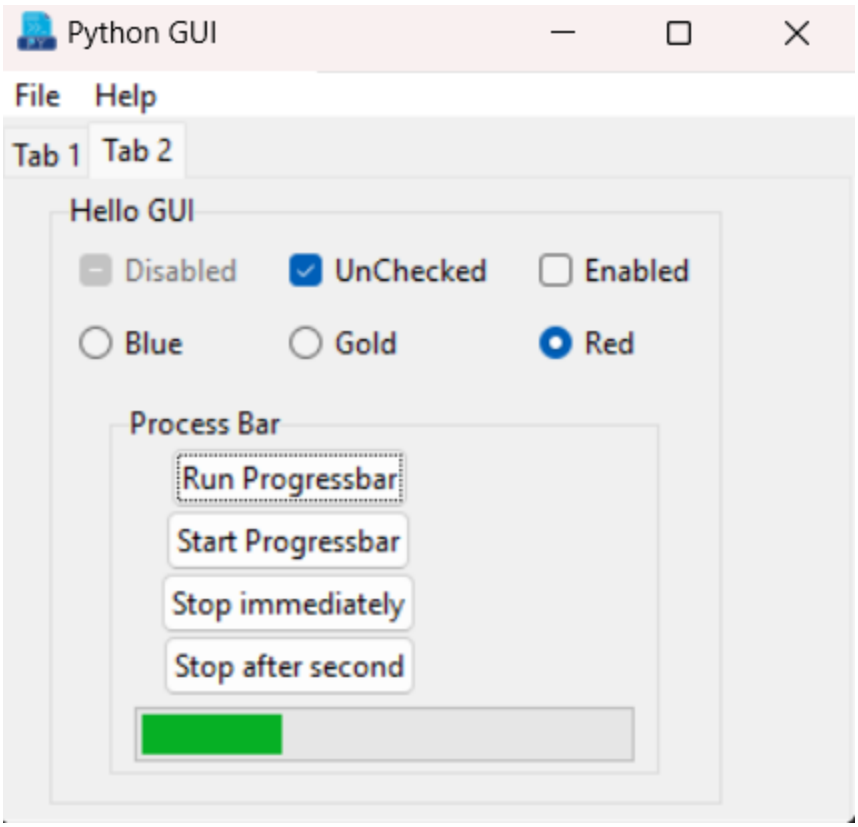
2. Chức năng của code

Ứng dụng cung cấp giao diện người dùng với các chức năng chính sau:

- **Tab 1:**
    - Người dùng có thể nhập tên vào ô nhập liệu.
    - Chọn một số từ combobox.
    - Sử dụng spinbox để chọn giá trị từ -1000 đến 1000.
    - Nút "Click Me!" khi bấm sẽ hiển thị giá trị của spinbox trong hộp văn bản.



- **Tab 2:**
    - Người dùng có thể chọn các trạng thái khác nhau từ các checkbutton.
    - Sử dụng các radiobutton để chọn màu sắc.
    - Thanh tiến trình (ProgressBar) cho phép người dùng bắt đầu, dừng và điều khiển tiến trình.



- **Tooltip:**
    - Khi người dùng di chuột vào các thành phần, một cửa sổ nhỏ sẽ hiện lên để hiển thị chú thích hướng dẫn người dùng.
- **Menu:**
    - Menu "File" chứa các tùy chọn để tạo mới và thoát chương trình.
    - Menu "Help" chứa tùy chọn để hiển thị thông tin về chương trình.

3. Mã nguồn

```
import tkinter as tk
from tkinter import ttk
```

```
from tkinter import messagebox

class ToolTip:
    def __init__(self, widget, text):
        self.widget = widget
        self.text = text
        self.tooltip_window = None
        self.widget.bind("<Enter>",
self.show_tooltip)
        self.widget.bind("<Leave>",
self.hide_tooltip)

    def show_tooltip(self, event=None):
        x, y, _, _ =
self.widget.bbox("insert")
        x += self.widget.winfo_rootx() + 25
        y += self.widget.winfo_rooty() + 25
        self.tooltip_window = tw =
tk.Toplevel(self.widget)
        tw.wm_overrideredirect(True)  #
Remove window decorations
        tw.wm_geometry(f"+{x}+{y}")
```

```
        label = tk.Label(tw,
text=self.text, justify='left',
                         background="lighty
ellow", relief='solid', borderwidth=1)
        label.pack(ipadx=1)

    def hide_tooltip(self, event=None):
        if self.tooltip_window:
            self.tooltip_window.destroy()
            self.tooltip_window = None

class App:
    def __init__(self, root):
        self.root = root
        self.root.title("Python GUI")
        self.root.iconbitmap('D:/Lập trình
Python nâng cao/Lab3/my_icon.ico')  #
Replace with a valid icon path

        self.create_menu()
        self.create_tabs()

    def create_menu(self):
        menu_bar = tk.Menu(self.root)
```

```
        self.root.config(menu=menu_bar)

        # File menu
        file_menu = tk.Menu(menu_bar,
tearoff=0)
        file_menu.add_command(label="New")
        file_menu.add_separator()
        file_menu.add_command(label="Exit",
command=self.on_exit)
        menu_bar.add_cascade(label="File",
menu=file_menu)

        # Help menu
        help_menu = tk.Menu(menu_bar,
tearoff=0)
        help_menu.add_command(label="About"
, command=self.show_about)
        menu_bar.add_cascade(label="Help",
menu=help_menu)

    def create_tabs(self):
        """Create tab control with two
tabs."""
```

```
        tab_control =
ttk.Notebook(self.root)

        # Tab 1
        tab1 = ttk.Frame(tab_control)
        tab_control.add(tab1, text='Tab 1')

        self.create_tab1_widgets(tab1)

        # Tab 2
        tab2 = ttk.Frame(tab_control)
        tab_control.add(tab2, text='Tab 2')
        self.create_tab2_widgets(tab2)

        tab_control.pack(expand=1,
fill="both")

    def create_tab1_widgets(self, parent):
        """Create widgets for Tab 1."""
        label_frame =
ttk.LabelFrame(parent, text="Hello GUI")
        label_frame.grid(column=0, row=0,
padx=8, pady=4)
```

```
        # Label and Entry for name
        ttk.Label(label_frame, text="Enter
a name:").grid(column=0, row=0, sticky='W')
        self.name = tk.StringVar()
        self.name_entry =
ttk.Entry(label_frame, width=12,
textvariable=self.name)
        self.name_entry.grid(column=0,
row=1, sticky='W')
        ToolTip(self.name_entry, "Enter
your name here")

        # Combobox for number selection
        ttk.Label(label_frame, text="Choose
a number:").grid(column=1, row=0,
sticky='W')
        self.number = tk.StringVar()
        self.number_chosen =
ttk.Combobox(label_frame, width=12,
textvariable=self.number, state='readonly')
        self.number_chosen['values'] = (1,
2, 4, 42, 100, 300)
        self.number_chosen.grid(column=1,
row=1)
```

```
        self.number_chosen.current(0)
        ToolTip(self.number_chosen, "Choose
a number from the dropdown")

        # Spinbox for value selection
        ttk.Label(label_frame).grid(column=
0, row=2, sticky='W')
        self.spin_value = tk.IntVar()
        self.spinbox =
ttk.Spinbox(label_frame, from_=-1000,
to=1000, textvariable=self.spin_value,
width=5)
        self.spinbox.grid(column=0, row=3,
sticky='W')
        ToolTip(self.spinbox, "Spin to
select a value")

        # Button with click action
        self.action_button =
ttk.Button(label_frame, text="Click Me!",
command=self.on_button_click)
        self.action_button.grid(column=2,
row=1)
```

```
        ToolTip(self.action_button, "Click
to execute action")

        # Text box for displaying
information
        self.text_box = tk.Text(parent,
height=5, width=40)
        self.text_box.grid(column=0, row=2,
padx=8, pady=8)

    def create_tab2_widgets(self, tab2):
        """Create widgets for Tab 2."""
        # LabelFrame for grouping widgets
        label_frame = ttk.LabelFrame(tab2,
text="Hello GUI")
        label_frame.grid(column=0, row=0,
padx=20, pady=5)

        # Checkbuttons


        chk_state_disabled =
tk.BooleanVar()
```

```
        chk_disabled =
ttk.Checkbutton(label_frame,
text='Disabled', var=chk_state_disabled,
state='disabled')
        chk_disabled.grid(column=0, row=0,
padx=8, pady=4, sticky='W')

        chk_state_unchecked =
tk.BooleanVar()
        chk_unchecked =
ttk.Checkbutton(label_frame,
text='UnChecked', var=chk_state_unchecked)
        chk_unchecked.grid(column=1, row=0,
padx=8, pady=4, sticky='W')

        chk_state_enabled =
tk.BooleanVar(value=True)
        chk_enabled =
ttk.Checkbutton(label_frame,
text='Enabled', var=chk_state_enabled)
        chk_enabled.grid(column=2, row=0,
padx=8, pady=4, sticky='W')

        # Radiobuttons for color choice
```

```
        color = tk.StringVar(value='Gold')
        blue_radio =
ttk.Radiobutton(label_frame, text='Blue',
value='Blue', variable=color)
        gold_radio =
ttk.Radiobutton(label_frame, text='Gold',
value='Gold', variable=color)
        red_radio =
ttk.Radiobutton(label_frame, text='Red',
value='Red', variable=color)
        blue_radio.grid(column=0, row=1,
padx=8, pady=4, sticky='W')
        gold_radio.grid(column=1, row=1,
padx=8, pady=4, sticky='W')
        red_radio.grid(column=2, row=1,
padx=8, pady=4, sticky='W')

        ToolTip(blue_radio, "This is a
Radiobutton control")
        ToolTip(gold_radio, "This is a
Radiobutton control")
        ToolTip(red_radio, "This is a
Radiobutton control")
```

```
        # Progress bar inside the same
label frame
        label_frame2 =
ttk.LabelFrame(label_frame, text="Process
Bar")
        label_frame2.grid(column=0, row=2,
columnspan=3, padx=20, pady=10)

        # Progress bar
        progress_bar =
ttk.Progressbar(label_frame2,
orient='horizontal', length=200,
mode='determinate')
        progress_bar.grid(column=0, row=6,
columnspan=3, padx=8, pady=4)

        def run_progressbar():
            progress_bar.start(10)

        def stop_immediately():
            progress_bar.stop()

        def stop_after_second():
```

```
            self.root.after(1000,
progress_bar.stop)

        # Control buttons
        run_button =
ttk.Button(label_frame2, text="Run
Progressbar", command=run_progressbar)
        run_button.grid(column=0, row=0,
padx=0, pady=0)

        start_button =
ttk.Button(label_frame2, text="Start
Progressbar", command=lambda:
progress_bar.step(25))
        start_button.grid(column=0, row=1,
padx=0, pady=0)

        stop_button =
ttk.Button(label_frame2, text="Stop
immediately", command=stop_immediately)
        stop_button.grid(column=0, row=2,
padx=0, pady=0)
```

```
        stop_after_sec_button =
ttk.Button(label_frame2, text="Stop after
second", command=stop_after_second)
        stop_after_sec_button.grid(column=0
, row=3, padx=0, pady=0)

    def on_button_click(self):
        """Handle the button click
event."""
        spin_value = self.spin_value.get()
        output =
f"GUI_OOP_2_classes.py\nSpinbox value:
{spin_value}\n"
        self.text_box.insert(tk.INSERT,
output)

    def on_exit(self):
        """Handle the exit event."""
        self.root.quit()

    def show_about(self):
        """Show the 'About' message box."""
```

```
        messagebox.showinfo("About", "This
is a Python GUI example.\nCreated using
Tkinter.")

# Main function to run the application
if __name__ == "__main__":
    root = tk.Tk()
    app = App(root)
    root.mainloop()
```

4. Đưa lên GitHub

https://github.com/Lattakone1/Python_NangCao/blob/main/Project_Bc_Python_n%C3%A2ng%20cao/CodeProject.py