iVEBSIM+: Visual Apps

Creating GUI Interfaces with Python & Tkinter

TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY

Funded by
the European Union

What are we doing today?

We are moving from "Text-Only" programming to modern **Graphical User Interfaces (GUI)**.

👉 Understand the visual "bricks" of a software.

👉 Learn the secret recipe to create windows.

👉 Build interactive buttons and text fields.

Funded by the European Union

Our Tool: Tkinter

Standard Library

No installation needed!

It is already built into Python.

Fast & Lean

It is small in size and launches instantly on any system.

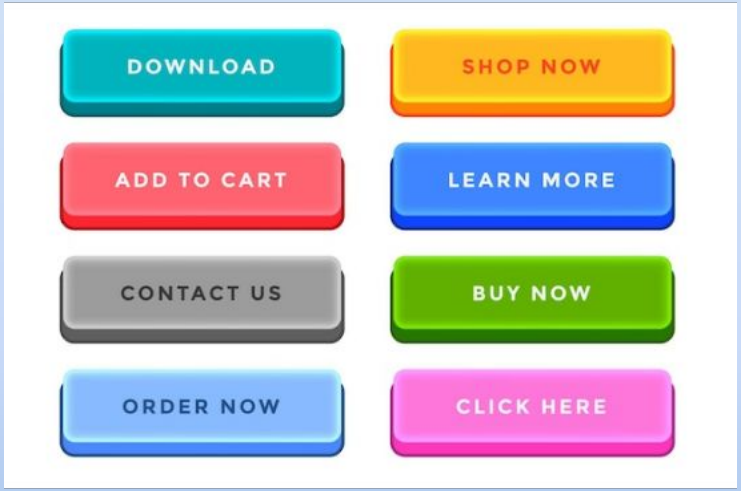
Universal

One code works on Windows, macOS, and Linux.

Funded by the European Union

Meet the "Widgets"

The components of your interface



A **Widget** is a component that performs a specific task.

Think of them as LEGO bricks for your app:

- Frame: The container.
- Label: The text.
- Button: The action.
- Entry: The input.

💡 *Note: We will use **Ttk** (Themed Tk), a modern version of these widgets that automatically matches the style of your operating system (Windows/Mac) for a cleaner look!*

Funded by the European Union

How to Create a Window

The standard 4-step workflow

Every time we create a visual program in Python, we follow this exact recipe:

1. **Import Tkinter** 👉 Tell Python to load the graphics toolkit.
2. **Create the Window Object** 👉Generate the blank window.
3. **Set Properties** 👉 Define the title, size, and look.
4. **Start the Main Loop (mainloop)** 👉 Keep the window open and running, waiting for user interactions.

Funded by the European Union

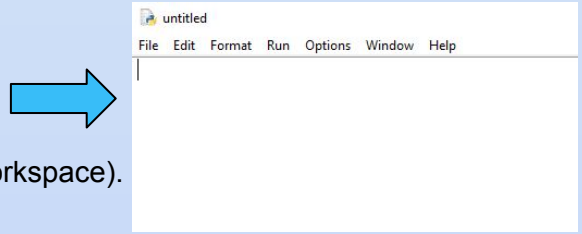
How to Write and Run your Code 🛠️

How to start: Your Python Editor

Step 1: Launch Python IDLE 🔍

- Press the **Windows Key** on your keyboard.
- Type **IDLE** in the search bar.
- Click on **IDLE (Python 3.x)** to open it.

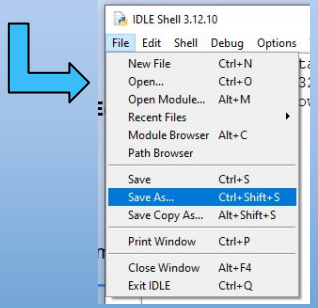
Step 2: Create a New File 📄



- In the top menu, click: **File → New File**.
- A new, blank white window will appear. This is your **Editor** (your workspace).

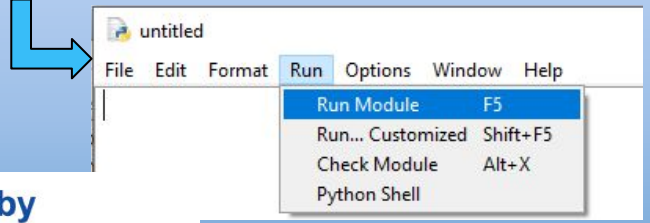
Step 3: Save your Work (Crucial!) 💾

- Go to **File → Save As...**.
- Choose a folder and give your file a name.
- **IMPORTANT:** You must add **.py** at the end of the name (es. exercise.py)



Step 4: Run your Program 🚀

- Once you have written your code, go to the menu: **Run → Run Module**.



iVEBSIM+
PESCARA 2026
TÜRKİYE ULUSAL AJANSI
TURKISH NATIONAL AGENCY
Funded by the European Union

Your First Code! 💻

Let's Build a Window!

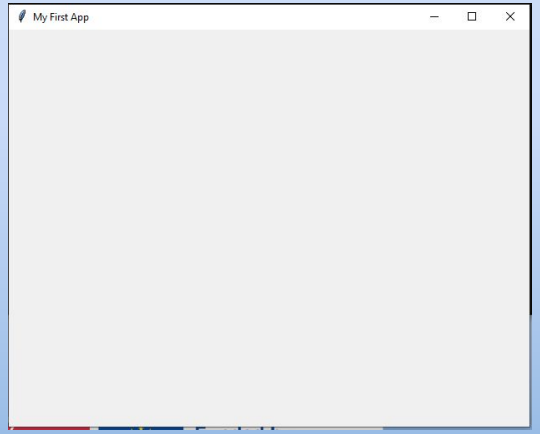
Copy this code into your Python editor:

```
import tkinter as tk

# Create the window
window = tk.Tk()

# Set window properties
window.title("My First App")
window.geometry("800x600")

# Start the application
window.mainloop()
```

🏃‍♀️ **Run the program:**

What happens?

You should see a blank window appear! 🎉



Funded by the European Union

Code Breakdown & Rules 🔍

Anatomy of the Code 🩺 Things to watch out for:

📦 `import tkinter as tk`

Loads the graphics toolkit. Writing `as tk` is just a shortcut to avoid typing the full word "tkinter" every time.

📐 `window.geometry("800x600")`

Sets Width x Height in pixels.

⚠️ *Warning:* Use the letter **"x"** inside the quotes, NOT the multiplication symbol (*).

🔄 `window.mainloop()`

This keeps the window alive and listening to your mouse/keyboard.

🛑 *Rule:* This must ALWAYS be the very last line of your script!

Funded by the European Union

Adding Content: Labels & Buttons

Let's meet our first interactive widgets

🏷️ The Label (Text Component) Used to display text on the screen:

```
>>>my_label = tk.Label(window, text="Hello Tkinter!", font=("Arial", 16))
```

Note: The first word inside the brackets (`window`) tells Python **where** to place the text.

🔘 The Button (Clickable Component) Creates a button that triggers an action:

```
>>>my_button = tk.Button(window, text="Click Me", command=say_hello)
```

`text`: What is written on the button.
`command`: The secret instructions (function) to run when clicked.

Funded by the European Union

The Golden Rule: Use `.pack()` 📌

Making widgets visible and organized

Creating a widget is not enough. To actually **see** it on the screen, you must "pack" it.

What does .pack() do?

It places the element into the window, stacking them from top to bottom.

Adding Space (Padding):

To prevent items from sticking together, we use space controls:

- pady=20 👉 Adds vertical spacing (top and bottom) in pixels.
- padx=10 👉 Adds horizontal spacing (left and right) in pixels.

```
# Places the text with some breathing room!
>>>my_label.pack(pady=20)
```

Funded by the European Union

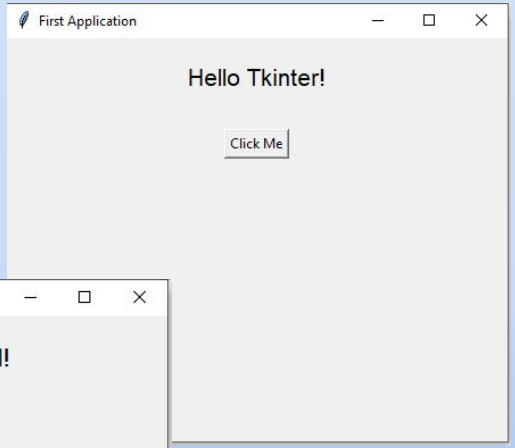
Lab Time: Make it Interactive!

Type this complete code and run it:

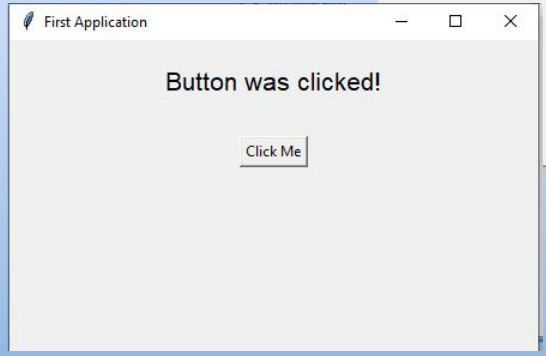
Copy this code into your Python editor:

```python
# 1. Setup Window
window = tk.Tk()
window.title("First Application")
window.geometry("800x600")

# 2. Create a Text Label
my_label = tk.Label(window, text="Hello Tkinter!", font=("Arial", 16))
my_label.pack(pady=20)

# 3. Define the Click Action (Function)
def say_hello():
    my_label.config(text="Button was clicked!") # Changes the text!

# 4. Create the Button
my_button = tk.Button(window, text="Click Me", command=say_hello)
my_button.pack(pady=10)

# 5. Run the Main Loop
window.mainloop()
```

Run the program:
What happens?



Click the button



🎯 **Goal:** Click the button and watch the text change from *"Hello Tkinter!"* to *"Button was clicked!"*!

Funded by the European Union

Getting Input: The Entry Widget

How to let users type text into your app

⌨️ The `tk.Entry` Component

Creates a single-line text box where users can type data.

```
entry_box = tk.Entry(window)
entry_box.pack()
```

🔮 The Magic Command: `.get()`

To read and use what the user typed, we use the `.get()` method inside our function:

```
# Grab the text from the box and save it in a variable
user_name = entry_box.get()
```

Funded by the European Union

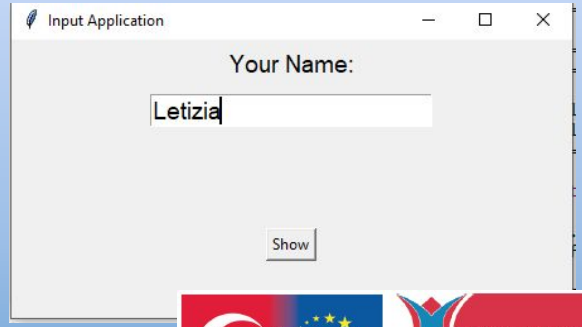
Lab Time: A Personalized App! 🧪

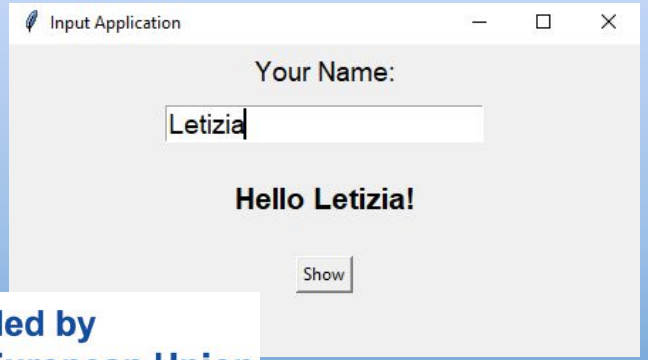
Type this code to create an interactive greeter:

Copy this code into your Python editor:

```python
import tkinter as tk

# 1. Setup Window
window = tk.Tk()
window.title("Input Application")
window.geometry("800x600")

# 2. Create Question Label
question_label = tk.Label(window, text="Your Name:", font=("Arial", 14))
question_label.pack(pady=5)

# 3. Create Entry Box
entry_box = tk.Entry(window, font=("Arial", 14))
entry_box.pack(pady=5)
```

```python
# 4. Create Empty Label for the Result
result_label = tk.Label(window, text="", font=("Arial", 16, "bold"))
result_label.pack(pady=20)

# 5. Define Action (Indentazione corretta: 4 spazi per riga)
def show_message():
    name = entry_box.get()
    result_label.config(text=f"Hello {name}!")

# 6. Create Button
my_button = tk.Button(window, text="Show", command=show_message)
my_button.pack(pady=5)

# 7. Run
window.mainloop()
```





Funded by the European Union

Advanced Inputs: Choosing Options
Radiobuttons and Checkbuttons

🔘 Radiobutton (Single Choice)

Used when the user must select **only one** option from a group. They share the same `variable`.

```
choice = tk.StringVar()
r1 = tk.Radiobutton(window, text="Option A", variable=choice, value="A")
r2 = tk.Radiobutton(window, text="Option B", variable=choice, value="B")
```

✅ Checkbutton (Multiple Choice)

Used for independent "tick boxes" where users can select multiple answers.

```
status = tk.IntVar()
check = tk.Checkbutton(window, text="I love Python", variable=status)
```

Funded by the European Union

Final Project: Login Screen 🚀

Putting it all together with style

We are going to build a realistic login system! Here are the new superpowers we will use:

🖼️ Adding Images (The Pillow Library)

We use `PIL` to load and resize images (like a school or company logo) inside our window.

🙈 Hiding Passwords

By adding `show="*"` to an Entry widget, the text turns into asterisks automatically!

💬 Popup Alerts (`messagebox`)

We can trigger standard system popups to tell the user if the login was successful or if they made a mistake.

Funded by the European Union

Lab Time: Build the Login Panel 🧪

Type this code (Part 1/3)

```
import tkinter as tk
from tkinter import messagebox
from PIL import Image, ImageTk    # Requires 'Pillow' library
```

👉**Imports & Libraries:** This block loads the required toolkits. We import **Tkinter** for the standard window, **messagebox** to create the popup alerts, and **Pillow (PIL)** to handle and resize our image files.

```
# 1. Login Logic Function
def login():
    username = entry_username.get()
    password = entry_password.get()

    if username == "admin" and password == "1234":
        messagebox.showinfo("Success", "Login successful!")
    else:
        messagebox.showerror("Error", "Invalid username or password!")
```

👉**The Login Logic:** This is the "brain" of the app. It reads what the user typed in the boxes. If the username is exactly `"admin"` AND the password is `"1234"`, it shows a green success popup. Otherwise, it shows a red error popup.

```
# 2. Window Setup
window = tk.Tk()
window.title("iVEBSIM+ Login Panel")
window.geometry("400x500")
window.resizable(False, False)
# Background color
window.configure(bg="white")
```

👉**Window Setup: t**his block builds the main window canvas. It sets the window title, defines the size (400 x 500 pixels), blocks the user from resizing it, and changes the background color to white.

Funded by the European Union

Lab Time: Build the Login Panel 🧪

Type this code (Part 2/3)

```
# 3. Load Logo Image (Make sure 'logo.png' is in the same folder)
image = Image.open("logo.png")
image = image.resize((200, 200))  # Logo size
logo = ImageTk.PhotoImage(image)

logo_label = tk.Label(window, image=logo, bg="white")
logo_label.pack(pady=20)
# 4. Title, Username and Password Fields
# ---------------- TITLE ----------------
title = tk.Label(window, text="iVEBSIM+ Login System",
                 font=("Arial", 16, "bold"),
                 bg="white",
                 fg="#1f4e5f")
title.pack(pady=10)

# ---------------- USERNAME ----------------
label_username = tk.Label(window, text="Username",
                          font=("Arial", 12),
                          bg="white")
label_username.pack(pady=5)
entry_username = tk.Entry(window, font=("Arial", 12), width=25)
entry_username.pack(pady=5)

# ---------------- PASSWORD ----------------
label_password = tk.Label(window, text="Password",
                          font=("Arial", 12),
                          bg="white")
label_password.pack(pady=5)
entry_password = tk.Entry(window, font=("Arial", 12), width=25, show="*")
entry_password.pack(pady=5)
```

👉**Loading the Logo Image:** This block displays the logo. It opens the image file, resizes it to a clean 200 x 200 size, puts it inside a standard Label component, and places it at the very top with some breathing room.

👉**The Interface Fields (Labels & Entries):** This block creates the text and the input fields. It adds a bold title, text prompts for Username and Password, and two entry boxes. The password box uses `show="*"` to hide the typed characters with asterisks.

Funded by the European Union

Lab Time: Build the Login Panel 🧪
Type this code (Part 3/3)

```
# 5. Login Button & Run
login_button = tk.Button(window,
                         text="Login",
                         font=("Arial", 12, "bold"),
                         bg="#0bb15d",
                         fg="white",
                         width=15,
                         command=login)
login_button.pack(pady=20)

window.mainloop()
```

👉**The Login Button & Main Loop:** This final block creates a green interactive button and starts the application. The button is connected to our `login` function. Finally, `window.mainloop()` turns the motor on and keeps the interface visible on the screen.

Funded by the European Union

Congratulations! Your App is Live! 🚀

You have successfully turned lines of code into a real, functional software interface. You are now a GUI Developer! 🎉

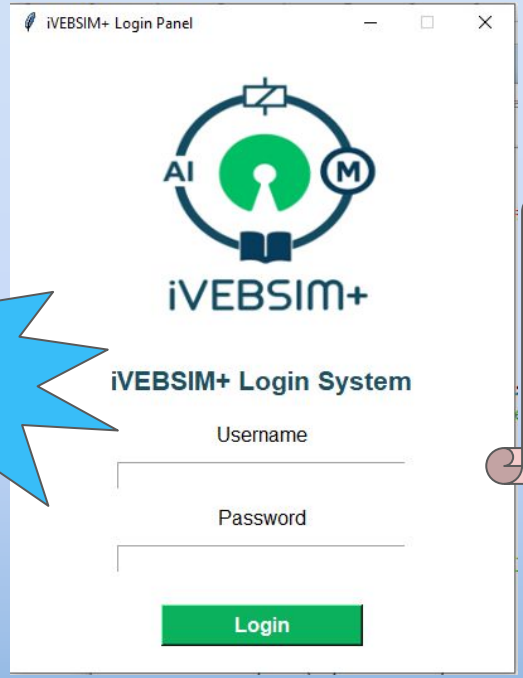
Logic_Panel.py.py - C:\Users\USER\Downloads\Logic_panel\Logic_Panel.py.py (3.12.10)

File Edit Format Run Options Window Help

Run Module F5
Run... Customized Shift+F5
Check Module Alt+X
Python Shell

```
import tkin
from tkinte
from PIL im                    # Requires 'Pillow' library

# 1. Login Logic Function
def login():
    username = entry_username.get()
    password = entry_password.get()

    if username == "admin" and password == "1234":
        messagebox.showinfo("Success", "Login successful!")
    else:
        messagebox.showerror("Error", "Invalid username or password!")

# 2. Window Setup
window = tk.Tk()
window.title("iVEBSIM+ Login Panel")
window.geometry("400x500")
window.resizable(False, False)
# Background color
window.configure(bg="white")

# 3. Load Logo Image (Make sure 'logo.png' is in the same folder)
image = Image.open("logo.png")
image = image.resize((200, 200))  # Logo size
logo = ImageTk.PhotoImage(image)

logo_label = tk.Label(window, image=logo, bg="white")
logo_label.pack(pady=20)
```

One click to bring your code to life...



...and here is your professional Result! ✨

Funded by the European Union