R.V. COLLEGE OF ENGINEERING, Bengaluru-560059
(Autonomous Institution Affiliated to VTU, Belgaum)

DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION

# PYTHON EXPERIENTIAL LEARNING(LAB)

ARNAV SINGHAL USN-1RV23EI009

KRISHNA PURWAR USN-1RV23EI027

PYTHON F-SECTION

# INTRODUCTION TO GUI

A Graphical User Interface (GUI) allows users to interact with electronic devices and software through visual elements like icons, buttons, windows, and menus. This differs from text-based or command-line interfaces, where interactions rely solely on text commands.

The main purpose of a GUI is to improve the user experience by making it more intuitive and visually engaging. Users can perform tasks by interacting with graphical components using devices such as a mouse, touchpad, or touchscreen.

GUIs are prevalent in operating systems, applications, and various software to provide a user-friendly environment.

**Why is GUI Preferred?**

GUI is favored for its user-friendly design, offering intuitive interactions through visual elements like icons, buttons, and windows. It enhances accessibility, lowers the learning curve, supports multitasking, and provides visual feedback. Its widespread adoption and consistent design across platforms ensure a seamless and enjoyable user experience.

# INTRODUCTION TO TKinter

Tkinter is a built-in Python library for creating Graphical User Interfaces (GUIs). It offers a set of tools and widgets (pre-built GUI components) that developers can use to create windows, dialogs, buttons, menus, and other GUI elements. Tkinter is based on the Tk GUI toolkit, which originated as part of the Tcl (Tool Command Language) scripting language.

**Key Concepts and Components of Tkinter**

1. **Widgets**: Tkinter provides a variety of widgets that you can use to build the user interface of your application. Common widgets include:
   - Label: For displaying text.
   - Button: For creating buttons.
   - Entry: For text input.
   - Frame: For organizing other widgets.

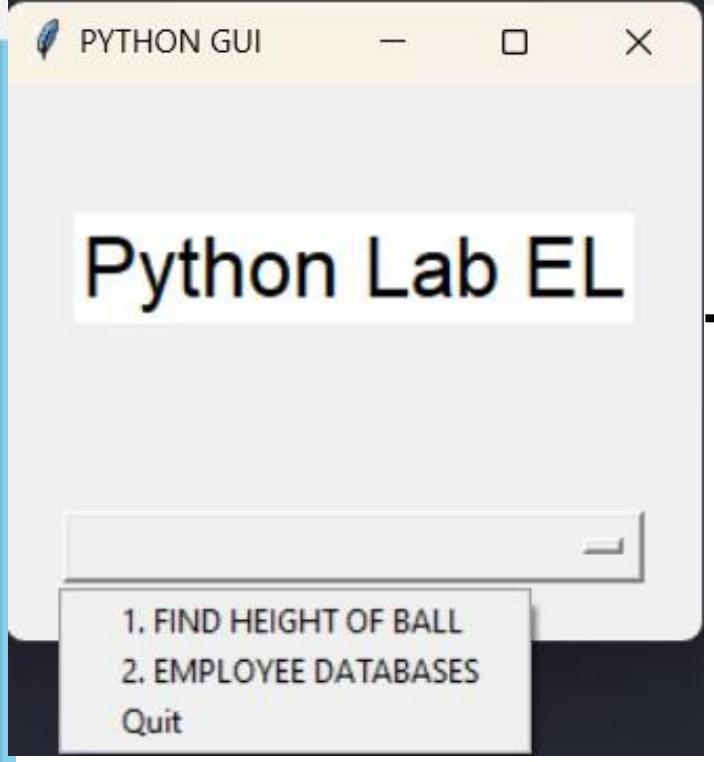
2. **Windows**: The main GUI window in Tkinter is created using the `Tk` class. Additional windows can be created using the `Toplevel` class. These windows serve as containers for organizing and displaying various widgets.

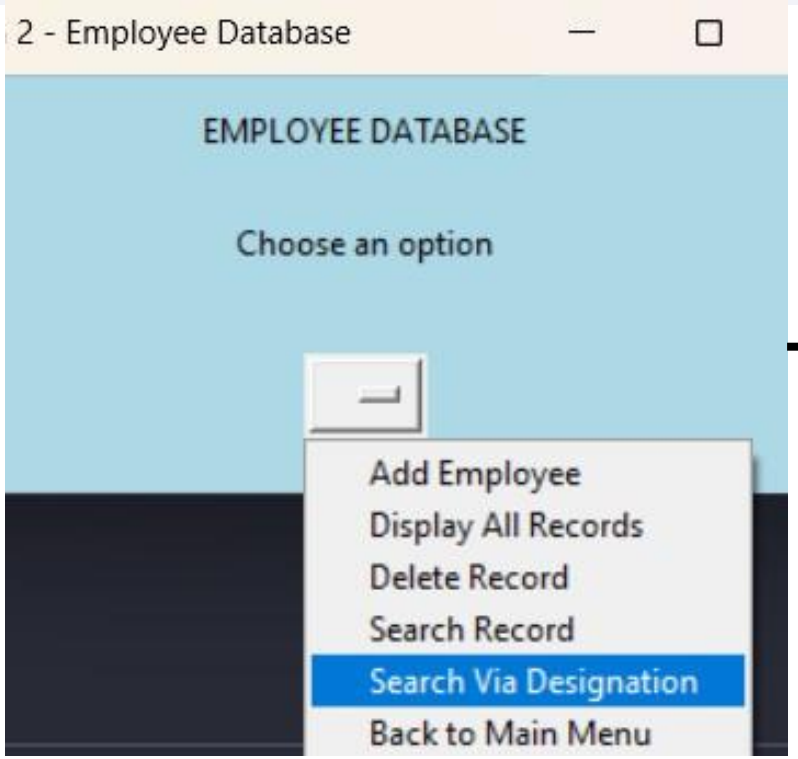
3. **Geometry Management**: Tkinter includes three geometry managers (pack, grid, and place) that help you organize and arrange widgets within a container (e.g., a window or a frame). These managers allow you to control the placement and sizing of widgets.

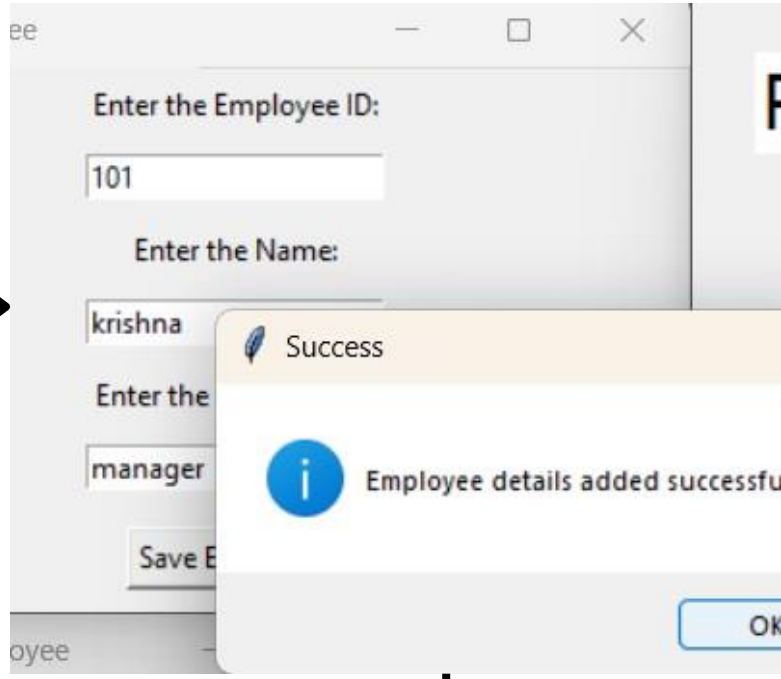
# EMPLOYEE DATABASE

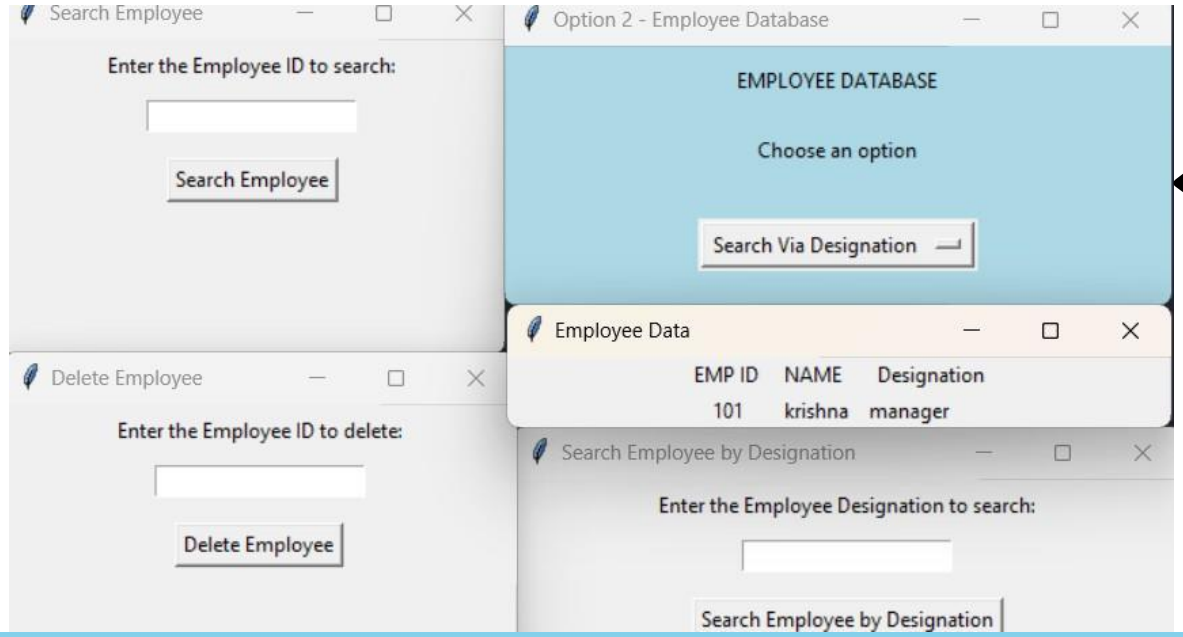
```
import tkinter as tk
from tkinter import messagebox, Label, PhotoImage, StringVar,
OptionMenu, Toplevel

emp = {}

def add_employee():
new_window = Toplevel(root)
new_window.title("Add Employee")

bel(new_window, text="Enter the Employee ID:").pack(padx=20, pady=5)
entry_id = tk.Entry(new_window)
entry_id.pack(padx=20, pady=5)

Label(new_window, text="Enter the Name:").pack(padx=20, pady=5)
entry_name = tk.Entry(new_window)
entry_name.pack(padx=20, pady=5)

bel(new_window, text="Enter the Designation:").pack(padx=20, pady=5)
entry_designation = tk.Entry(new_window)
entry_designation.pack(padx=20, pady=5)

def save_employee():
try:
emp_id = int(entry_id.get())
emp_name = entry_name.get()
emp_designation = entry_designation.get()

if not emp_name or not emp_designation:
raise ValueError("Empty fields")

emp[emp_id] = [emp_name, emp_designation]
messagebox.showinfo("Success", "Employee details added successfully.")
new_window.destroy()
except ValueError:
messagebox.showerror("Error", "Please enter valid and complete
Employee ID and details.")

tk.Button(new_window, text="Save Employee",
command=save_employee).pack(pady=10)

def display_all_records():
new_window = Toplevel(root)
new_window.title('Employee Data')

header = "{:<10} {:<10} {:<10}".format('EMP ID', 'NAME', 'Designation')
tk.Label(new_window, text=header).pack()
for key, value in emp.items():
name, designation = value
record = "{:<10} {:<10} {:<10}".format(key, name, designation)
tk.Label(new_window, text=record).pack()

def delete_record():
new_window = Toplevel(root)
new_window.title("Delete Employee")

Label(new_window, text="Enter the Employee ID to
delete:").pack(padx=20, pady=5)
entry_id = tk.Entry(new_window)
entry_id.pack(padx=20, pady=5)

def delete_employee():
try:
emp_id = int(entry_id.get())
if emp_id in emp:
emp.pop(emp_id)
messagebox.showinfo("Success", "Employee details deleted
successfully.")
new_window.destroy()
else:
def search_employee_by_designation():
designation = entry_designation.get().lower()
matching_employees = [(emp_id, details) for emp_id, details in
emp.items() if designation in details[1].lower()]

if matching_employees:
display_matching_records(matching_employees)
else:
messagebox.showinfo("Error", "No employees with the given
designation found.")

tk.Button(new_window, text="Search Employee by
Designation",
command=search_employee_by_designation).pack(pady=10)

def display_matching_records(matching_employees):
new_window = Toplevel(root)
new_window.title('Matching Employee Data')

header = "{:<10} {:<10} {:<10}".format('EMP ID', 'NAME',
'Designation')
tk.Label(new_window, text=header).pack()

for emp_id, details in matching_employees:
record = "{:<10} {:<10} {:<10}".format(emp_id, details[0],
details[1])
tk.Label(new_window, text=record).pack()
messagebox.showinfo("Error", "Employee not found.")
except ValueError:
messagebox.showerror("Error", "Please enter a valid Employee
ID.")

tk.Button(new_window, text="Delete Employee",
command=delete_employee).pack(pady=10)
```

```
def search_record():
new_window = Toplevel(root)
new_window.title("Search Employee")

Label(new_window, text="Enter the Employee ID to search:").pack(padx=20, pady=5)
entry_id = tk.Entry(new_window)
entry_id.pack(padx=20, pady=5)

def search_employee():
try:
emp_id = int(entry_id.get())
emp_details = emp.get(emp_id)
if emp_details:
messagebox.showinfo("Employee Details", f"Name: {emp_details[0]}\nDesignation: {emp_details[1]}")
else:
messagebox.showinfo("Error", "Employee not found.")
except ValueError:
messagebox.showerror("Error", "Please enter a valid Employee ID.")

tk.Button(new_window, text="Search Employee", command=search_employee).pack(pady=10)

def search_record_by_designation():
new_window = Toplevel(root)
new_window.title("Search Employee by Designation")

Label(new_window, text="Enter the Employee Designation to search:").pack(padx=20, pady=5)
entry_designation = tk.Entry(new_window)
entry_designation.pack(padx=20, pady=5)
```


PYTHON GUI
Python Lab EL
1. FIND HEIGHT OF BALL
2. EMPLOYEE DATABASES
Quit


2 - Employee Database
EMPLOYEE DATABASE
Choose an option
Add Employee
Display All Records
Delete Record
Search Record
Search Via Designation
Back to Main Menu

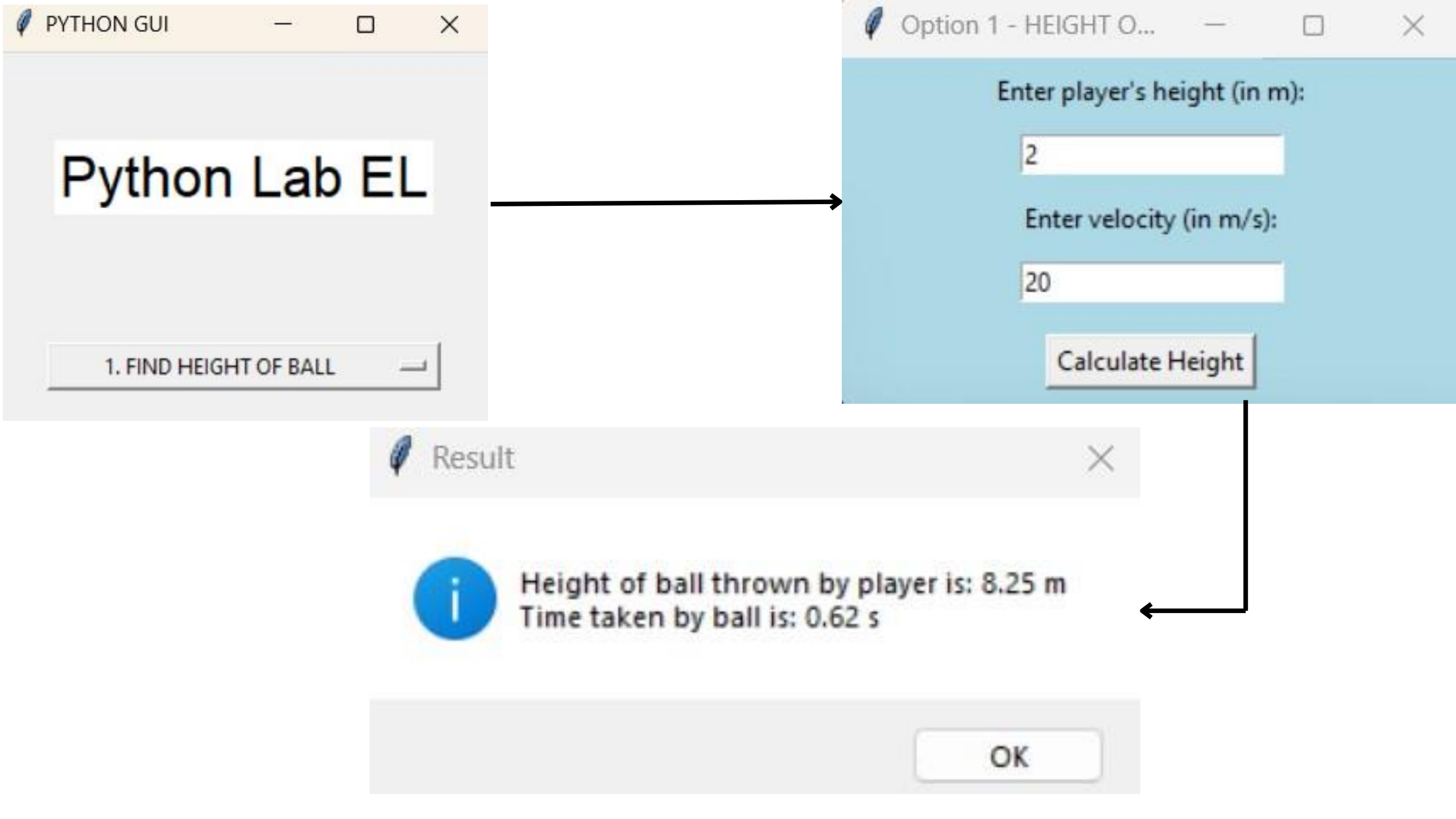

Enter the Employee ID:
101
Enter the Name:
krishna
Enter the
manager
Save E
Success
Employee details added successfu
OK


Search Employee
Enter the Employee ID to search:
Search Employee
Option 2 - Employee Database
EMPLOYEE DATABASE
Choose an option
Search Via Designation
Employee Data
EMP ID NAME Designation
101 krishna manager
Delete Employee
Enter the Employee ID to delete:
Delete Employee
Search Employee by Designation
Enter the Employee Designation to search:
Search Employee by Designation

# HEIGHT OF BALL

```
def option2():
new_window = Toplevel(root)
new_window.title("Option 2 - Employee Database")
new_window.configure(bg='light blue')

Label(new_window, text="EMPLOYEE DATABASE", bg='light
blue').pack(pady=10)
Label(new_window, text="Choose an option", bg='light
blue').pack(pady=10)

selected_employee_option = StringVar()
options = ["Add Employee", "Display All Records", "Delete Record",
"Search Record", "Search Via Designation", "Back to Main Menu"]
dropdown = OptionMenu(new_window, selected_employee_option,
*options)
dropdown.pack(padx=20, pady=20)

def on_employee_option_selected(*args):
selected_option = selected_employee_option.get()
if selected_option == "Add Employee":
add_employee()
elif selected_option == "Display All Records":
display_all_records()
elif selected_option == "Delete Record":
delete_record()
elif selected_option == "Search Record":
search_record()
elif selected_option == "Search Via Designation":
search_record_by_designation()
elif selected_option == "Back to Main Menu":
new_window.destroy()

selected_employee_option.trace_add("write",
on_employee_option_selected)
```

```
root = tk.Tk()
root.title("PYTHON GUI")

Label(root, text="Python Lab EL", bg='white', fg='black', font=('Arial',
24)).pack(pady=50)

selected_option = StringVar()
options = ["1. FIND HEIGHT OF BALL", "2. EMPLOYEE DATABASES",
"Quit"]
dropdown = OptionMenu(root, selected_option, *options)
dropdown.pack(padx=20, pady=20)
dropdown.config(width=30)

def on_option_selected(*args):
option = selected_option.get()
if option.startswith("1"):
option1()
elif option.startswith("2"):
option2()
elif option == "Quit":
root.destroy()

selected_option.trace_add("write", on_option_selected)

root.mainloop()
```


PYTHON GUI
Python Lab EL
1. FIND HEIGHT OF BALL
Option 1 - HEIGHT O...
Enter player's height (in m):
2
Enter velocity (in m/s):
20
Calculate Height
Result
Height of ball thrown by player is: 8.25 m
Time taken by ball is: 0.62 s
OK

# THANK YOU!