PD LAB
ASSIGNMENT - 2

Name: Raunak Thanawala
Registration Number: 231070051
Branch: Computer Engineering
Batch: 3

Aim:-

To create a contact application in python using tkinter and dictionaries.

Theory:-

Dictionary:

A dictionary is a collection which is ordered, changeable and does not allow duplicates. They are used to store data values in key:value pairs. To initialize a dictionary we use curly braces ({}).
Eg. dict = { “name”: “Raunak”, “age”: “19”, “Batch”: “Computer”}

Where dict[name] = Raunak, dict[age] = 19 and dict[Batch] = Computer

Tkinter:

Tkinter is Python's basic GUI library used for making cross-platform desktop applications.
We can use various widgets like buttons, labels, text fields, and more to build interactive user interfaces.
It offers three geometry managers (pack, grid, and place) for widget layout.

For Windows and MacOS:
tkinter is pre downloaded in the python installation. We can check if it exists with the command:

python -m tkinter

For Linux:
On debian based Linux OS we have to use the command:

sudo apt-get install python3-tk

Code and Output:

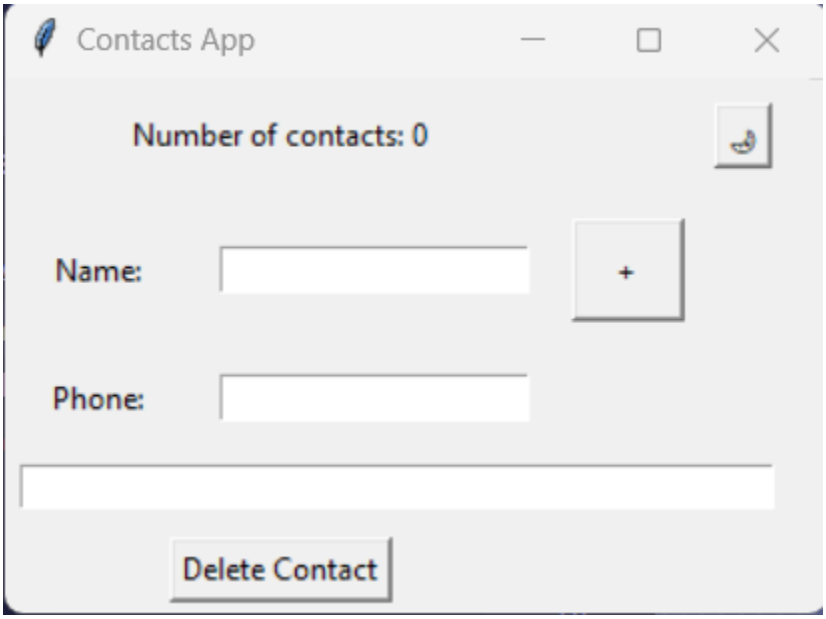
```
import tkinter as tk
from tkinter import messagebox
```

```
from tkinter import font as tkfont

x = 0
class ContactsApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Contacts App")
        self.contacts = {}
        self.is_dark_mode = False
        # Create and place widgets
        self.create_widgets()

    def create_widgets(self):
        # Name Label and Entry
        tk.Label(self.root, text="Name:").grid(row=1, column=0, padx=10, pady=10)
        self.name_entry = tk.Entry(self.root)
        self.name_entry.grid(row=1, column=1, padx=5, pady=5)

        # Phone Label and Entry
        tk.Label(self.root, text="Phone:").grid(row=2, column=0, padx=10,
pady=10)
        self.phone_entry = tk.Entry(self.root)
        self.phone_entry.grid(row=2, column=1, padx=5, pady=5)

        # Add Contact Button
        tk.Button(self.root, text="+",
command=self.add_contact,width=5,height=2).grid(row=1,column=2 , pady=10)

        # Contacts Listbox
        self.contacts_listbox = tk.Listbox(self.root, width=50, height=1)
        self.contacts_listbox.grid(row=4, column=0, columnspan=4, padx=5, pady=5)

        # Delete Contact Button
        tk.Button(self.root, text="Delete Contact",
command=self.delete_contact).grid(row=5, columnspan=2, pady=5)

        self.contact_count_label = tk.Label(self.root, text="Number of contacts:
0")
        self.contact_count_label.grid(row=0, columnspan=2, pady=10)
```

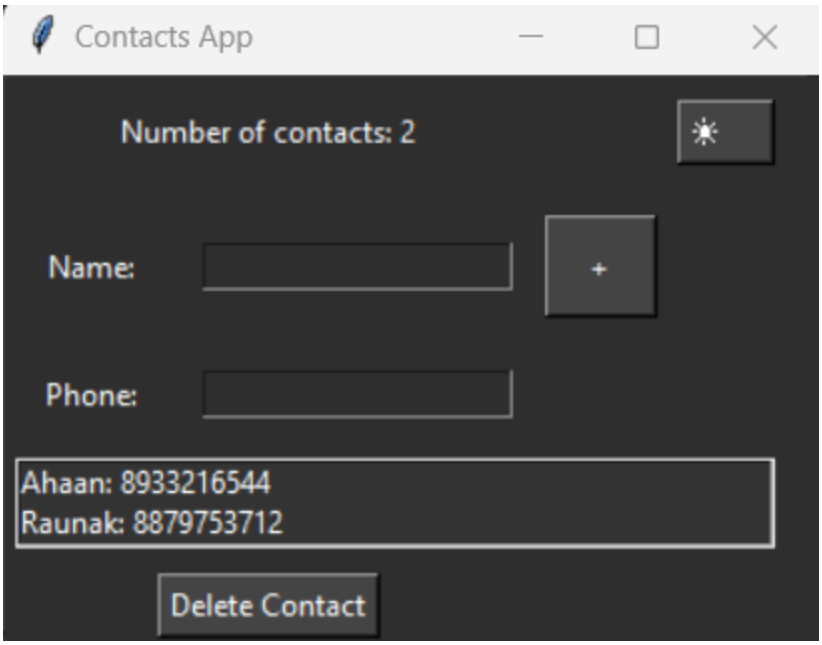
```
        self.toggle_button = tk.Button(self.root, text="🌙",
command=self.toggle_mode)
        self.toggle_button.grid(row=0, column=3, pady=10)

    def apply_theme(self):
        if self.is_dark_mode:
            bg_color = '#2e2e2e'
            fg_color = '#ffffff'
            button_bg = '#444444'
            button_fg = '#ffffff'
            listbox_bg = '#333333'
            listbox_fg = '#ffffff'
        else:
            bg_color = '#f0f0f0'
            fg_color = '#000000'
            button_bg = '#e0e0e0'
            button_fg = '#000000'
            listbox_bg = '#ffffff'
            listbox_fg = '#000000'

        self.root.configure(bg=bg_color)
        self.toggle_button.config(bg=button_bg, fg=button_fg)
        self.contacts_listbox.config(bg=listbox_bg, fg=listbox_fg)

        # Update other widgets
        for widget in self.root.winfo_children():
            if isinstance(widget, tk.Label):
                widget.config(bg=bg_color, fg=fg_color)
            elif isinstance(widget, tk.Entry):
                widget.config(bg=bg_color, fg=fg_color)
            elif isinstance(widget, tk.Button):
                widget.config(bg=button_bg, fg=button_fg)

    def toggle_mode(self):
        self.is_dark_mode = not self.is_dark_mode
        mode_text = "☼" if self.is_dark_mode else "🌙"
        self.toggle_button.config(text=mode_text)
```

```
        self.apply_theme()
    def add_contact(self):
        name = self.name_entry.get().strip()
        phone = self.phone_entry.get().strip()

        if name and phone:

            self.contacts[name] = phone
            self.update_contacts_listbox()
            self.name_entry.delete(0, tk.END)
            self.phone_entry.delete(0, tk.END)
        else:
            messagebox.showwarning("Input Error", "Both name and phone number are
required.")
        self.update_contact_count()

    def update_contacts_listbox(self):
        self.contacts_listbox.delete(0, tk.END)
        for name, phone in sorted(self.contacts.items()):
            self.contacts_listbox.insert(tk.END, f"{name}: {phone}")

        # Adjust the height of the Listbox based on the number of contacts
        num_items = len(self.contacts)
        max_visible_items = 10
        new_height = min(num_items, max_visible_items)
        self.contacts_listbox.config(height=new_height)

    def update_contact_count(self):
        num_contacts = len(self.contacts)
        self.contact_count_label.config(text=f"Number of contacts:
{num_contacts}")

    def delete_contact(self):
        selected_index = self.contacts_listbox.curselection()
        if selected_index:
            selected_contact = self.contacts_listbox.get(selected_index[0])
            name = selected_contact.split(':')[0].strip()
            del self.contacts[name]
            self.update_contacts_listbox()
```

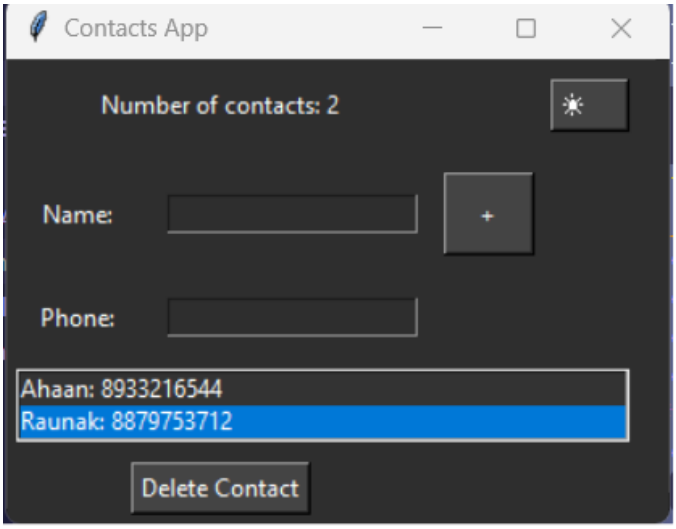
```
        else:
            messagebox.showwarning("Selection Error", "Please select a contact to
delete.")
        self.update_contact_count()

if __name__ == "__main__":
    root = tk.Tk()
    app = ContactsApp(root)
    root.mainloop()
```



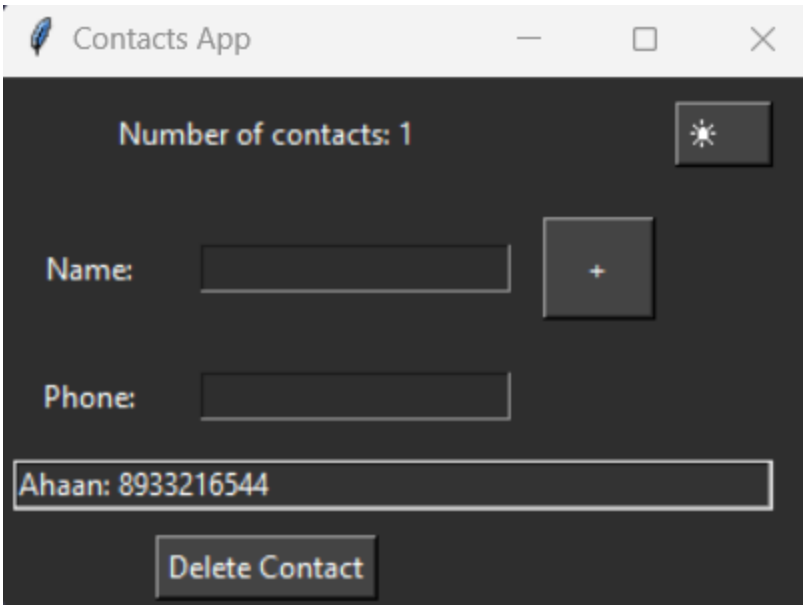
Basic Output



Output with 2 Contacts saved and Dark mode enabled



Selecting a contact so we can delete it



Contacts after Deleting the Contact

Conclusion:

Thus we have written a program to store contacts in a dictionary and then have made a Contacts application using tkinter.