

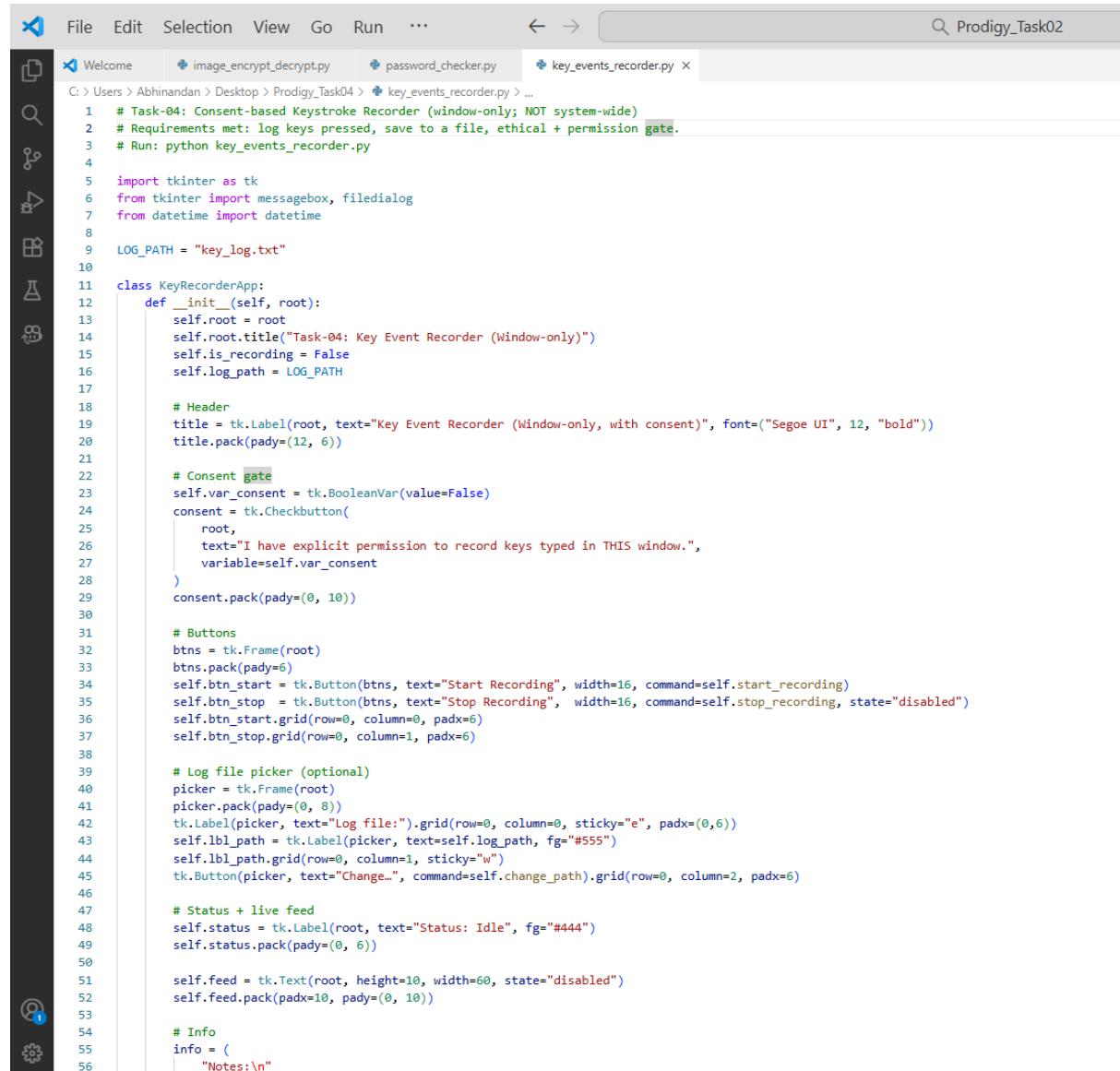
**Title:** Task-04 — Consent-based Keystroke Recorder (Window-Only)

**Objective:** Log keys pressed within a focused app window and save to a file with explicit consent and start/stop controls.

**Ethics & Scope:** Recording is limited to this window; shows consent checkbox and visible status; never record other users or apps without permission.

**Implementation:** Python tkinter; binds <KeyPress> on start; writes timestamped events to key\_log.txt; unbinds on stop.

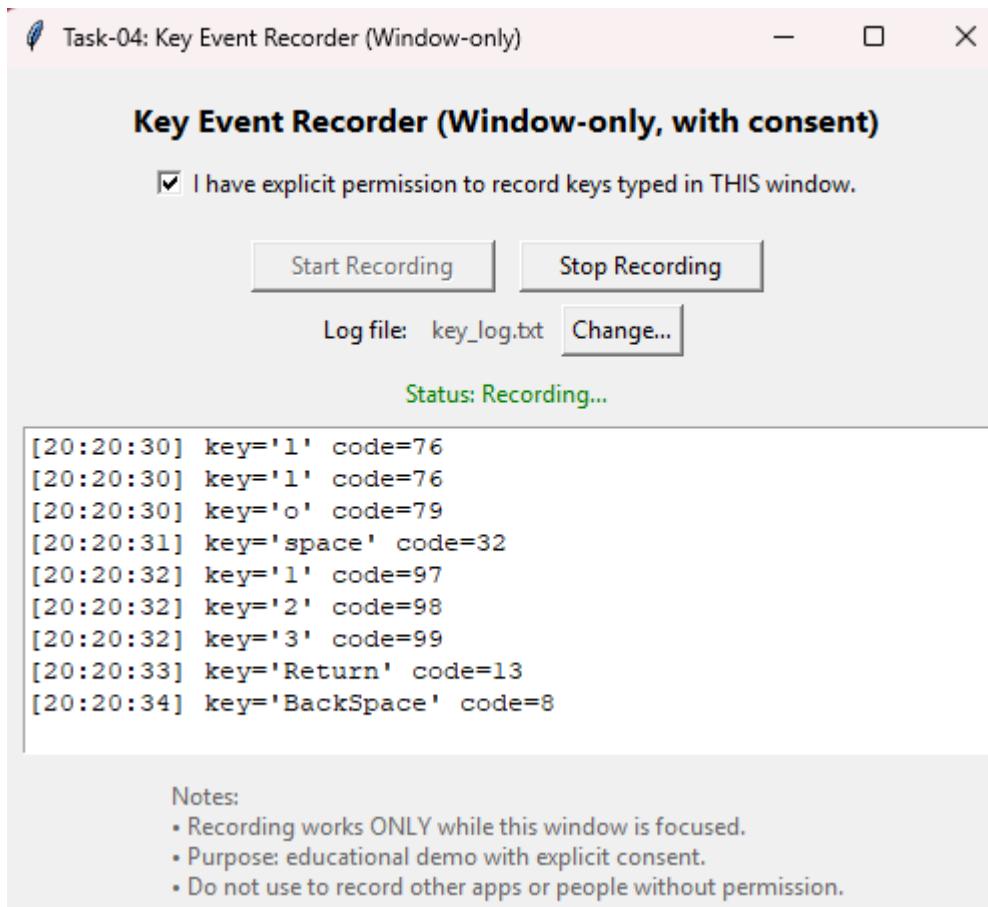
**Screenshots:**



```
C:\Users\Abhinandan>Desktop>Prodigy_Task04>key_events_recorder.py > ...
1  # Task-04: Consent-based Keystroke Recorder (window-only; NOT system-wide)
2  # Requirements met: log keys pressed, save to a file, ethical + permission gate.
3  # Run: python key_events_recorder.py
4
5  import tkinter as tk
6  from tkinter import messagebox, filedialog
7  from datetime import datetime
8
9  LOG_PATH = "key_log.txt"
10
11 class KeyRecorderApp:
12     def __init__(self, root):
13         self.root = root
14         self.root.title("Task-04: Key Event Recorder (Window-only)")
15         self.is_recording = False
16         self.log_path = LOG_PATH
17
18         # Header
19         title = tk.Label(root, text="Key Event Recorder (Window-only, with consent)", font=("Segoe UI", 12, "bold"))
20         title.pack(pady=(12, 6))
21
22         # Consent gate
23         self.var_consent = tk.BooleanVar(value=False)
24         consent = tk.Checkbutton(
25             root,
26             text="I have explicit permission to record keys typed in THIS window.",
27             variable=self.var_consent
28         )
29         consent.pack(pady=(0, 10))
30
31         # Buttons
32         btns = tk.Frame(root)
33         btns.pack(pady=6)
34         self.btn_start = tk.Button(btns, text="Start Recording", width=16, command=self.start_recording)
35         self.btn_stop = tk.Button(btns, text="Stop Recording", width=16, command=self.stop_recording, state="disabled")
36         self.btn_start.grid(row=0, column=0, padx=6)
37         self.btn_stop.grid(row=0, column=1, padx=6)
38
39         # Log file picker (optional)
40         picker = tk.Frame(root)
41         picker.pack(pady=(0, 8))
42         tk.Label(picker, text="Log file:").grid(row=0, column=0, sticky="e", padx=(0, 6))
43         self.lbl_path = tk.Label(picker, text=self.log_path, fg="#555")
44         self.lbl_path.grid(row=0, column=1, sticky="w")
45         tk.Button(picker, text="Change...", command=self.change_path).grid(row=0, column=2, padx=6)
46
47         # Status + live feed
48         self.status = tk.Label(root, text="Status: Idle", fg="#444")
49         self.status.pack(pady=(0, 6))
50
51         self.feed = tk.Text(root, height=10, width=60, state="disabled")
52         self.feed.pack(padx=10, pady=(0, 10))
53
54         # Info
55         info = (
56             "Notes:\n"
```

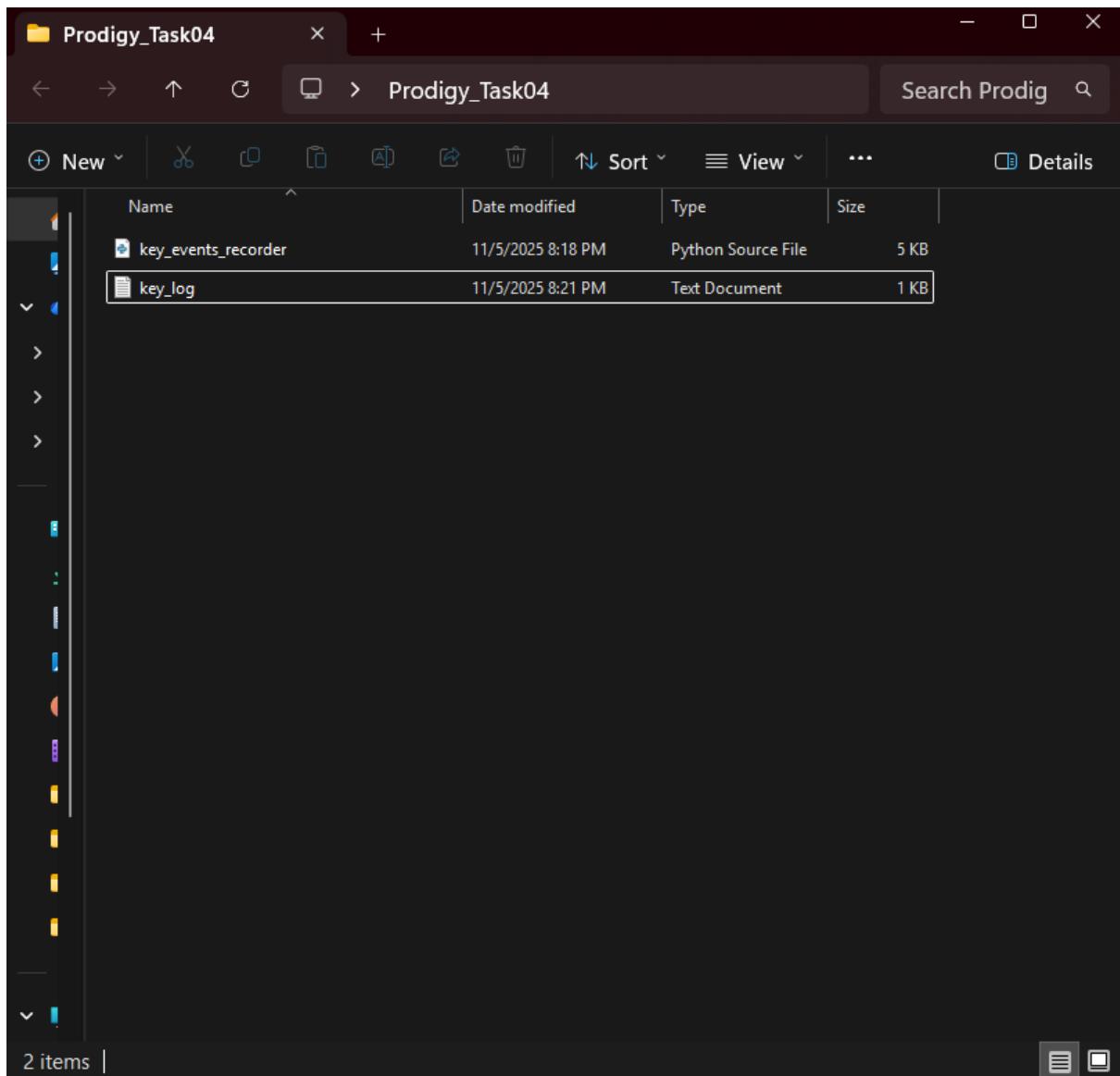
The screenshot shows a code editor window titled "Prodigy\_Task02". The file being edited is "key.events.recorder.py". The code is a Python script for a key recorder application. It includes imports for `tk`, `filedialog`, and `messagebox`. The script defines a class `KeyRecorderApp` with methods for initializing the window, saving log files, starting and stopping recording, handling key events, and closing the application. It uses Tkinter for the graphical interface and file operations.

```
C:\> Users > Abhinandan > Desktop > Prodigy_Task04 > key.events.recorder.py ...
11 class KeyRecorderApp:
12     def __init__(self, root):
13         ...
14         # Recording works ONLY while this window is focused.\n        # Purpose: educational demo with explicit consent.\n        # Do not use to record other apps or people without permission.\n    )
15     tk.Label(root, text=info, justify="left", fg="#666").pack(padx=10, pady=(0,10))
16
17     # Key binding (bound when recording)
18     self.root.protocol("WM_DELETE_WINDOW", self.on_close)
19
20     def change_path(self):
21         path = filedialog.asksaveasfilename(
22             title="Choose log file",
23             defaultextension=".txt",
24             filetypes=[("Text files", "*.txt"), ("All files", "*.*")]
25         )
26         if path:
27             self.log_path = path
28             self.lbl_path.config(text=path)
29
30     def start_recording(self):
31         if not self.var_consent.get():
32             messagebox.showwarning("Consent required", "Please tick the consent box before recording.")
33             return
34         self.is_recording = True
35         self.btn_start.config(state="disabled")
36         self.btn_stop.config(state="normal")
37         self.status.config(text="Status: Recording.", fg="green")
38         # Bind keypress only now
39         self.root.bind("<KeyPress>", self.on_key)
40
41         # Create/append file header
42         with open(self.log_path, "a", encoding="utf-8") as f:
43             f.write(f"\n--- Session start: {datetime.now().isoformat(timespec='seconds')}\n")
44
45     def stop_recording(self):
46         if self.is_recording:
47             self.is_recording = False
48             self.btn_start.config(state="normal")
49             self.btn_stop.config(state="disabled")
50             self.status.config(text="Status: Stopped", fg="#444")
51             # Unbind
52             self.root.unbind("<KeyPress>")
53             with open(self.log_path, "a", encoding="utf-8") as f:
54                 f.write(f"\n--- Session end: {datetime.now().isoformat(timespec='seconds')}\n")
55
56     def on_key(self, event: tk.Event):
57         if not self.is_recording:
58             return
59         key_sym = event.keysym           # e.g., a, A, space, Return
60         key_code = event.keycode         # platform-specific code
61         ts = datetime.now().strftime("%H:%M:%S")
62         line = f"[{ts}] key:{key_sym}r code:{key_code}\n"
63
64         # Append to file
65         with open(self.log_path, "a", encoding="utf-8") as f:
66             f.write(line)
67
68         # Append to on-screen feed
69         self.feed.config(state="normal")
70         self.feed.insert("end", line)
71         self.feed.see("end")
72         self.feed.config(state="disabled")
73
74     def on_close(self):
75         self.stop_recording()
76         self.root.destroy()
77
78 if __name__ == "__main__":
79     root = tk.Tk()
80     app = KeyRecorderApp(root)
81     root.mainloop()
```



```
--- Session start: 2025-11-05T20:20:19 ---
[20:20:26] key='h' code=72
[20:20:27] key='e' code=69
[20:20:30] key='l' code=76
[20:20:30] key='l' code=76
[20:20:30] key='o' code=79
[20:20:31] key='space' code=32
[20:20:32] key='1' code=97
[20:20:32] key='2' code=98
[20:20:32] key='3' code=99
[20:20:33] key='Return' code=13
[20:20:34] key='BackSpace' code=8
--- Session end: 2025-11-05T20:20:37 ---

--- Session start: 2025-11-05T20:21:16 ---
--- Session end: 2025-11-05T20:21:27 ---
```



**Conclusion:** Demonstrated safe, consent-based keystroke logging for educational purposes.