Python TUTORIAL

Tkinter Object-Oriented Frames

Summary: in this tutorial, you'll learn how to inherit (https://www.pythontutorial.net/python-oop/python-inheritance/) from the `ttk.Frame` class and use it in the root window.

In the previous tutorial (https://www.pythontutorial.net/tkinter/tkinter-object-oriented-window/) , you've learned how to subclass the `Tkinter.Tk` class. However, a Tkinter application should have only one `Tk` instance.

Therefore, it's common to inherit from the `ttk.Frame` (https://www.pythontutorial.net/tkinter/tkinter-frame/) class and use the subclass in the root window.

To inherit the `ttk.Frame` class, you use the following syntax:

```
class MainFrame(ttk.Frame):
    pass
```

Since a `Frame` needs a container, you need to add an argument to its `__init__()` method and call the `__init__()` method of the `ttk.Frame` class like this:

```
class MainFrame(ttk.Frame):
    def __init__(self, container):
        super().__init__(container)
```

The following shows the complete `MainFrame` class that has a label (https://www.pythontutorial.net/tkinter/tkinter-label/) and a button (https://www.pythontutorial.net/tkinter/tkinter-button/) . When you click the button, it shows a message box (https://www.pythontutorial.net/tkinter/tkinter-messagebox/) :

```
class MainFrame(ttk.Frame):
    def __init__(self, container):
        super().__init__(container)

        options = {'padx': 5, 'pady': 5}

        # label
        self.label = ttk.Label(self, text='Hello, Tkinter!')
        self.label.pack(**options)

        # button
        self.button = ttk.Button(self, text='Click Me')
        self.button['command'] = self.button_clicked
        self.button.pack(**options)

        # show the frame on the container
        self.pack(**options)

    def button_clicked(self):
        showinfo(title='Information',
                 message='Hello, Tkinter!')
```

The following defines an `App` class that inherits from the `Tk` class:

```
class App(tk.Tk):
    def __init__(self):
        super().__init__()
        # configure the root window
        self.title('My Awesome App')
        self.geometry('300x100')
```

And you can bootstrap the application via the `if __name__ == "__main__"` block.

```
if __name__ == "__main__":
    app = App()
    frame = MainFrame(app)
    app.mainloop()
```

In this code:

- First, create a new instance of the `App` class.
- Second, create a new instance of the `MainFrame` class and set its container to the app instance.
- Third, start the application by calling the app(). It'll execute the `__call__()` method that will invoke the `mainloop()` of the root window.

Put it all together:

```
import tkinter as tk
from tkinter import ttk
from tkinter.messagebox import showinfo



class MainFrame(ttk.Frame):
    def __init__(self, container):
        super().__init__(container)

        options = {'padx': 5, 'pady': 5}

        # label
        self.label = ttk.Label(self, text='Hello, Tkinter!')
        self.label.pack(**options)

        # button
        self.button = ttk.Button(self, text='Click Me')
        self.button['command'] = self.button_clicked
        self.button.pack(**options)

        # show the frame on the container
        self.pack(**options)

    def button_clicked(self):
        showinfo(title='Information',
                 message='Hello, Tkinter!')


class App(tk.Tk):
    def __init__(self):
        super().__init__()
        # configure the root window
        self.title('My Awesome App')
        self.geometry('300x100')


if __name__ == "__main__":
    app = App()
    frame = MainFrame(app)
    app.mainloop()
```

Output:

More Object-oriented Frame example

The following example uses the classes to convert the **Replace** window from the `Frame` tutorial (https://www.pythontutorial.net/tkinter/tkinter-frame/) :

```
import tkinter as tk
from tkinter import ttk


class InputFrame(ttk.Frame):
    def __init__(self, container):
        super().__init__(container)
        # setup the grid layout manager
        self.columnconfigure(0, weight=1)
        self.columnconfigure(0, weight=3)

        self.__create_widgets()

    def __create_widgets(self):
        # Find what
        ttk.Label(self, text='Find what:').grid(column=0, row=0, sticky=tk.W)
        keyword = ttk.Entry(self, width=30)
        keyword.focus()
        keyword.grid(column=1, row=0, sticky=tk.W)

        # Replace with:
        ttk.Label(self, text='Replace with:').grid(
            column=0, row=1, sticky=tk.W)
        replacement = ttk.Entry(self, width=30)
        replacement.grid(column=1, row=1, sticky=tk.W)
        # Match Case checkbox
        match_case = tk.StringVar()
        match_case_check = ttk.Checkbutton(
            self,
            text='Match case',
            variable=match_case,
            command=lambda: print(match_case.get()))
        match_case_check.grid(column=0, row=2, sticky=tk.W)

        # Wrap Around checkbox
        wrap_around = tk.StringVar()
        wrap_around_check = ttk.Checkbutton(
            self,
            variable=wrap_around,
            text='Wrap around',
            command=lambda: print(wrap_around.get()))
        wrap_around_check.grid(column=0, row=3, sticky=tk.W)

        for widget in self.winfo_children():
            widget.grid(padx=0, pady=5)



class ButtonFrame(ttk.Frame):
    def __init__(self, container):
        super().__init__(container)
        # setup the grid layout manager
        self.columnconfigure(0, weight=1)

        self.__create_widgets()

    def __create_widgets(self):
        ttk.Button(self, text='Find Next').grid(column=0, row=0)
        ttk.Button(self, text='Replace').grid(column=0, row=1)
        ttk.Button(self, text='Replace All').grid(column=0, row=2)
        ttk.Button(self, text='Cancel').grid(column=0, row=3)
        for widget in self.winfo_children():
            widget.grid(padx=0, pady=3)



class App(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title('Replace')
        self.geometry('400x150')
        self.resizable(0, 0)
        # windows only (remove the minimize/maximize button)
        self.attributes('-toolwindow', True)

        # layout on the root window
        self.columnconfigure(0, weight=4)
        self.columnconfigure(1, weight=1)

        self.__create_widgets()

    def __create_widgets(self):
        # create the input frame
        input_frame = InputFrame(self)
        input_frame.grid(column=0, row=0)

        # create the button frame
        button_frame = ButtonFrame(self)
        button_frame.grid(column=1, row=0)



if __name__ == "__main__":
    app = App()
    app.mainloop()
```

Summary

- Subclass the `ttk.Frame` and initialize the widgets on the frame.
- Use the subclass of the `ttk.Frame` in a root window.