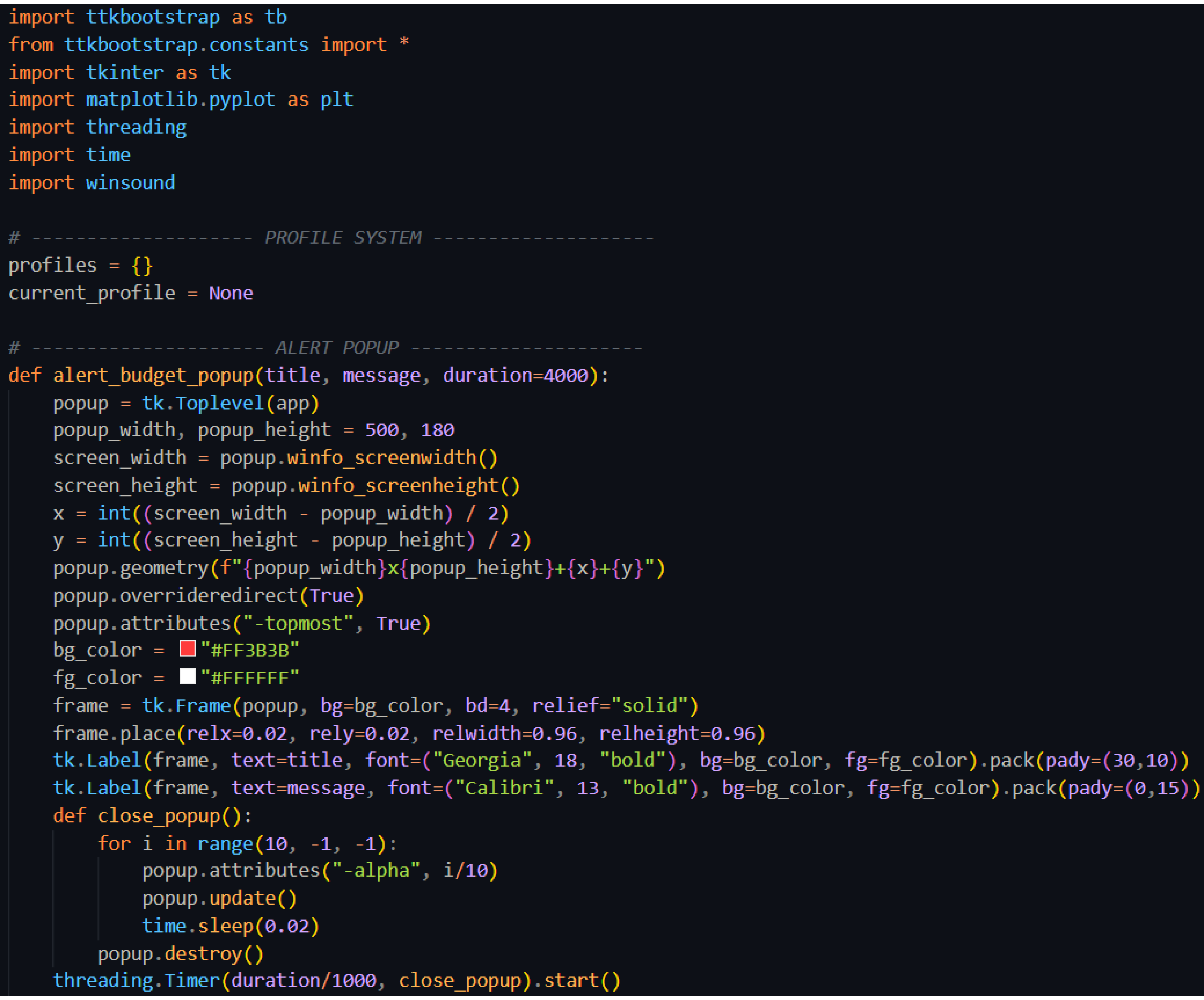
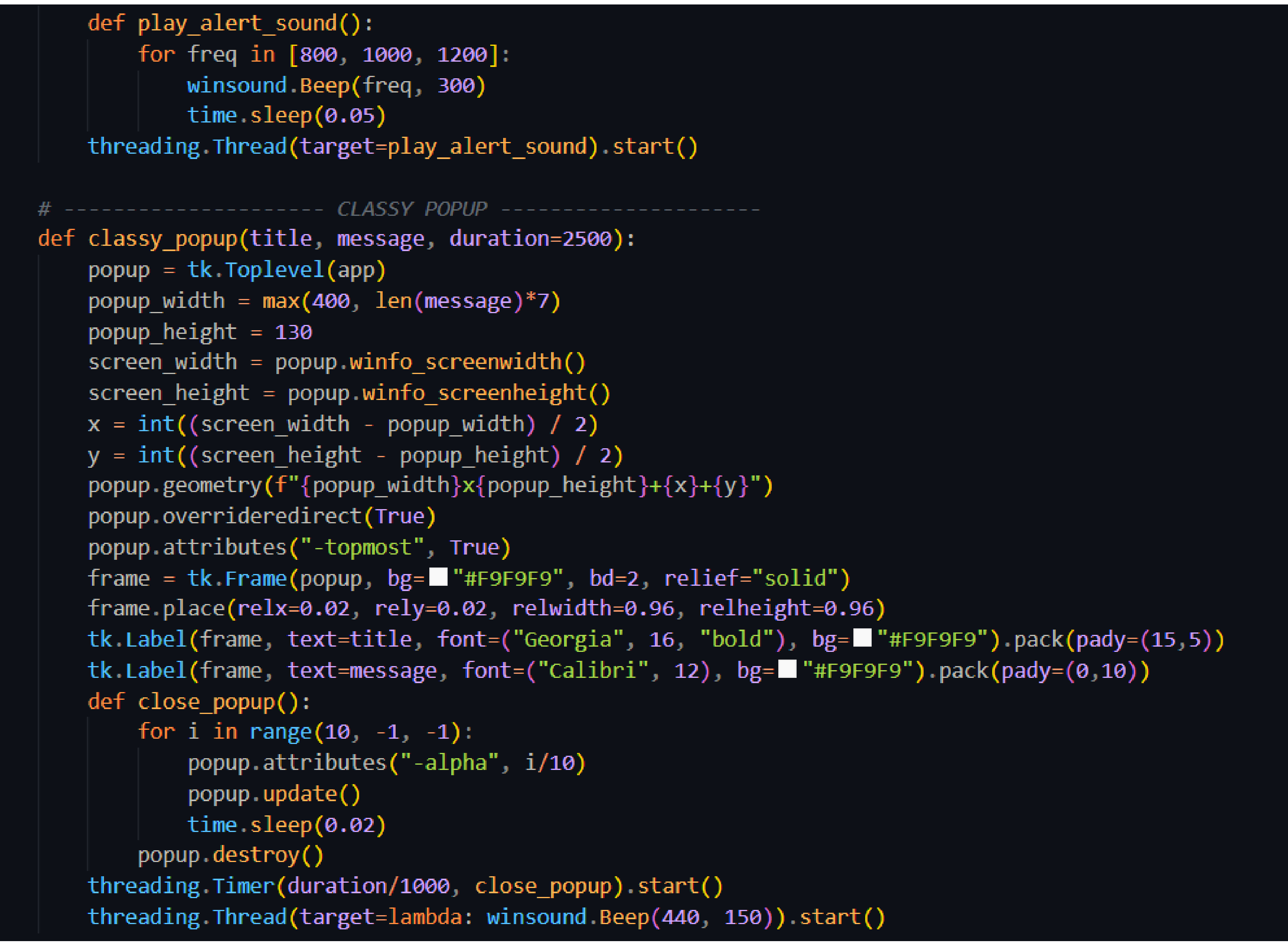
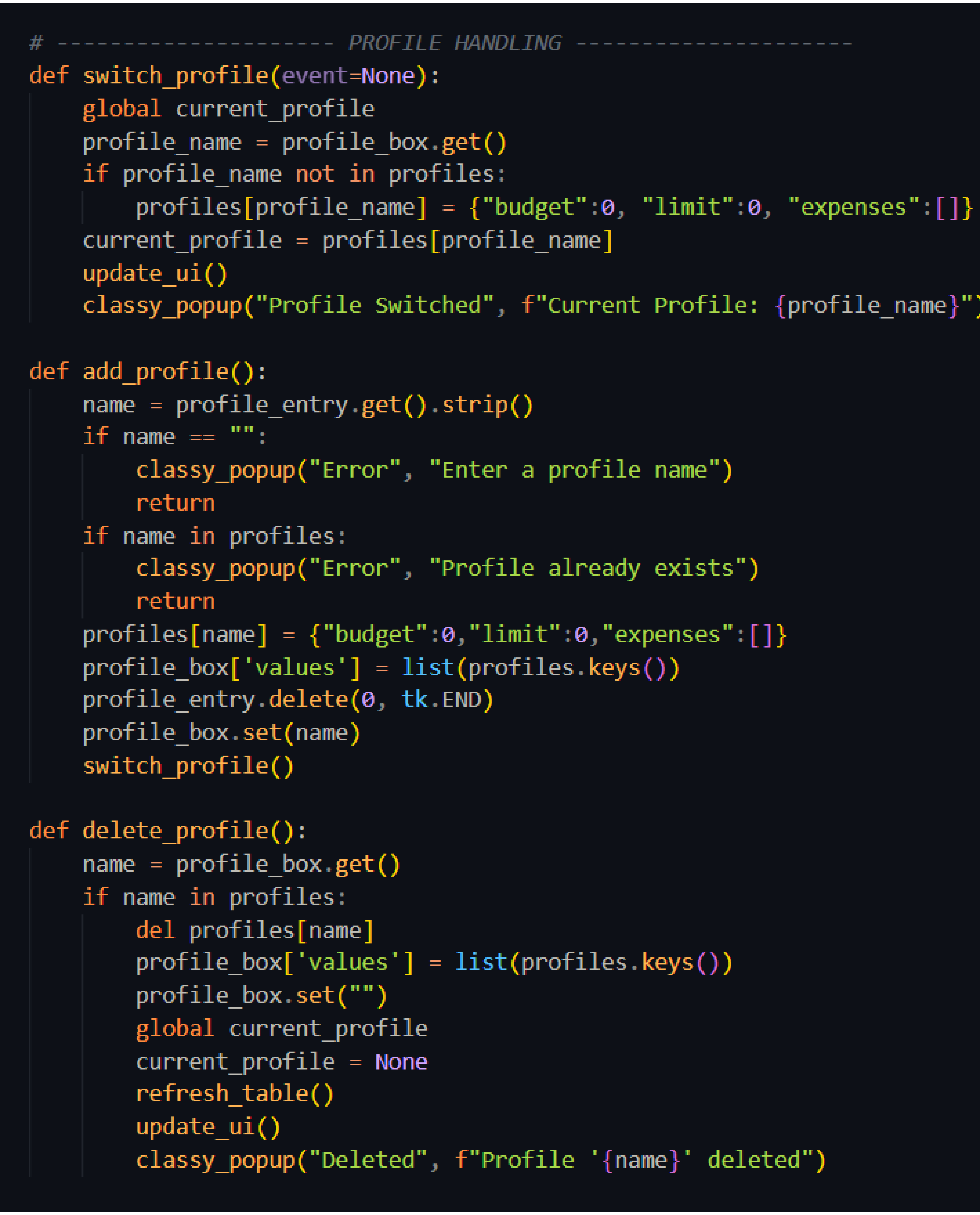


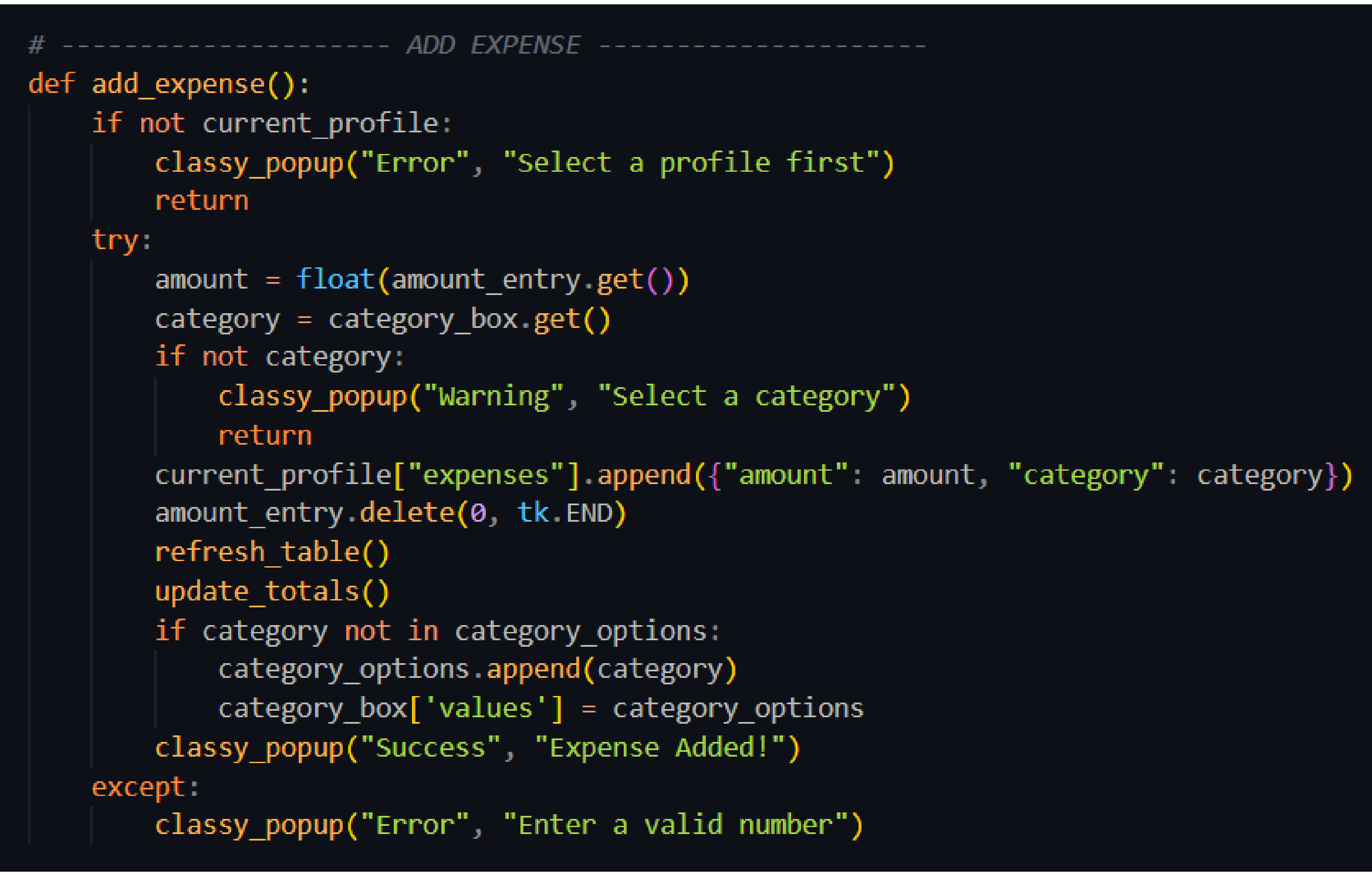
```python
import ttkbootstrap as tb
from ttkbootstrap.constants import *
import tkinter as tk
import matplotlib.pyplot as plt
import threading
import time
import winsound

# -------------------- PROFILE SYSTEM --------------------
profiles = {}
current_profile = None

# --------------------- ALERT POPUP --------------------
def alert_budget_popup(title, message, duration=4000):
    popup = tk.Toplevel(app)
    popup_width, popup_height = 500, 180
    screen_width = popup.winfo_screenwidth()
    screen_height = popup.winfo_screenheight()
    x = int((screen_width - popup_width) / 2)
    y = int((screen_height - popup_height) / 2)
    popup.geometry(f"{popup_width}x{popup_height}+{x}+{y}")
    popup.overrideredirect(True)
    popup.attributes("-topmost", True)
    bg_color = "#FF3B3B"
    fg_color = "#FFFFFF"
    frame = tk.Frame(popup, bg=bg_color, bd=4, relief="solid")
    frame.place(relx=0.02, rely=0.02, relwidth=0.96, relheight=0.96)
    tk.Label(frame, text=title, font=("Georgia", 18, "bold"), bg=bg_color, fg=fg_color).pack(pady=(30,10))
    tk.Label(frame, text=message, font=("Calibri", 13, "bold"), bg=bg_color, fg=fg_color).pack(pady=(0,15))
    def close_popup():
        for i in range(10, -1, -1):
            popup.attributes("-alpha", i/10)
            popup.update()
            time.sleep(0.02)
        popup.destroy()
    threading.Timer(duration/1000, close_popup).start()
```



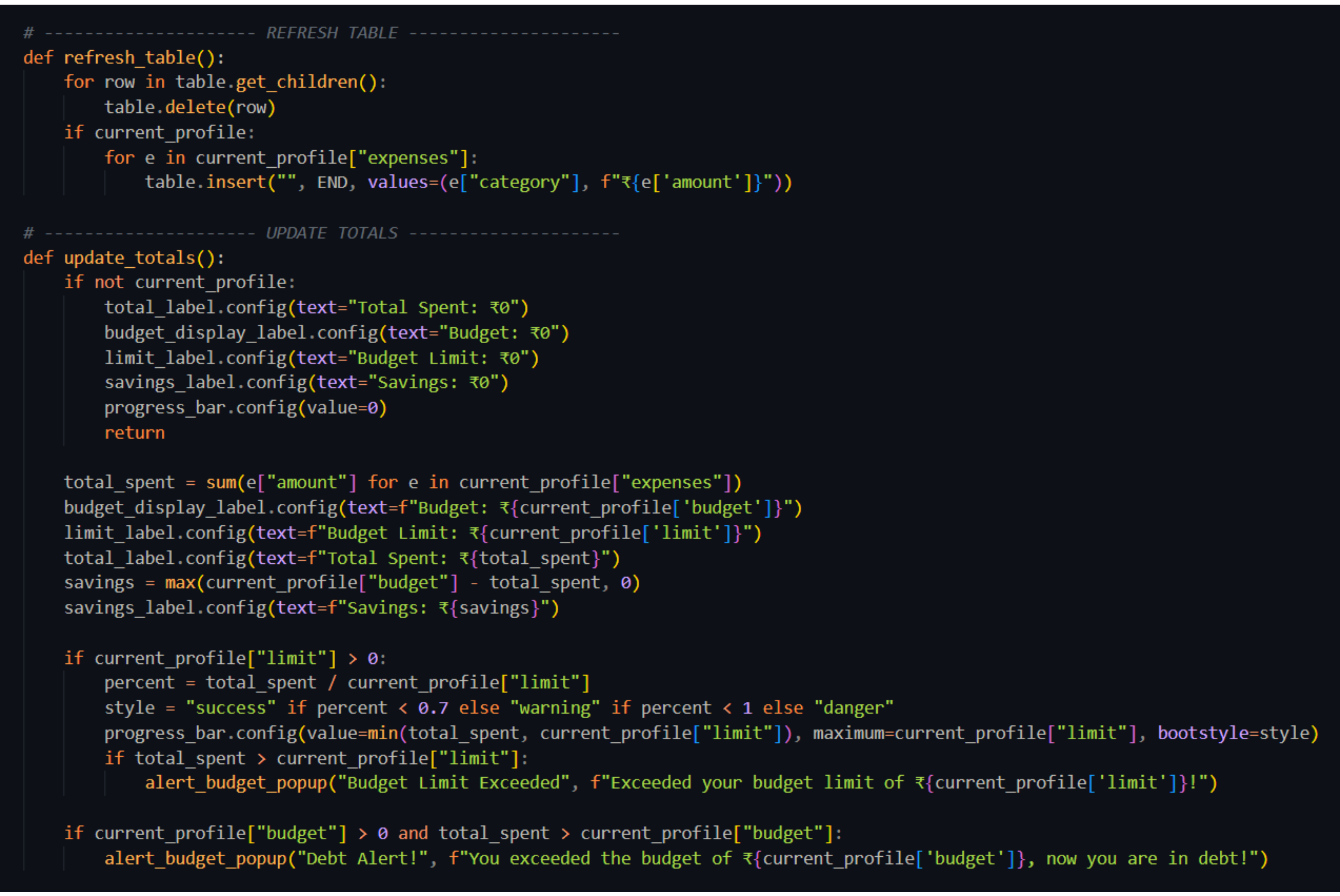
```python
    def play_alert_sound():
        for freq in [800, 1000, 1200]:
            winsound.Beep(freq, 300)
            time.sleep(0.05)
    threading.Thread(target=play_alert_sound).start()

# --------------------- CLASSY POPUP ---------------------
def classy_popup(title, message, duration=2500):
    popup = tk.Toplevel(app)
    popup_width = max(400, len(message)*7)
    popup_height = 130
    screen_width = popup.winfo_screenwidth()
    screen_height = popup.winfo_screenheight()
    x = int((screen_width - popup_width) / 2)
    y = int((screen_height - popup_height) / 2)
    popup.geometry(f"{popup_width}x{popup_height}+{x}+{y}")
    popup.overrideredirect(True)
    popup.attributes("-topmost", True)
    frame = tk.Frame(popup, bg="#F9F9F9", bd=2, relief="solid")
    frame.place(relx=0.02, rely=0.02, relwidth=0.96, relheight=0.96)
    tk.Label(frame, text=title, font=("Georgia", 16, "bold"), bg="#F9F9F9").pack(pady=(15,5))
    tk.Label(frame, text=message, font=("Calibri", 12), bg="#F9F9F9").pack(pady=(0,10))
    def close_popup():
        for i in range(10, -1, -1):
            popup.attributes("-alpha", i/10)
            popup.update()
            time.sleep(0.02)
        popup.destroy()
    threading.Timer(duration/1000, close_popup).start()
    threading.Thread(target=lambda: winsound.Beep(440, 150)).start()
```

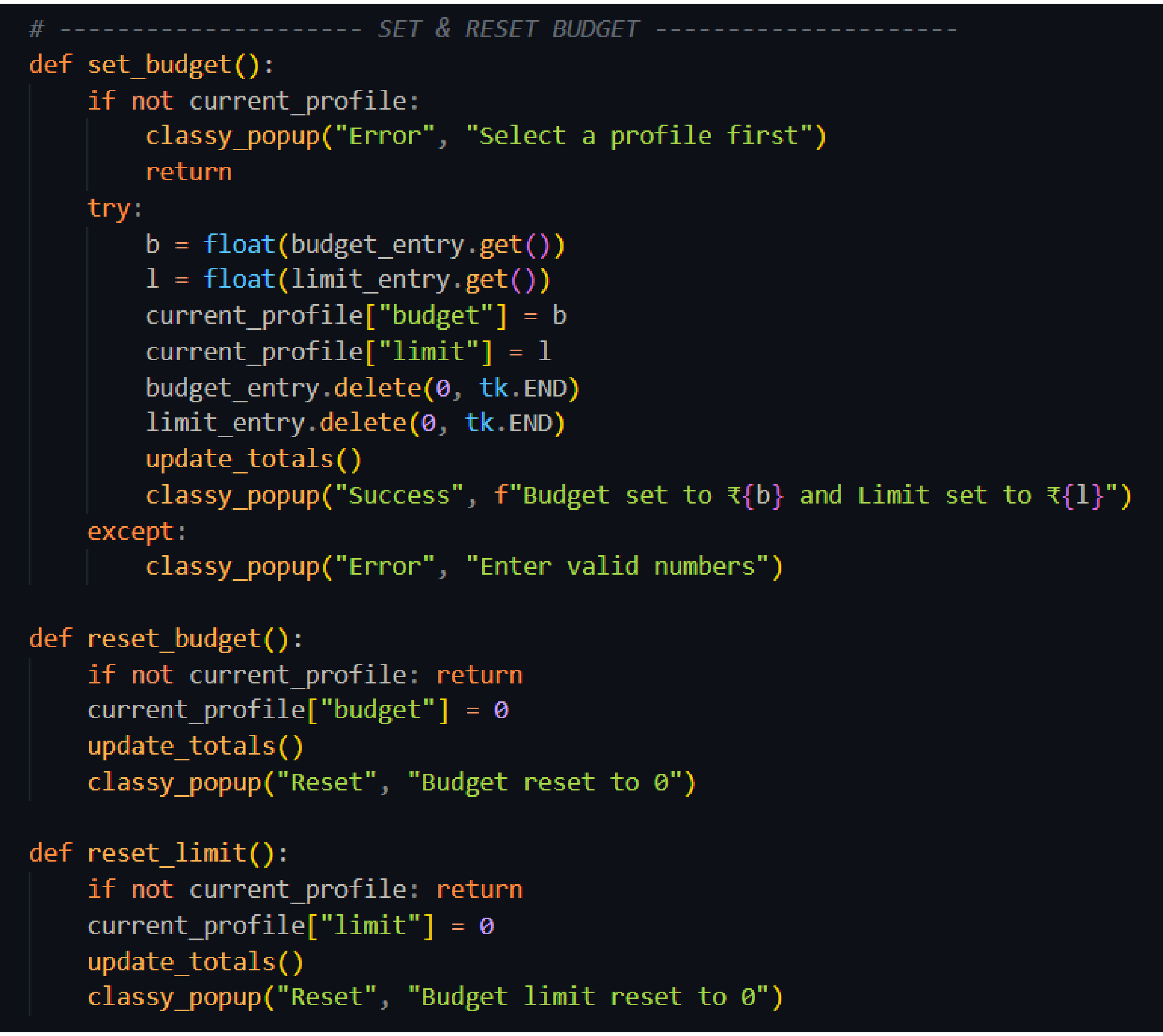


```python
# --------------------- PROFILE HANDLING ---------------------
def switch_profile(event=None):
    global current_profile
    profile_name = profile_box.get()
    if profile_name not in profiles:
        profiles[profile_name] = {"budget":0, "limit":0, "expenses":[]}
    current_profile = profiles[profile_name]
    update_ui()
    classy_popup("Profile Switched", f"Current Profile: {profile_name}"

def add_profile():
    name = profile_entry.get().strip()
    if name == "":
        classy_popup("Error", "Enter a profile name")
        return
    if name in profiles:
        classy_popup("Error", "Profile already exists")
        return
    profiles[name] = {"budget":0,"limit":0,"expenses":[]}
    profile_box['values'] = list(profiles.keys())
    profile_entry.delete(0, tk.END)
    profile_box.set(name)
    switch_profile()

def delete_profile():
    name = profile_box.get()
    if name in profiles:
        del profiles[name]
        profile_box['values'] = list(profiles.keys())
        profile_box.set("")
        global current_profile
        current_profile = None
        refresh_table()
        update_ui()
        classy_popup("Deleted", f"Profile '{name}' deleted")
```

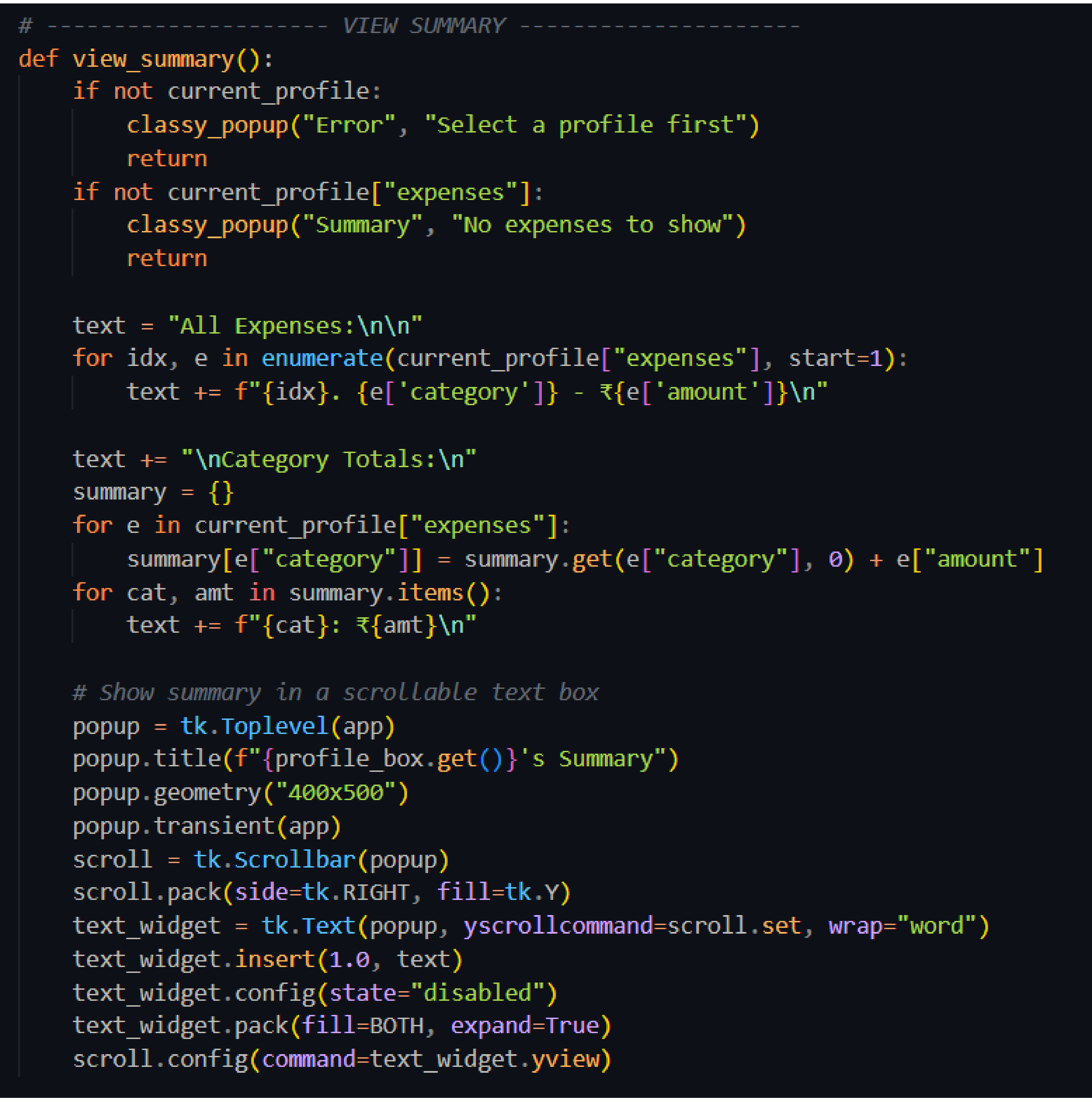


```python
# --------------------- ADD EXPENSE ---------------------
def add_expense():
    if not current_profile:
        classy_popup("Error", "Select a profile first")
        return
    try:
        amount = float(amount_entry.get())
        category = category_box.get()
        if not category:
            classy_popup("Warning", "Select a category")
            return
        current_profile["expenses"].append({"amount": amount, "category": category})
        amount_entry.delete(0, tk.END)
        refresh_table()
        update_totals()
        if category not in category_options:
            category_options.append(category)
            category_box['values'] = category_options
        classy_popup("Success", "Expense Added!")
    except:
        classy_popup("Error", "Enter a valid number")
```

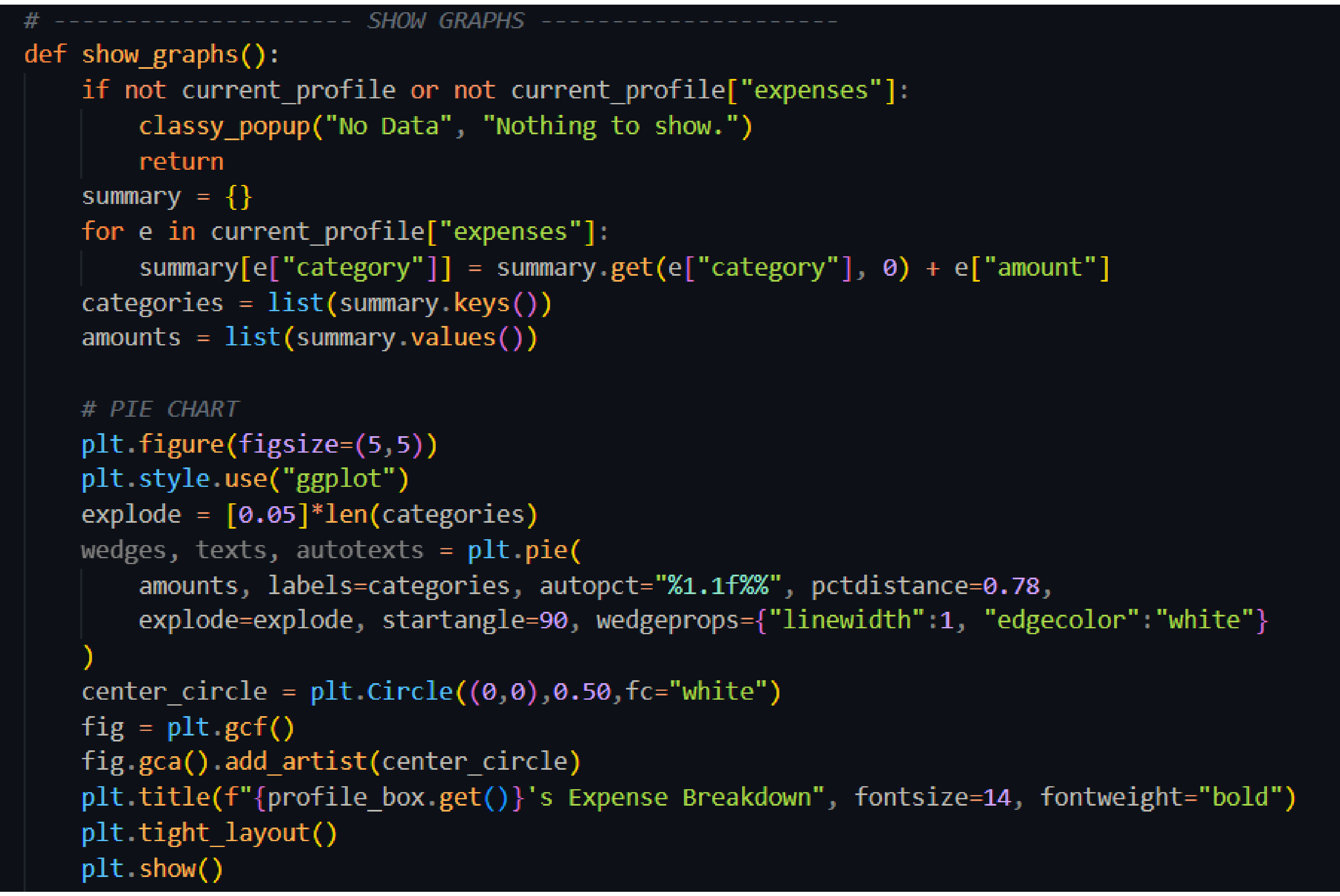

```python
# -------------------- REFRESH TABLE --------------------
def refresh_table():
    for row in table.get_children():
        table.delete(row)
    if current_profile:
        for e in current_profile["expenses"]:
            table.insert("", END, values=(e["category"], f"₹{e['amount']}"))

# -------------------- UPDATE TOTALS --------------------
def update_totals():
    if not current_profile:
        total_label.config(text="Total Spent: ₹0")
        budget_display_label.config(text="Budget: ₹0")
        limit_label.config(text="Budget Limit: ₹0")
        savings_label.config(text="Savings: ₹0")
        progress_bar.config(value=0)
        return

    total_spent = sum(e["amount"] for e in current_profile["expenses"])
    budget_display_label.config(text=f"Budget: ₹{current_profile['budget']}")
    limit_label.config(text=f"Budget Limit: ₹{current_profile['limit']}")
    total_label.config(text=f"Total Spent: ₹{total_spent}")
    savings = max(current_profile["budget"] - total_spent, 0)
    savings_label.config(text=f"Savings: ₹{savings}")

    if current_profile["limit"] > 0:
        percent = total_spent / current_profile["limit"]
        style = "success" if percent < 0.7 else "warning" if percent < 1 else "danger"
        progress_bar.config(value=min(total_spent, current_profile["limit"]), maximum=current_profile["limit"], bootstyle=style)
        if total_spent > current_profile["limit"]:
            alert_budget_popup("Budget Limit Exceeded", f"Exceeded your budget limit of ₹{current_profile['limit']}!")

    if current_profile["budget"] > 0 and total_spent > current_profile["budget"]:
        alert_budget_popup("Debt Alert!", f"You exceeded the budget of ₹{current_profile['budget']}, now you are in debt!")
```

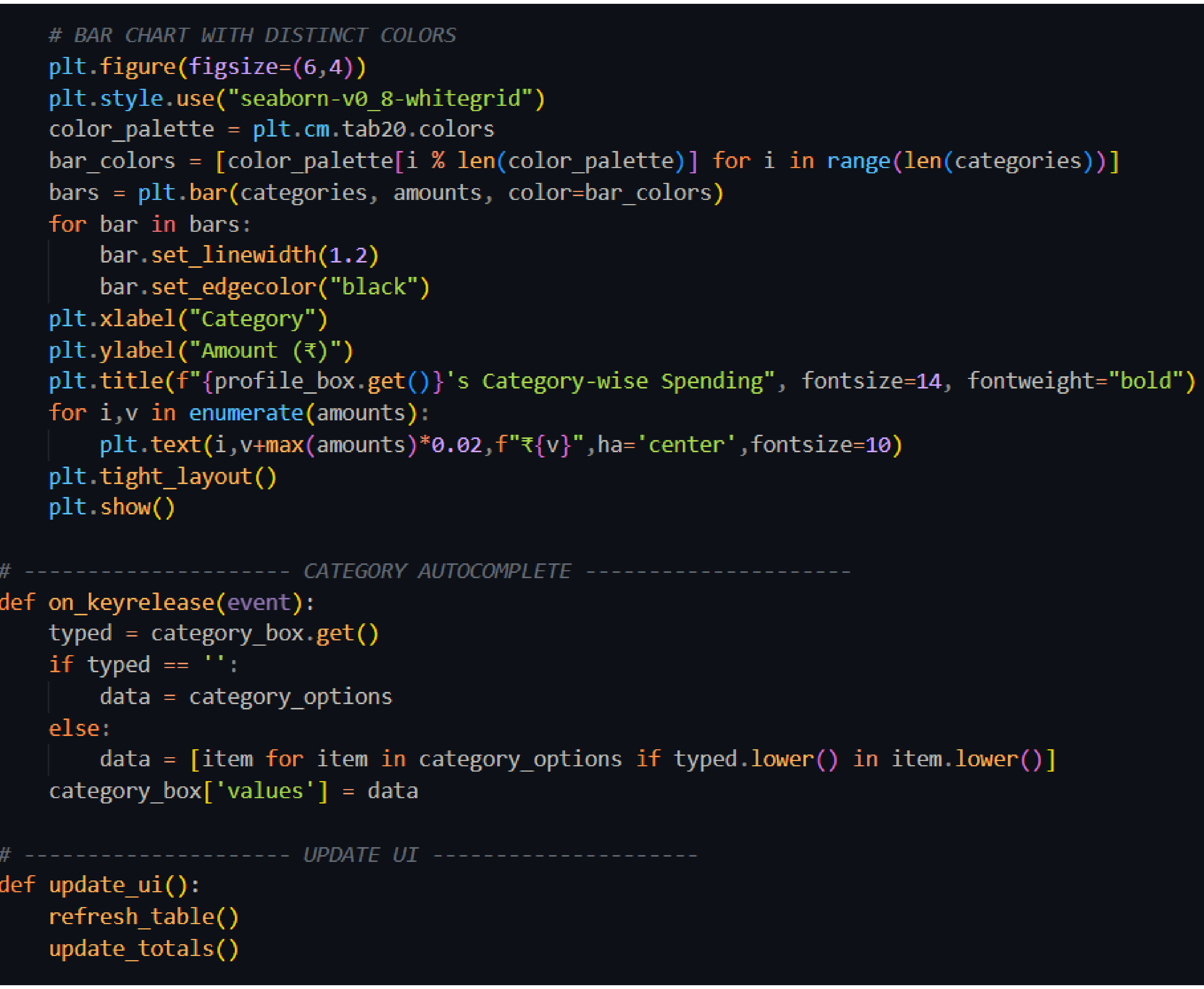


```
# -------------------- SET & RESET BUDGET --------------------
def set_budget():
    if not current_profile:
        classy_popup("Error", "Select a profile first")
        return
    try:
        b = float(budget_entry.get())
        l = float(limit_entry.get())
        current_profile["budget"] = b
        current_profile["limit"] = l
        budget_entry.delete(0, tk.END)
        limit_entry.delete(0, tk.END)
        update_totals()
        classy_popup("Success", f"Budget set to ₹{b} and Limit set to ₹{l}")
    except:
        classy_popup("Error", "Enter valid numbers")

def reset_budget():
    if not current_profile: return
    current_profile["budget"] = 0
    update_totals()
    classy_popup("Reset", "Budget reset to 0")

def reset_limit():
    if not current_profile: return
    current_profile["limit"] = 0
    update_totals()
    classy_popup("Reset", "Budget limit reset to 0")
```

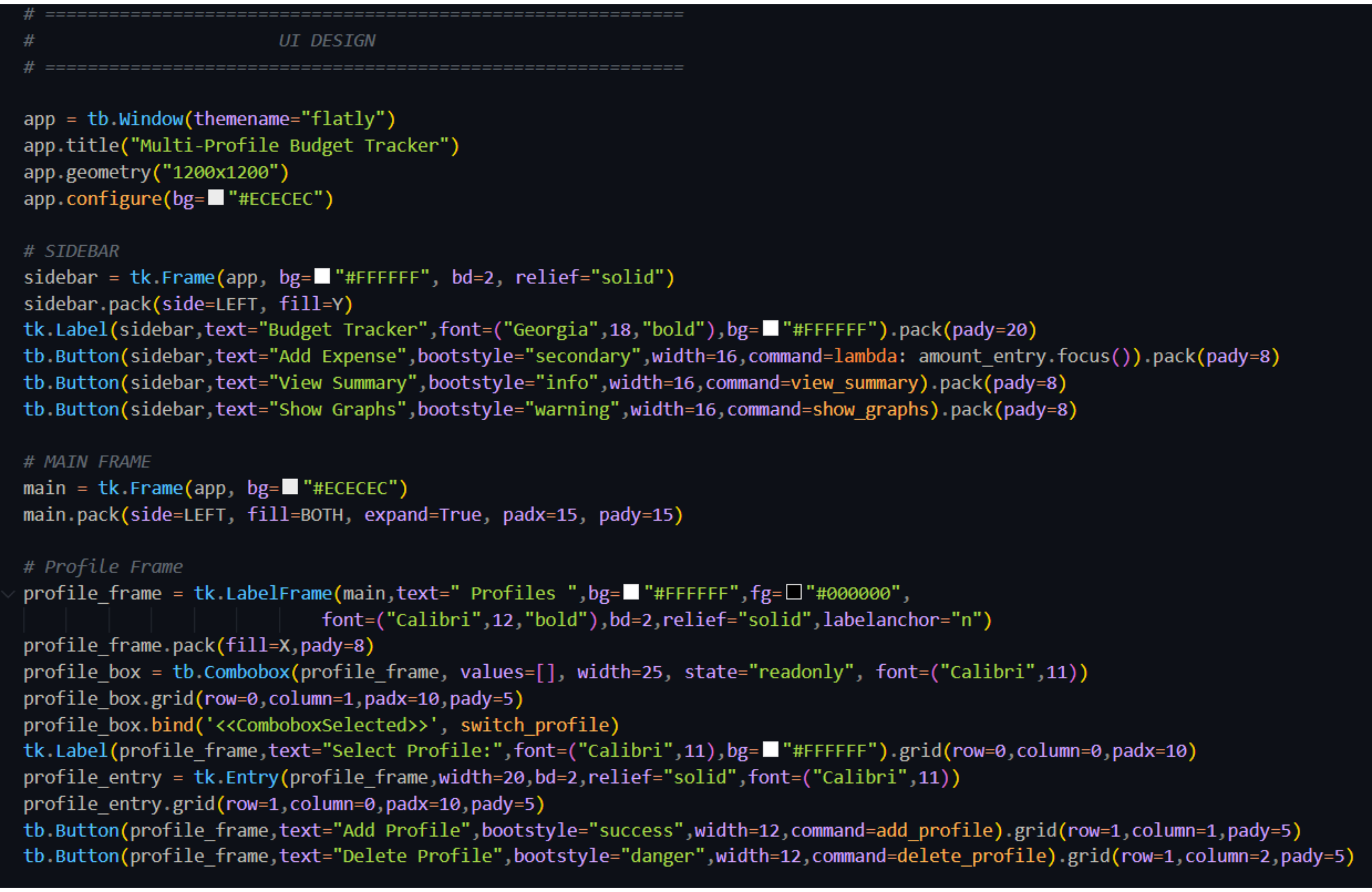


```python
# -------------------- VIEW SUMMARY --------------------
def view_summary():
    if not current_profile:
        classy_popup("Error", "Select a profile first")
        return
    if not current_profile["expenses"]:
        classy_popup("Summary", "No expenses to show")
        return

    text = "All Expenses:\n\n"
    for idx, e in enumerate(current_profile["expenses"], start=1):
        text += f"{idx}. {e['category']} - ₹{e['amount']}\n"

    text += "\nCategory Totals:\n"
    summary = {}
    for e in current_profile["expenses"]:
        summary[e["category"]] = summary.get(e["category"], 0) + e["amount"]
    for cat, amt in summary.items():
        text += f"{cat}: ₹{amt}\n"

    # Show summary in a scrollable text box
    popup = tk.Toplevel(app)
    popup.title(f"{profile_box.get()}'s Summary")
    popup.geometry("400x500")
    popup.transient(app)
    scroll = tk.Scrollbar(popup)
    scroll.pack(side=tk.RIGHT, fill=tk.Y)
    text_widget = tk.Text(popup, yscrollcommand=scroll.set, wrap="word")
    text_widget.insert(1.0, text)
    text_widget.config(state="disabled")
    text_widget.pack(fill=BOTH, expand=True)
    scroll.config(command=text_widget.yview)
```

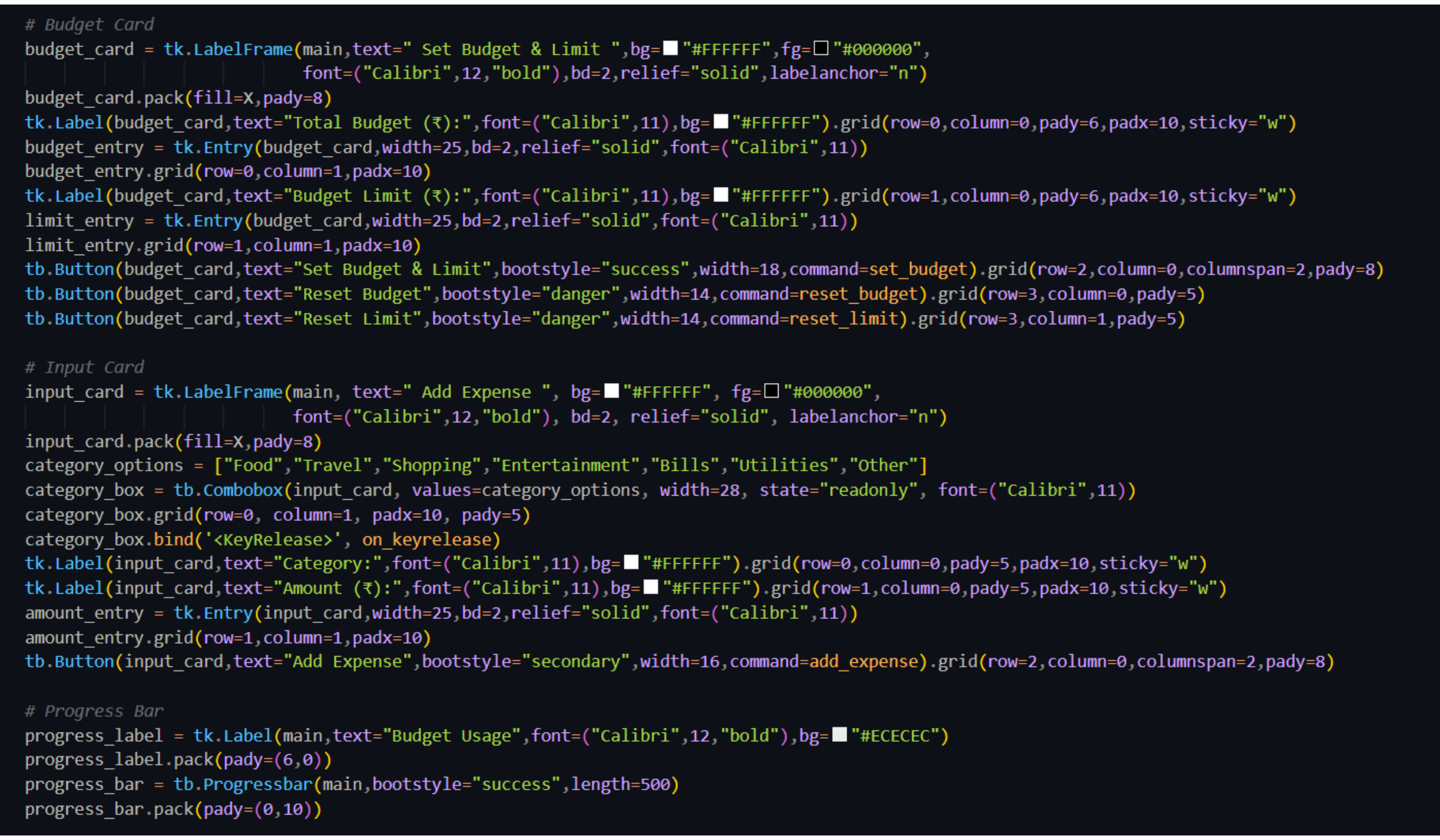

```python
# --------------------- SHOW GRAPHS ---------------------
def show_graphs():
    if not current_profile or not current_profile["expenses"]:
        classy_popup("No Data", "Nothing to show.")
        return
    summary = {}
    for e in current_profile["expenses"]:
        summary[e["category"]] = summary.get(e["category"], 0) + e["amount"]
    categories = list(summary.keys())
    amounts = list(summary.values())

    # PIE CHART
    plt.figure(figsize=(5,5))
    plt.style.use("ggplot")
    explode = [0.05]*len(categories)
    wedges, texts, autotexts = plt.pie(
        amounts, labels=categories, autopct="%1.1f%%", pctdistance=0.78,
        explode=explode, startangle=90, wedgeprops={"linewidth":1, "edgecolor":"white"}
    )
    center_circle = plt.Circle((0,0),0.50,fc="white")
    fig = plt.gcf()
    fig.gca().add_artist(center_circle)
    plt.title(f"{profile_box.get()}'s Expense Breakdown", fontsize=14, fontweight="bold")
    plt.tight_layout()
    plt.show()
```



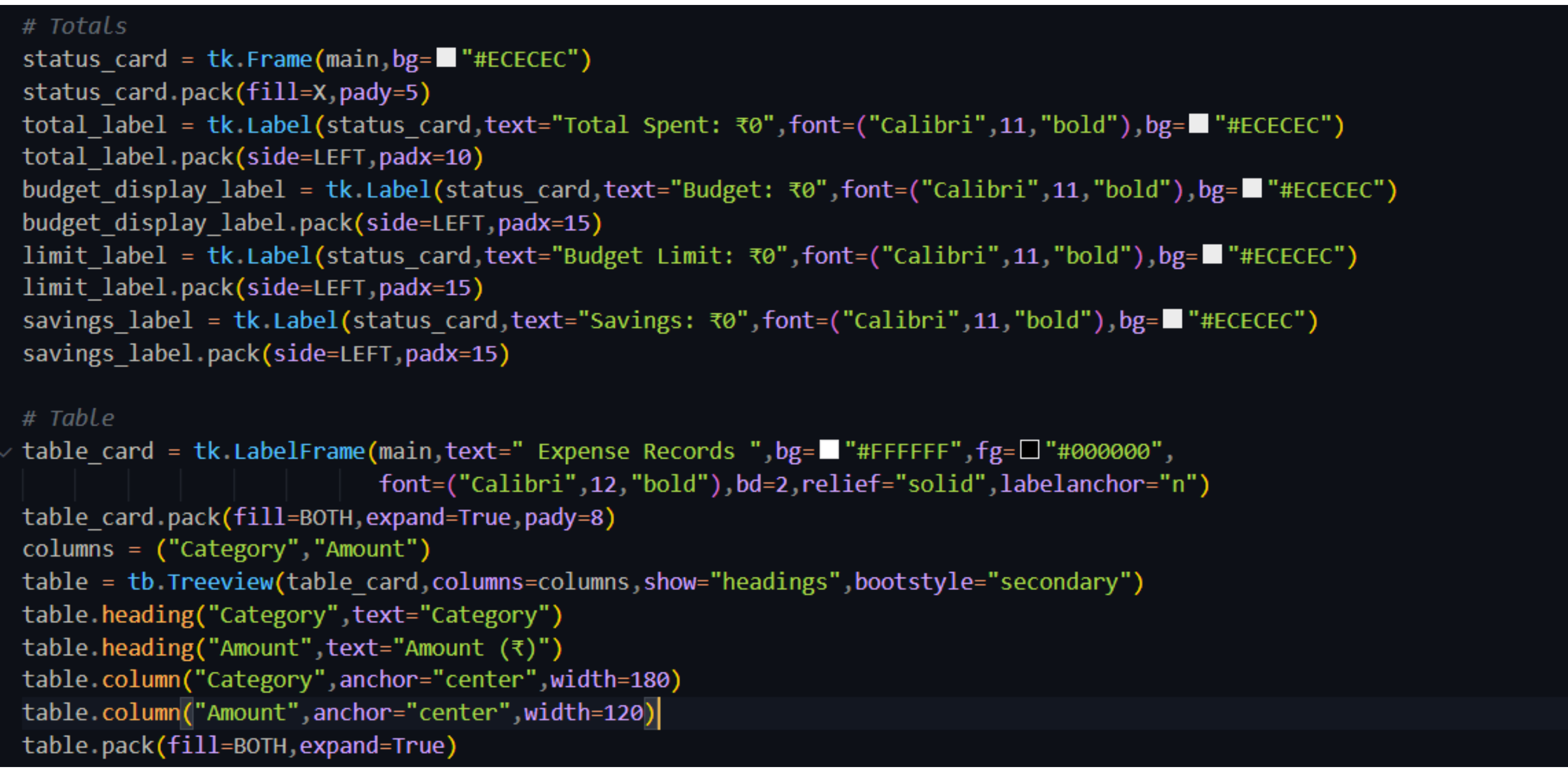
```python
    # BAR CHART WITH DISTINCT COLORS
    plt.figure(figsize=(6,4))
    plt.style.use("seaborn-v0_8-whitegrid")
    color_palette = plt.cm.tab20.colors
    bar_colors = [color_palette[i % len(color_palette)] for i in range(len(categories))]
    bars = plt.bar(categories, amounts, color=bar_colors)
    for bar in bars:
        bar.set_linewidth(1.2)
        bar.set_edgecolor("black")
    plt.xlabel("Category")
    plt.ylabel("Amount (₹)")
    plt.title(f"{profile_box.get()}'s Category-wise Spending", fontsize=14, fontweight="bold")
    for i,v in enumerate(amounts):
        plt.text(i,v+max(amounts)*0.02,f"₹{v}",ha='center',fontsize=10)
    plt.tight_layout()
    plt.show()

# -------------------- CATEGORY AUTOCOMPLETE --------------------
def on_keyrelease(event):
    typed = category_box.get()
    if typed == '':
        data = category_options
    else:
        data = [item for item in category_options if typed.lower() in item.lower()]
    category_box['values'] = data

# -------------------- UPDATE UI --------------------
def update_ui():
    refresh_table()
    update_totals()
```



```python
# ============================================================
#                       UI DESIGN
# ============================================================

app = tb.Window(themename="flatly")
app.title("Multi-Profile Budget Tracker")
app.geometry("1200x1200")
app.configure(bg="#ECECEC")

# SIDEBAR
sidebar = tk.Frame(app, bg="#FFFFFF", bd=2, relief="solid")
sidebar.pack(side=LEFT, fill=Y)
tk.Label(sidebar,text="Budget Tracker",font=("Georgia",18,"bold"),bg="#FFFFFF").pack(pady=20)
tb.Button(sidebar,text="Add Expense",bootstyle="secondary",width=16,command=lambda: amount_entry.focus()).pack(pady=8)
tb.Button(sidebar,text="View Summary",bootstyle="info",width=16,command=view_summary).pack(pady=8)
tb.Button(sidebar,text="Show Graphs",bootstyle="warning",width=16,command=show_graphs).pack(pady=8)

# MAIN FRAME
main = tk.Frame(app, bg="#ECECEC")
main.pack(side=LEFT, fill=BOTH, expand=True, padx=15, pady=15)

# Profile Frame
profile_frame = tk.LabelFrame(main,text=" Profiles ",bg="#FFFFFF",fg="#000000",
                            font=("Calibri",12,"bold"),bd=2,relief="solid",labelanchor="n")
profile_frame.pack(fill=X,pady=8)
profile_box = tb.Combobox(profile_frame, values=[], width=25, state="readonly", font=("Calibri",11))
profile_box.grid(row=0,column=1,padx=10,pady=5)
profile_box.bind('<<ComboboxSelected>>', switch_profile)
tk.Label(profile_frame,text="Select Profile:",font=("Calibri",11),bg="#FFFFFF").grid(row=0,column=0,padx=10)
profile_entry = tk.Entry(profile_frame,width=20,bd=2,relief="solid",font=("Calibri",11))
profile_entry.grid(row=1,column=0,padx=10,pady=5)
tb.Button(profile_frame,text="Add Profile",bootstyle="success",width=12,command=add_profile).grid(row=1,column=1,pady=5)
tb.Button(profile_frame,text="Delete Profile",bootstyle="danger",width=12,command=delete_profile).grid(row=1,column=2,pady=5)
```



```python
# Budget Card
budget_card = tk.LabelFrame(main,text=" Set Budget & Limit ",bg="#FFFFFF",fg="#000000",
                            font=("Calibri",12,"bold"),bd=2,relief="solid",labelanchor="n")
budget_card.pack(fill=X,pady=8)
tk.Label(budget_card,text="Total Budget (₹):",font=("Calibri",11),bg="#FFFFFF").grid(row=0,column=0,pady=6,padx=10,sticky="w")
budget_entry = tk.Entry(budget_card,width=25,bd=2,relief="solid",font=("Calibri",11))
budget_entry.grid(row=0,column=1,padx=10)
tk.Label(budget_card,text="Budget Limit (₹):",font=("Calibri",11),bg="#FFFFFF").grid(row=1,column=0,pady=6,padx=10,sticky="w")
limit_entry = tk.Entry(budget_card,width=25,bd=2,relief="solid",font=("Calibri",11))
limit_entry.grid(row=1,column=1,padx=10)
tb.Button(budget_card,text="Set Budget & Limit",bootstyle="success",width=18,command=set_budget).grid(row=2,column=0,columnspan=2,pady=8)
tb.Button(budget_card,text="Reset Budget",bootstyle="danger",width=14,command=reset_budget).grid(row=3,column=0,pady=5)
tb.Button(budget_card,text="Reset Limit",bootstyle="danger",width=14,command=reset_limit).grid(row=3,column=1,pady=5)

# Input Card
input_card = tk.LabelFrame(main, text=" Add Expense ", bg="#FFFFFF", fg="#000000",
                           font=("Calibri",12,"bold"), bd=2, relief="solid", labelanchor="n")
input_card.pack(fill=X,pady=8)
category_options = ["Food","Travel","Shopping","Entertainment","Bills","Utilities","Other"]
category_box = tb.Combobox(input_card, values=category_options, width=28, state="readonly", font=("Calibri",11))
category_box.grid(row=0, column=1, padx=10, pady=5)
category_box.bind('<KeyRelease>', on_keyrelease)
tk.Label(input_card,text="Category:",font=("Calibri",11),bg="#FFFFFF").grid(row=0,column=0,pady=5,padx=10,sticky="w")
tk.Label(input_card,text="Amount (₹):",font=("Calibri",11),bg="#FFFFFF").grid(row=1,column=0,pady=5,padx=10,sticky="w")
amount_entry = tk.Entry(input_card,width=25,bd=2,relief="solid",font=("Calibri",11))
amount_entry.grid(row=1,column=1,padx=10)
tb.Button(input_card,text="Add Expense",bootstyle="secondary",width=16,command=add_expense).grid(row=2,column=0,columnspan=2,pady=8)

# Progress Bar
progress_label = tk.Label(main,text="Budget Usage",font=("Calibri",12,"bold"),bg="#ECECEC")
progress_label.pack(pady=(6,0))
progress_bar = tb.Progressbar(main,bootstyle="success",length=500)
progress_bar.pack(pady=(0,10))
```



```python
# Totals
status_card = tk.Frame(main,bg="#ECECEC")
status_card.pack(fill=X,pady=5)
total_label = tk.Label(status_card,text="Total Spent: ₹0",font=("Calibri",11,"bold"),bg="#ECECEC")
total_label.pack(side=LEFT,padx=10)
budget_display_label = tk.Label(status_card,text="Budget: ₹0",font=("Calibri",11,"bold"),bg="#ECECEC")
budget_display_label.pack(side=LEFT,padx=15)
limit_label = tk.Label(status_card,text="Budget Limit: ₹0",font=("Calibri",11,"bold"),bg="#ECECEC")
limit_label.pack(side=LEFT,padx=15)
savings_label = tk.Label(status_card,text="Savings: ₹0",font=("Calibri",11,"bold"),bg="#ECECEC")
savings_label.pack(side=LEFT,padx=15)

# Table
table_card = tk.LabelFrame(main,text=" Expense Records ",bg="#FFFFFF",fg="#000000",
                           font=("Calibri",12,"bold"),bd=2,relief="solid",labelanchor="n")
table_card.pack(fill=BOTH,expand=True,pady=8)
columns = ("Category","Amount")
table = tb.Treeview(table_card,columns=columns,show="headings",bootstyle="secondary")
table.heading("Category",text="Category")
table.heading("Amount",text="Amount (₹)")
table.column("Category",anchor="center",width=180)
table.column("Amount",anchor="center",width=120)
table.pack(fill=BOTH,expand=True)
```