**Python Tkinter**

# What is Tkinter for Python?

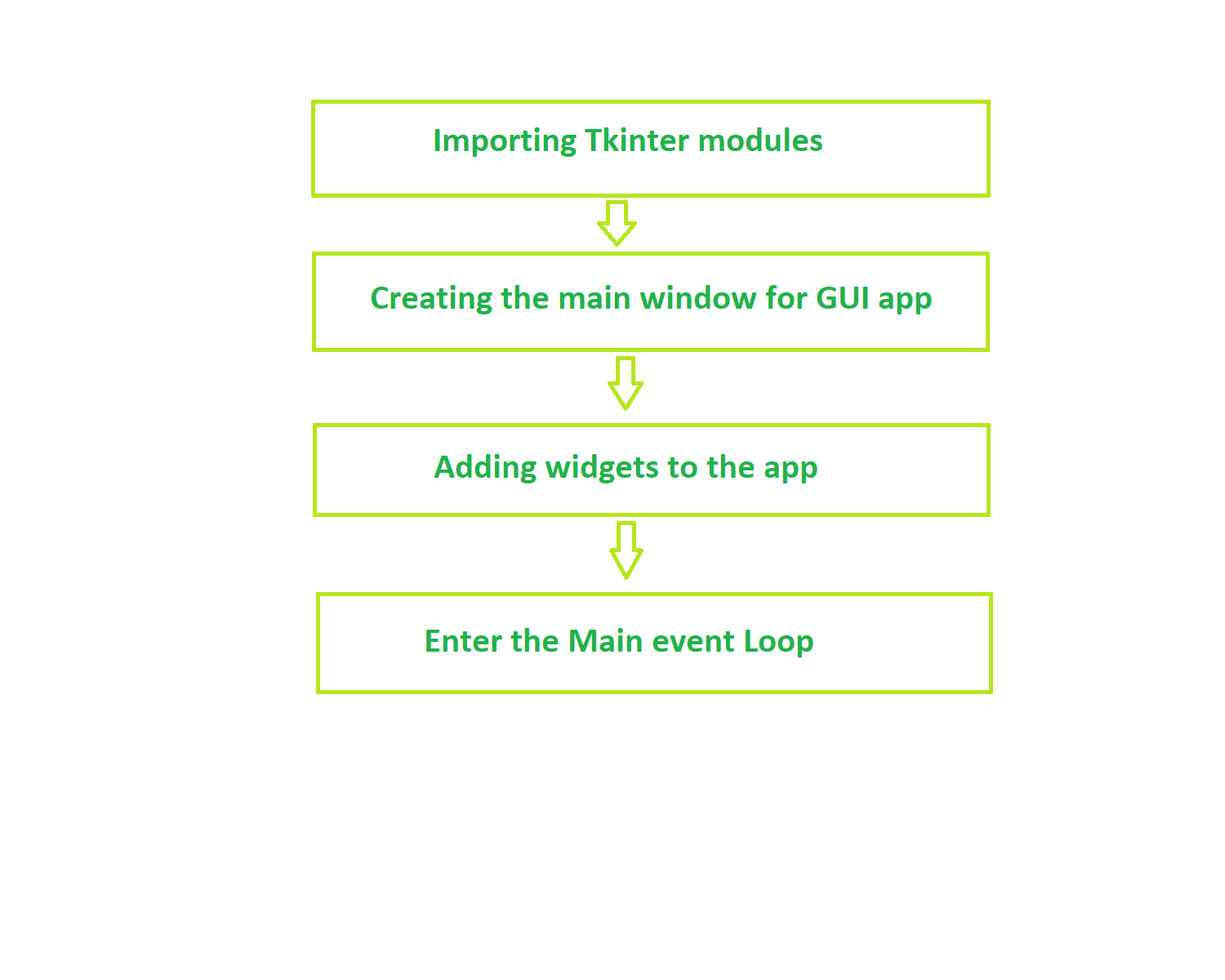
* **Tkinter** is a standard [Python GUI (Graphical User Interface) library](https://www.geeksforgeeks.org/python3-gui-application-overview/) that provides a set of tools and widgets to create desktop applications with graphical interfaces. Tkinter is included with most Python installations, making it easily accessible for developers who want to build GUI applications without requiring additional installations or libraries.
* The name "**Tkinter**" comes from "**Tk interface**", referring to the Tk GUI toolkit that Tkinter is based on. Tkinter provides a way to create windows, buttons, labels, text boxes, and other GUI components to build interactive applications.

## Where is Python Tkinter used?

Here are some common use cases for Tkinter:

1. **Creating windows and dialog boxes**: Tkinter can be used to create windows and dialog boxes that allow users to interact with your program. These can be used to display information, gather input, or present options to the user.
2. **Building a GUI for a desktop application**: Tkinter can be used to create the interface for a desktop application, including buttons, menus, and other interactive elements.
3. **Adding a GUI to a command-line program**: Tkinter can be used to add a GUI to a command-line program, making it easier for users to interact with the program and input arguments.
4. **Creating custom widgets**: Tkinter includes a variety of built-in widgets, such as buttons, labels, and text boxes, but it also allows you to create your own custom widgets.
5. **Prototyping a GUI**: Tkinter can be used to quickly prototype a GUI, allowing you to test and iterate on different design ideas before committing to a final implementation.

## ****Fundamental structure of Tkinter program****



# What are Widgets in Tkinter?

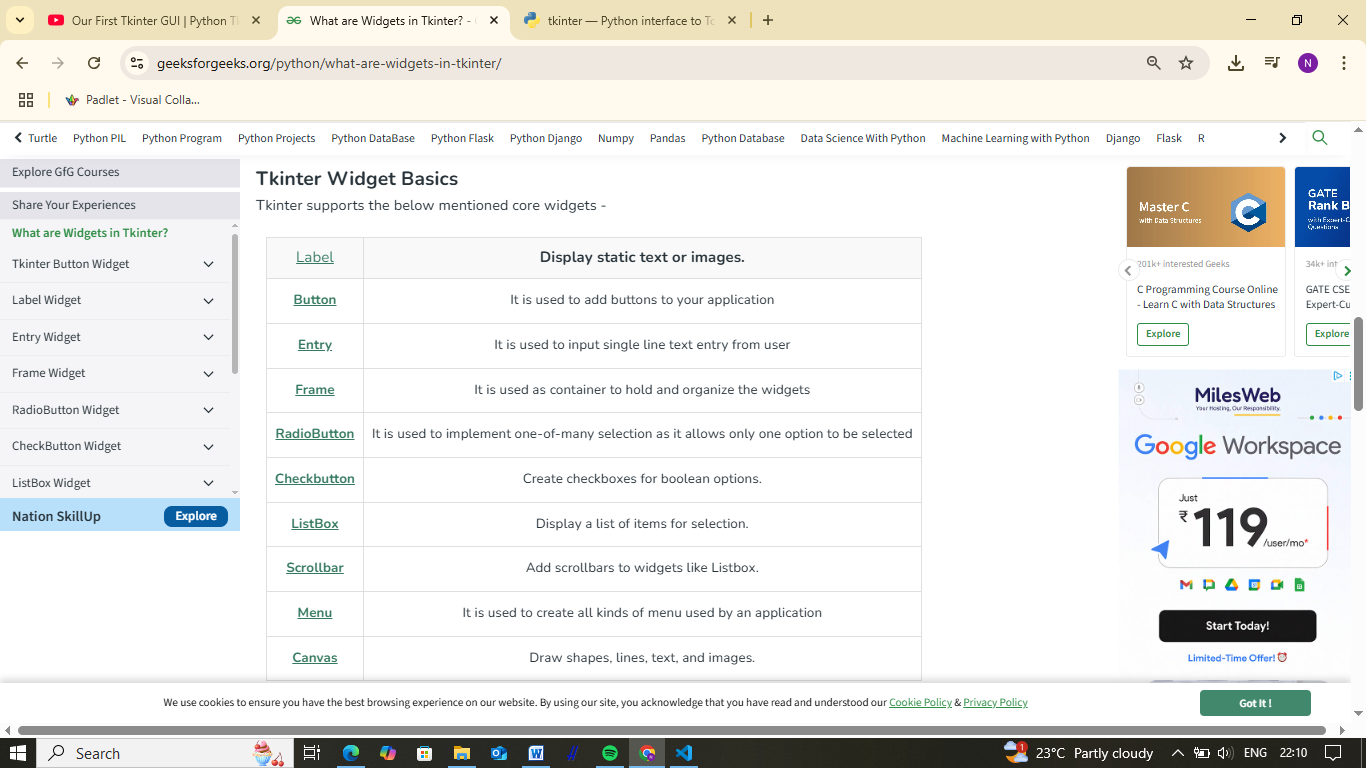
# **tkinter**we can use to build out interfaces- such as buttons, menus, **interfaces,** and various kind of entry fields and display areas. We call these elements **Widgets**.

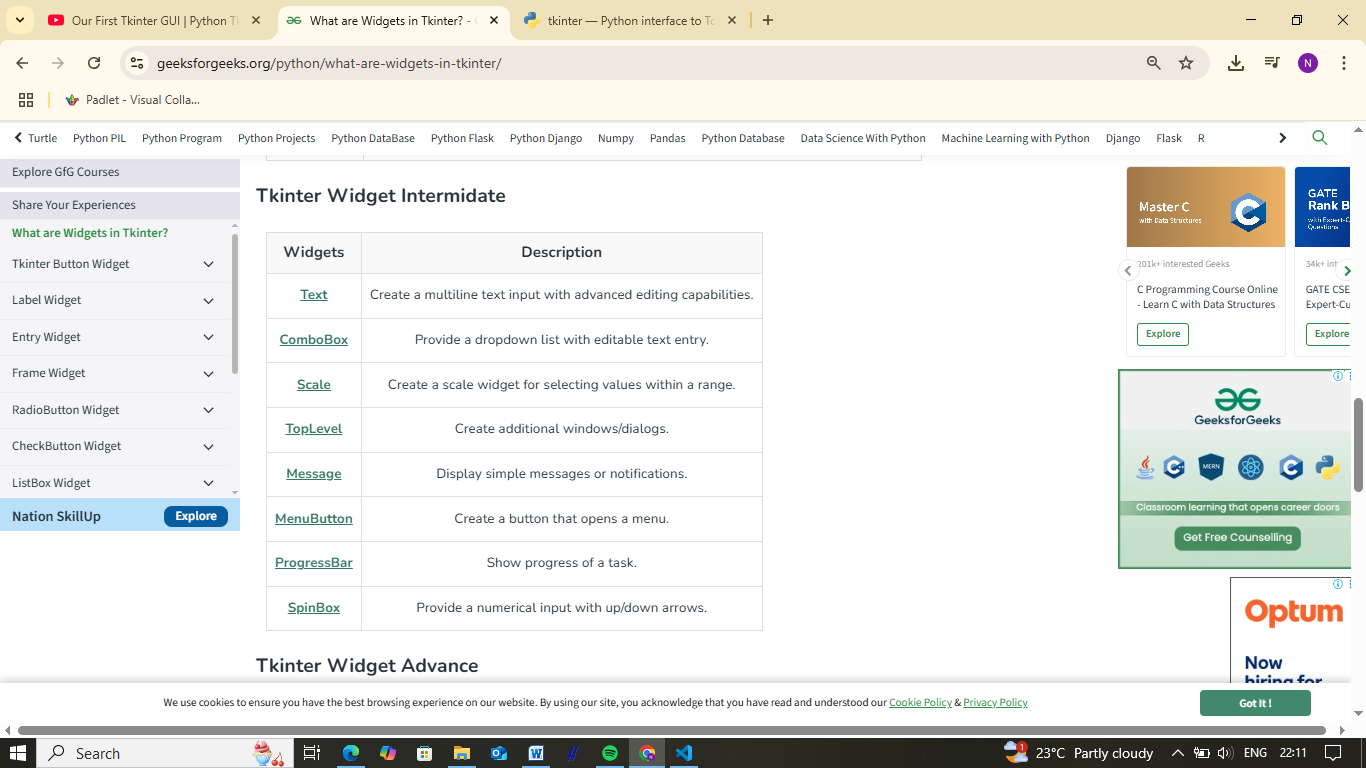
# In Tkinter, a widget is essentially a graphical component that the user can interact with. They can range from simple elements like buttons and labels to more complex ones like text entry fields, listboxes, and canvases. Each widget serves a specific purpose and can be customized to fit the design and functionality requirements of your application.

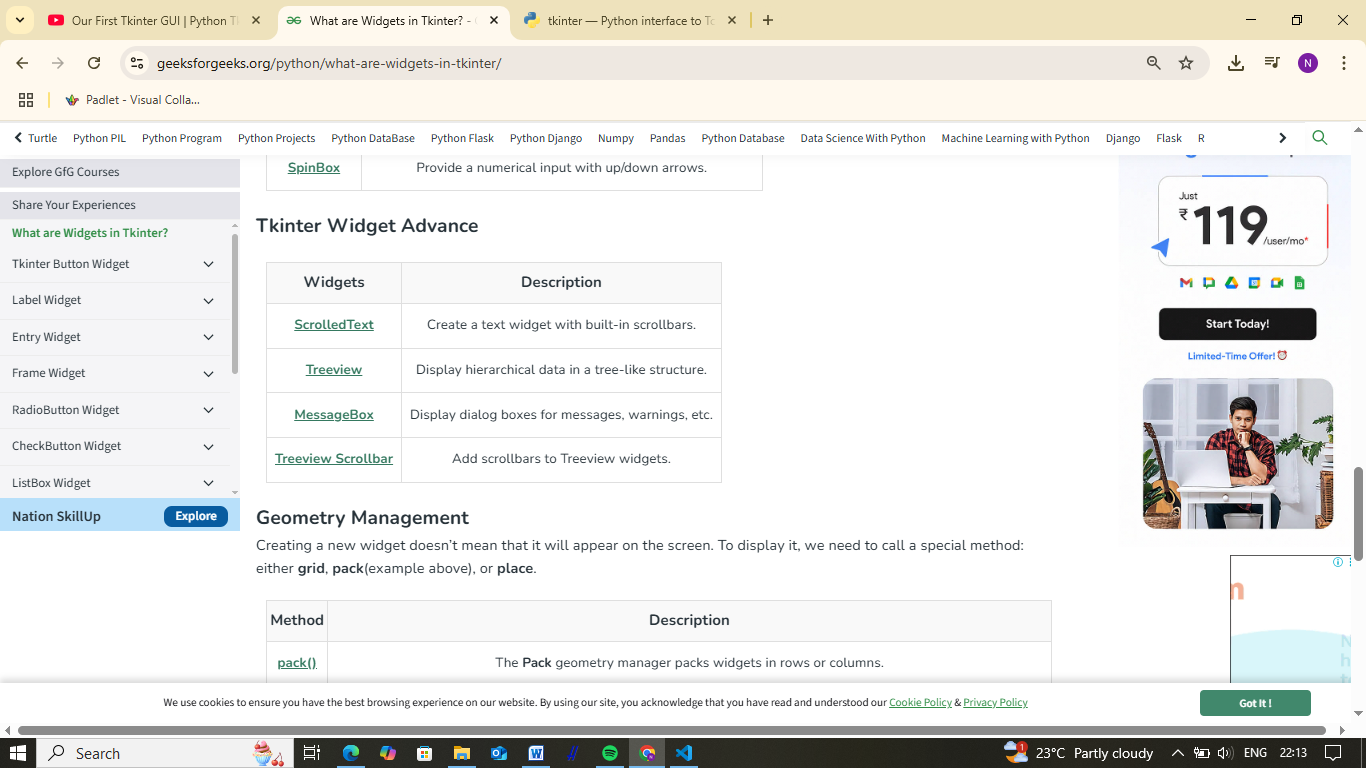
## ****How Do Tkinter Widgets Work?****

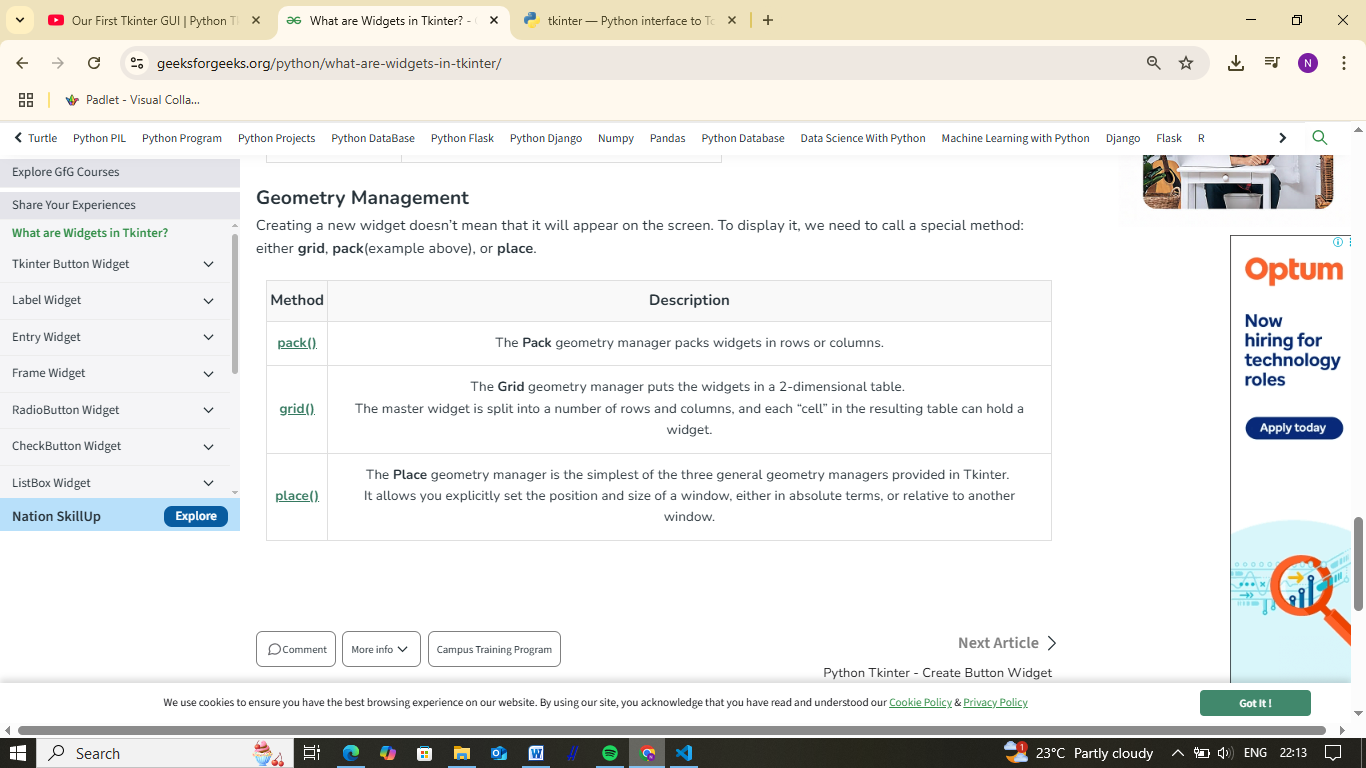
Each widget in Tkinter is an instance of a specific class defined in the Tkinter Module. These classes provide methods and attributes that allow you to configure the widget's appearance, behavior, and functionality. Widgets are typically added to the application's window or frames using layout managers like pack(), grid(), or place(), which determine their position and size within the interface.

**Example:** VS CODE “example.py”









# Python Tkinter - Create Button Widget

The Tkinter Button widget is a graphical control element used in Python's Tkinter library to create clickable buttons in a graphical user interface (GUI). It provides a way for users to trigger actions or events when clicked.

## Tkinter Button Widget Syntax

The syntax to use the button widget is given below.

***Syntax:*** *Button(parent, options)*

***Parameters***

* ***parent****: This parameter is used to represents the parent window.*
* ***options****:There are many options which are available and they can be used as key-value pairs separated by commas.*

### ****Tkinter****Button****Options****

* **activebackground**: Background color when the button is under the cursor.
* **activeforeground**: Foreground color when the button is under the cursor.
* **anchor**: Width of the border around the outside of the button
* **bd or borderwidth**: Width of the border around the outside of the button
* **bg or background:** Normal background color.
* **command**: Function or method to be called when the button is clicked.
* **cursor**: Selects the cursor to be shown when the mouse is over the button.
* **text**: Text displayed on the button.
* **disabledforeground**: Foreground color is used when the button is disabled.
* **fg or foreground:** Normal foreground (text) color.
* **font**: Text font to be used for the button's label.
* **height**: Height of the button in text lines
* **highlightbackground**: Color of the focus highlight when the widget does not have focus.
* **highlightcolor**: The color of the focus highlight when the widget has focus.
* **highlightthickness**: Thickness of the focus highlight.
* **image**: Image to be displayed on the button (instead of text).
* **justify**: tk.LEFT to left-justify each line; tk.CENTER to center them; or tk.RIGHT to right-justify.
* **overrelief**: The relief style to be used while the mouse is on the button; default relief is tk.RAISED.
* **padx**, **pady**: padding left and right of the text. / padding above and below the text.
* **width**: Width of the button in letters (if displaying text) or pixels (if displaying an image).
* **underline**: Default is -1, underline=1 would underline the second character of the button's text.
* **width**: Width of the button in letters
* **wraplength**: If this value is set to a positive number, the text lines will be wrapped to fit within this length.

### ****Methods****

1. **flash()**: Causes the button to flash several times between active and normal colors. Leaves the button in the state it was in originally. Ignored if the button is disabled.
2. **invoke()**: Calls the button's command callback, and returns what that function returns. Has no effect if the button is disabled or there is no callback.

## Creating a Button using Tkinter

GO TO :- VS CODE “CREATE BUTTON.py”

## ****Creation of Button without using TK Themed Widget :-****

Creation of Button using **tk** themed widget (tkinter.ttk). This will give you the effects of modern graphics. Effects will change from one OS to another because it is basically for the appearance.

Example :-

**Button without using TK Themed Widget.py**

## Steps to Check which Button was clicked in Tkinter :-

**Step 1:** First, import the library Tkinter.

from tkinter import \*

**Step 2:** Now, create a GUI app using Tkinter.

app = Tk()

**Step 3:**Then, create a function with one parameter, i.e., of the text you want to show when a button is clicked

Def which\_button(button\_press):  
 print (button\_press)

**Step 4:** Further, create and display the first button by calling the which\_button function you declared in step 3.

# Creating and displaying of button b1  
b1 = Button(app, text="Apple", command=lambda: which\_button("Apple"))  
b1.grid(padx=10, pady=10)

**Step 5:** Moreover, create and display the second button by calling the which\_button function you declared in step 3.

# Creating and displaying of button b2  
b2 = Button(app, text="Banana", command=lambda: which\_button("Banana"))  
b2.grid(padx=10, pady=10)

**Step 6:**Finally, create an infinite loop for displaying the app on the screen.

app.mainloop()

Example :-Button clicked.py

Online Reference :- https://www.geeksforgeeks.org/python/python-gui-tkinter/