Project On
CSE-3632
Operating Systems Lab

Project Report : Task Manager Using Python

SUBMITTED TO —

Mohammad Zainal Abedin

Assistant Professor ,Dept of CSE

SUBMITTED BY—

Atique Shahriar	**C221068**
Anas Chowdhury	**C221064**

Project Name : Task Manager Using Python

Abstract

The Task Manager project is an abstraction of a system-level process management tool that simplifies interaction with complex system functionalities. It provides a user-friendly graphical interface for viewing and managing processes without requiring command-line knowledge or in-depth system understanding

Introduction

This project involves creating a Task Manager application using Python. The tool simulates a basic task manager functionality for viewing, managing, and terminating system processes. The application is built using the Tkinter library for GUI creation and psutil for process management.

Objectives

The main objectives of this project are:

- To develop a simple task manager application for viewing system processes.
- To enable process termination via a GUI interface.
- To implement features like refreshing the process list and displaying process details such as PID, name, CPU usage, and memory usage.

Tools and Technologies Used

- **Python**: The programming language used for developing the application.
- **Tkinter**: Python’s standard library for building graphical user interfaces.
- **psutil**: A Python library for system and process management, used to retrieve information about running processes.

Features of the Application

- **Process List**: Displays running processes with columns for PID, Name, CPU Usage, and Memory Usage.
- **Refresh Button**: Refreshes the process list to show the most current state of running processes.
- **Kill Process Button**: Terminates a selected process by its PID.
- **Error Handling**: Handles errors such as invalid selections, access denial, and process termination gracefully.

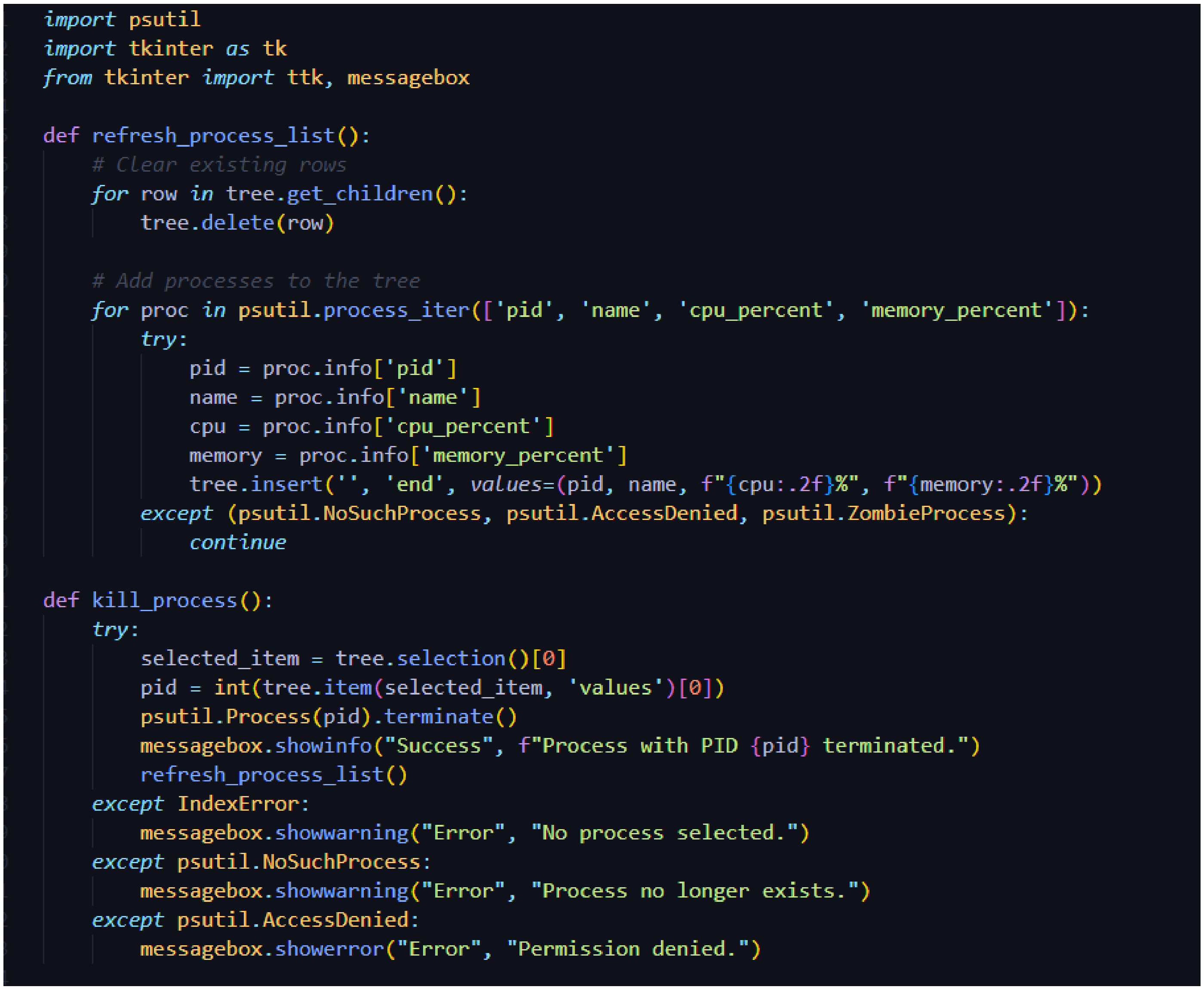
Implementation

Code Explanation:

- **Imports**:
  - psutil: For managing processes (retrieving details, terminating processes).
  - tkinter and ttk: For creating the GUI interface (buttons, tables).
  - messagebox: For showing alert dialogs.
- **refresh_process_list()**:
  - Retrieves and displays the current list of system processes.
  - Clears any previously displayed processes and updates the table with fresh data.
- **kill_process()**:
  - Terminates the selected process by PID.
  - Displays success or error messages based on the result.
- **GUI Components**:
  - **Treeview**: A table-like widget that displays process information.
  - **Buttons**: Two buttons—one for refreshing the process list and one for killing a selected process.
  - **Message Boxes**: For showing feedback, errors, or success message

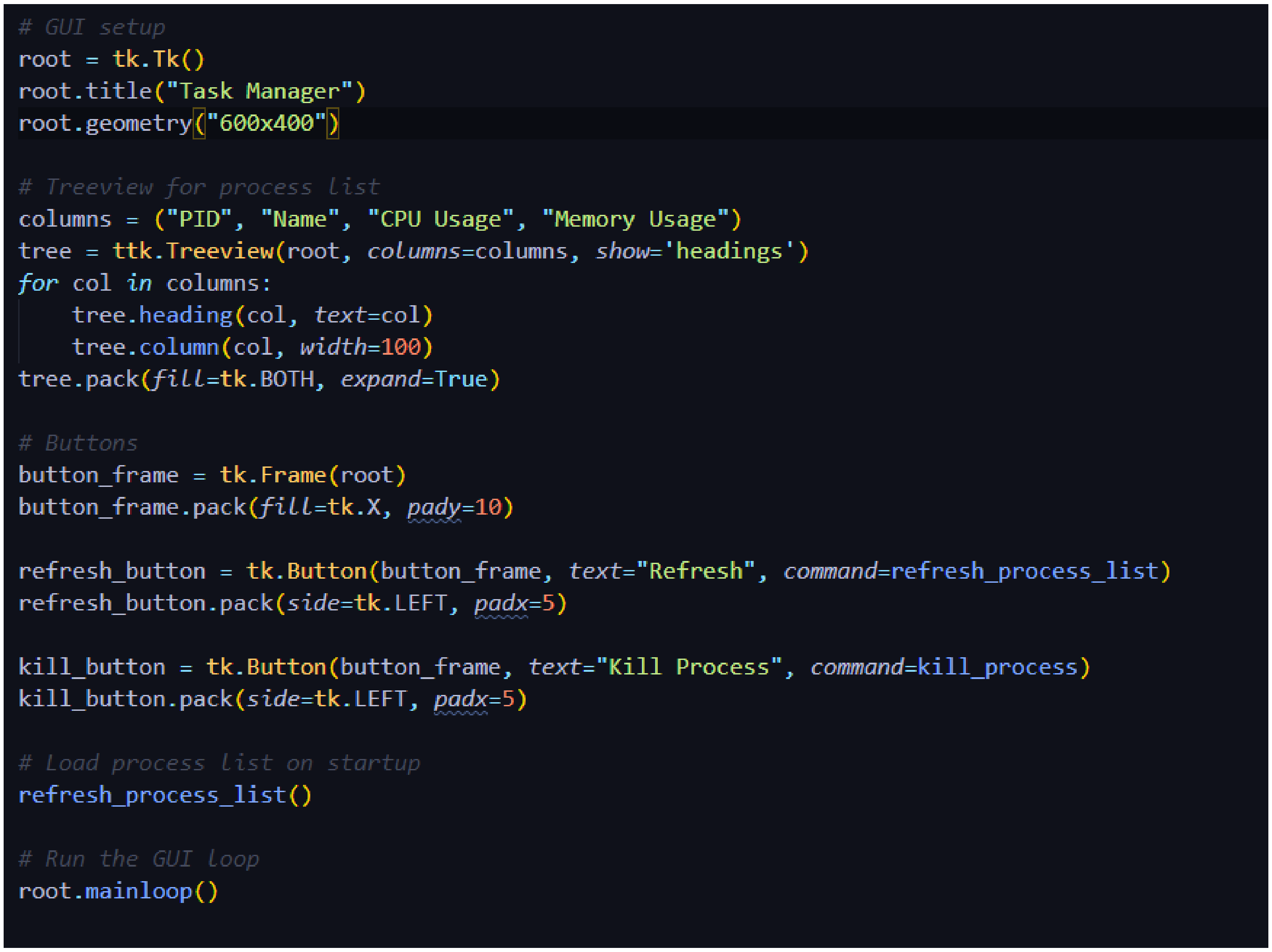
Code Structure

Main Function Implementation



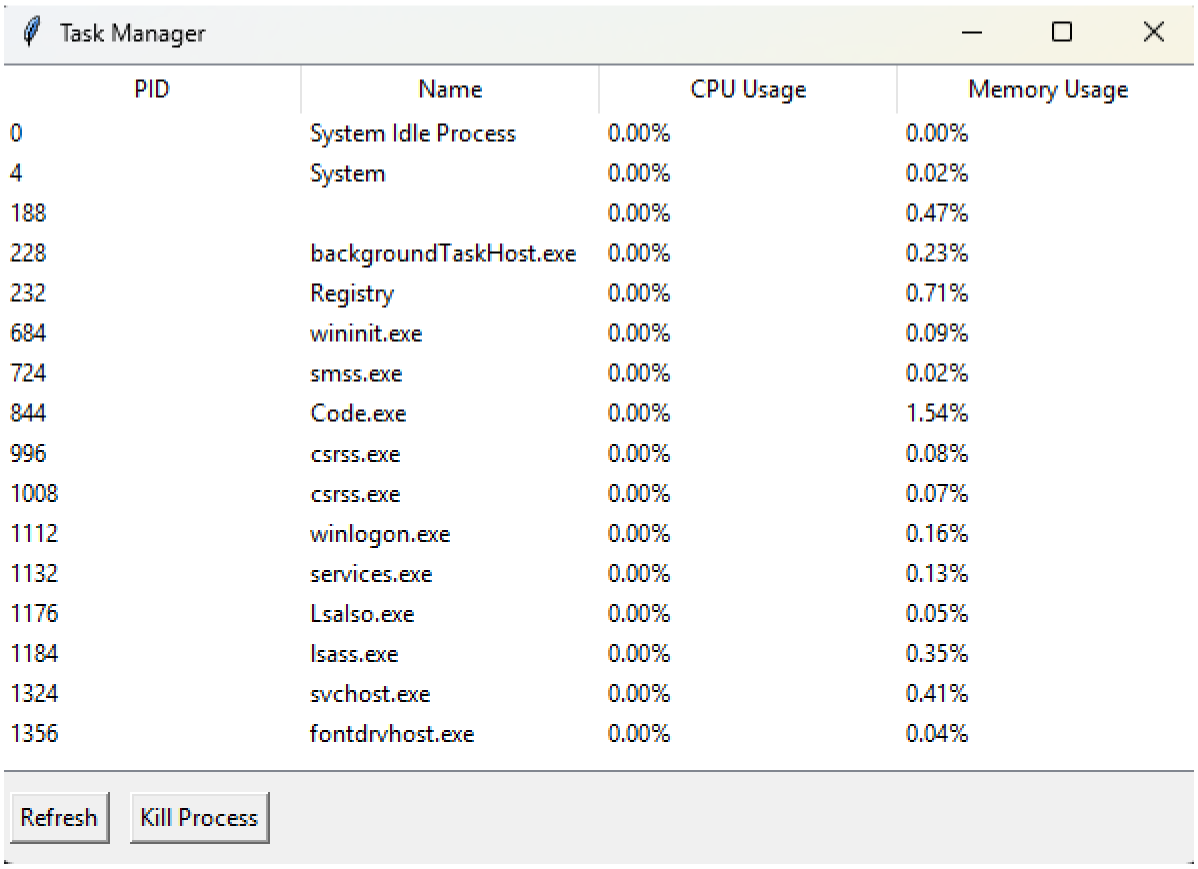
```python
import psutil
import tkinter as tk
from tkinter import ttk, messagebox

def refresh_process_list():
    # Clear existing rows
    for row in tree.get_children():
        tree.delete(row)

    # Add processes to the tree
    for proc in psutil.process_iter(['pid', 'name', 'cpu_percent', 'memory_percent']):
        try:
            pid = proc.info['pid']
            name = proc.info['name']
            cpu = proc.info['cpu_percent']
            memory = proc.info['memory_percent']
            tree.insert('', 'end', values=(pid, name, f"{cpu:.2f}%", f"{memory:.2f}%"))
        except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess):
            continue

def kill_process():
    try:
        selected_item = tree.selection()[0]
        pid = int(tree.item(selected_item, 'values')[0])
        psutil.Process(pid).terminate()
        messagebox.showinfo("Success", f"Process with PID {pid} terminated.")
        refresh_process_list()
    except IndexError:
        messagebox.showwarning("Error", "No process selected.")
    except psutil.NoSuchProcess:
        messagebox.showwarning("Error", "Process no longer exists.")
    except psutil.AccessDenied:
        messagebox.showerror("Error", "Permission denied.")
```

GUI Implementation



```
# GUI setup
root = tk.Tk()
root.title("Task Manager")
root.geometry("600x400")

# Treeview for process list
columns = ("PID", "Name", "CPU Usage", "Memory Usage")
tree = ttk.Treeview(root, columns=columns, show='headings')
for col in columns:
    tree.heading(col, text=col)
    tree.column(col, width=100)
tree.pack(fill=tk.BOTH, expand=True)

# Buttons
button_frame = tk.Frame(root)
button_frame.pack(fill=tk.X, pady=10)

refresh_button = tk.Button(button_frame, text="Refresh", command=refresh_process_list)
refresh_button.pack(side=tk.LEFT, padx=5)

kill_button = tk.Button(button_frame, text="Kill Process", command=kill_process)
kill_button.pack(side=tk.LEFT, padx=5)

# Load process list on startup
refresh_process_list()

# Run the GUI loop
root.mainloop()
```

Testing and Results



PID	Name	CPU Usage	Memory Usage
0	System Idle Process	0.00%	0.00%
4	System	0.00%	0.02%
188		0.00%	0.47%
228	backgroundTaskHost.exe	0.00%	0.23%
232	Registry	0.00%	0.71%
684	wininit.exe	0.00%	0.09%
724	smss.exe	0.00%	0.02%
844	Code.exe	0.00%	1.54%
996	csrss.exe	0.00%	0.08%
1008	csrss.exe	0.00%	0.07%
1112	winlogon.exe	0.00%	0.16%
1132	services.exe	0.00%	0.13%
1176	LsaIso.exe	0.00%	0.05%
1184	lsass.exe	0.00%	0.35%
1324	svchost.exe	0.00%	0.41%
1356	fontdrvhost.exe	0.00%	0.04%

Fig 1.1 GUI Output

Conclusion

This project successfully implements a basic task manager that can monitor and manage system processes. It demonstrates the power of Python in handling system-level tasks with a simple graphical user interface. Through the use of the psutil library, the task manager can access and manage processes, providing a practical tool for users.

Future Improvements

- Add more information about each process (e.g., memory usage in MB, status).
- Implement filtering and searching to easily find processes by name or PID.
- Add features like sorting the process list by CPU or memory usage.
- Include the ability to minimize and maximize the application window.

Necessary Link

- https://github.com/atique-vai/os_Lab_project
- https://docs.python.org/3/library/tkinter.html
- https://psutil.readthedocs.io/en/latest/