Developing a Full Tkinter Object-Oriented Application

If this Python Tutorial saves you hours of work, please whitelist it in your ad blocker 😭 and

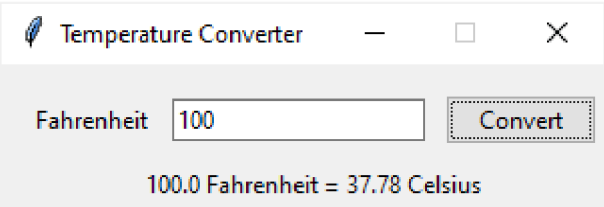
Donate Now

(https://www.pythontutorial.net/donation/)

to help us ❤️ pay for the web hosting fee and CDN to keep the website running.

Summary: in this tutorial, you'll learn how to develop a full Tkinter object-oriented application.

You'll convert the temperature converter application (https://www.pythontutorial.net/tkinter/tkinter-example/) to a new one that uses object-oriented programming approach (https://www.pythontutorial.net/python-oop/) :



First, define a class called `TemperatureConverter` . The class has one static method (https://www.pythontutorial.net/python-oop/python-static-methods/) that converts a temperature from Fahrenheit to Celsius:

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

```
class TemperatureConverter:
    @staticmethod
    def fahrenheit_to_celsius(f):
        return (f - 32) * 5 / 9
```

Second, define a `ConverterFrame` class that inherits from the `ttk.Frame` class. The `ConverterFrame` class will be responsible for creating widgets and handling events:

```
class ConverterFrame(ttk.Frame):
    def __init__(self, container):
        super().__init__(container)
        # field options
        options = {'padx': 5, 'pady': 5}

        # temperature label
        self.temperature_label = ttk.Label(self, text='Fahrenheit')
        self.temperature_label.grid(column=0, row=0, sticky=tk.W, **options)

        # temperature entry
        self.temperature = tk.StringVar()
        self.temperature_entry = ttk.Entry(self, textvariable=self.temperature
        self.temperature_entry.grid(column=1, row=0, **options)
        self.temperature_entry.focus()

        self.convert_button = ttk.Button(self, text='Convert')
        self.convert_button['command'] = self.convert
        self.convert_button.grid(column=2, row=0, sticky=tk.W, **options)

        # result label
        self.result_label = ttk.Label(self)
        self.result_label.grid(row=1, columnspan=3, **options)

        # add padding to the frame and show it
        self.grid(padx=10, pady=10, sticky=tk.NSEW)
```

```
    def convert(self):
        """  Handle button click event
        """
        try:
            f = float(self.temperature.get())
            c = TemperatureConverter.fahrenheit_to_celsius(f)
            result = f'{f} Fahrenheit = {c:.2f} Celsius'
            self.result_label.config(text=result)
        except ValueError as error:
            showerror(title='Error', message=error)
```

How it works:

- The `ConverterFrame` needs a container, therefore, its `__init__()` method has the `container` argument.
- Inside the `__init__()` method of the `ConverterCFrame` class, call the `__init__()` method of its superclass.
- Assign the widgets to the `self` object so that you can reference them in other methods of the `ConverterFrame` class.
- Assign the `command` option of the `convert` button to the `self.convert` method.

Third, define an `App` class that inherits from the `tk.Tk` class:

```
class App(tk.Tk):
    def __init__(self):
        super().__init__()

        self.title('Temperature Converter')
        self.geometry('300x70')
        self.resizable(False, False)
```

Finally, bootstrap the application from the `if __name__ == "__main__"` block:

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

Put it all together:

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


class TemperatureConverter:
    @staticmethod
    def fahrenheit_to_celsius(f):
        return (f - 32) * 5 / 9


class ConverterFrame(ttk.Frame):
    def __init__(self, container):
        super().__init__(container)
        # field options
        options = {'padx': 5, 'pady': 5}

        # temperature label
        self.temperature_label = ttk.Label(self, text='Fahrenheit')
        self.temperature_label.grid(column=0, row=0, sticky=tk.W, **options)

        # temperature entry
        self.temperature = tk.StringVar()
        self.temperature_entry = ttk.Entry(self, textvariable=self.temperature
        self.temperature_entry.grid(column=1, row=0, **options)
        self.temperature_entry.focus()

        self.convert_button = ttk.Button(self, text='Convert')
```

```
        self.convert_button['command'] = self.convert
        self.convert_button.grid(column=2, row=0, sticky=tk.W, **options)

        # result label
        self.result_label = ttk.Label(self)
        self.result_label.grid(row=1, columnspan=3, **options)

        # add padding to the frame and show it
        self.grid(padx=10, pady=10, sticky=tk.NSEW)

    def convert(self):
        """  Handle button click event
        """
        try:
            f = float(self.temperature.get())
            c = TemperatureConverter.fahrenheit_to_celsius(f)
            result = f'{f} Fahrenheit = {c:.2f} Celsius'
            self.result_label.config(text=result)
        except ValueError as error:
            showerror(title='Error', message=error)


class App(tk.Tk):
    def __init__(self):
        super().__init__()

        self.title('Temperature Converter')
        self.geometry('300x70')
        self.resizable(False, False)


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

In this tutorial, you have learned how to develop a full object-oriented Tkinter application.