Bazla Bilquees

U3312671

Introduction To Information Technology

ASSIGNMENT # 02 – PYTHON FUNDAMENTAL

Case Study 2 – Smart Classroom Monitor

Step 1 — Understand the Problem:

Create a console program to monitor classroom state: projector status, topic, attendance management, temperature log and alert for over-capacity and out-of-range temperature. Produce a report summarizing the room state.

Step 2 — Inputs & Outputs

Variable	Device	Type	Unit	Logic	Description
Projector statue	Projector switch	Input	Boolean	On=1, OFF=0	Give the status of the project 1= On, 0=Off
Capacity	System Database	Input	Students/Persons	>or equal to 0	Maximum seating capacity of classroom
Attendance Names	RFID/Manual Entry	Input	Students	-	Students attending class
Topic	Instructor Input	Input	-	-	Lecture topic name
Temperature Log	Temperature Sensor	Input	C	Numeric	Records classroom temperature samples
Attendance Count	System (processor)	Output	Students	>or equal to 0	Number of students in the class
Capacity Alert	LCD/ Screen/ Alarm	Output	Boolean	Full=1, Normal=0	Warning if attendance exceeds capacity.
Temp Statics	System (processor)	Output	C	Min/Max/Avg	Temperature summary of classroom

Temp Alert	Alarm/SMS	Output	Boolean	Low=1, High=1, Normal=0	Alert if temperature <16C or >28C
Projector Reminder	Display	Output	Text	Reminder=1	If projector OFF while topic assigned

Step 3 — Algorithm

Initialized system state

- Set projector_ on = False
- Set topic = None
- Set attendance = {} (empty set)
- Preferably, set temperature_ log = [] (list).
- Maximum number of students allowed.

Main menu and menu options:

- Toggle projector
- Set topic
- Add student
- Remove student
- Add temperature reading
- Show report
- Exit

Continue till user selects Exit:

- If Toggle projector:
- Change projector on/off ON/Off.
- If Set topic:
- Enter a string value and save in topic.

If Add student:

- Input student name.
- Add name to attendance set.

If Remove student:

- Input student name.

- Check-out in case of attendance.

If Add temperature reading:

- Input float value (°C).
- Append to temperature_log.

If Show report:

- Record the number of students in attendance.
- count bigger than capacity: print "ROOM FULL".
- Computations of temperature statistics = (min,max,average).
- When any temperature of less than 16 C or more than 28 C occurs, then display temperature caution.
- When setting topic and projector off then display reminder Projector is OFF topic is set.

If Exit chosen:

- Stop the program.
- End.

Step 4 — Flow Chart

u3312671-Bazla Bilquees-Assignment

Step 4 – PSEUDOCODE

```
BEGIN
ROOM ← {'projector_on': False, 'capacity': 30, 'topic': ''}
attendance ← set()
temperatures ← []
WHILE True
print menu
GET choice
IF toggle projector → toggle_ projector()
IF set topic → set_ topic(input)
IF add student → add_ student(name) [check capacity]
IF remove student → remove_ student(name)
IF add temp → add_ temp(value) [warn if out-of-range]
IF report → report()
IF exit → break
```

END WHILE
END

Step 5 Python Code

```
#Name: Bazla Bilquees
#Student ID: u3312671
#Case Study 2: Smart Classroom Monitor (robust implementation)
import sys
import traceback
from datetime import datetime
# -Room state and data structures -
ROOM = {
  "projector_on": False,   # bool
  "capacity": 30,        # int
  "topic": ""           # str
}

attendance = set()      # set of student names (strings)
temperatures = []       # list of float temperature readings (°C)



# ---------- Input helpers ----------
def safe_input(prompt):
  """
  Wrapper for input() which handles Ctrl+C / Ctrl+D gracefully.
  Returns a stripped string, or None if the user cancelled.
  """
```

```
    try:
        s = input(prompt)
    except (EOFError, KeyboardInterrupt):
        print("\nInput cancelled by user. Exiting program.")
        return None
    return s.strip()
def get_nonempty_string(prompt):
    while True:
        s = safe_input(prompt)
        if s is None:
            return None
        if s != "":
            return s
        print("Please enter a non-empty value.")
# - Classroom operations -
def toggle_projector():
    ROOM["projector_on"] = not ROOM["projector_on"]
    state = "ON" if ROOM["projector_on"] else "OFF"
    print(f"Projector switched {state}.")



def set_topic():
    topic = get_nonempty_string("Enter lecture topic (or blank to cancel): ")
    if topic is None:
        return
    ROOM["topic"] = topic
```

```
    print(f"Topic set to: {ROOM['topic']}")
def add_student():
    name = get_nonempty_string("Enter student name to add: ")
    if name is None:
        return
    if len(attendance) >= ROOM["capacity"]:
        print("⚠ ROOM FULL — cannot add more students.")
        return
    if name in attendance:
        print(f"{name} is already marked present (no duplicate entries).")
    else:
        attendance.add(name)
        print(f"Added {name}. Attendance count: {len(attendance)}/{ROOM['capacity']}")
def remove_student():
    name = get_nonempty_string("Enter student name to remove: ")
    if name is None:
        return
    if name in attendance:
        attendance.remove(name)
        print(f"Removed {name}. Attendance count: {len(attendance)}/{ROOM['capacity']}")
    else:
        print(f"{name} not found in attendance.")
def add_temperature():
    s = safe_input("Enter temperature reading in °C (e.g. 22.5): ")
    if s is None:
        return
```

```
    try:
        t = float(s)
    except ValueError:
        print("Invalid number. Please enter a numeric temperature (e.g. 21.3).")
        return
    temperatures.append(t)
    print(f"Temperature {t:.1f}°C added. Total readings: {len(temperatures)}")
    if t < 16 or t > 28:
        print("⚠ Temperature out of recommended range (<16°C or >28°C).")
def temp_stats():
    if not temperatures:
        return None, None, None
    tmin = min(temperatures)
    tmax = max(temperatures)
    tavg = sum(temperatures) / len(temperatures)
    return tmin, tmax, tavg
def show_temperature_stats():
    tmin, tmax, tavg = temp_stats()
    if tmin is None:
        print("No temperature readings available.")
        return
    print(f"Temperature stats — Min: {tmin:.2f}°C, Max: {tmax:.2f}°C, Avg: {tavg:.2f}°C")
    if tmin < 16 or tmax > 28:
        print("⚠ ALERT: Temperature readings out of recommended range.")
def report():
    print("\n" + "=" * 36)
```

```
    print("CLASSROOM REPORT")
    print(f"Projector: {'ON' if ROOM['projector_on'] else 'OFF'}")
    print(f"Topic: {ROOM['topic'] or '<none>'}")
    print(f"Capacity: {ROOM['capacity']}")
    print(f"Attendance ({len(attendance)}): {', '.join(sorted(attendance)) or '<none>'}")
    tmin, tmax, tavg = temp_stats()
    if tmin is None:
        print("Temperature readings: <none>")
    else:
        print(f"Temperature: min={tmin:.2f}°C, max={tmax:.2f}°C, avg={tavg:.2f}°C")
        if tmin < 16 or tmax > 28:
            print("⚠ Temperature ALERT: values out of range (<16°C or >28°C).")
    # Alerts
    if len(attendance) > ROOM["capacity"]:
        print("⚠ ROOM OVER CAPACITY!")
    if ROOM["topic"] and not ROOM["projector_on"]:
        print("⚠ Reminder: Topic is set but projector is OFF.")
    print("=" * 36 + "\n")


# -Utility: change capacity (optional) -
def set_capacity():
    s = safe_input(f"Enter new capacity (current {ROOM['capacity']}) or blank to cancel: ")
    if s is None or s == "":
        return
    try:
```

```
        val = int(s)
        if val < 0:
            print("Capacity must be a non-negative integer.")
            return
    except ValueError:
        print("Please enter an integer value.")
        return
    ROOM['capacity'] = val
    print(f"Capacity set to {ROOM['capacity']}")
# - Main program loop -
def main():
    MENU_TEXT = """\nSMART CLASSROOM MONITOR
1) Toggle projector
2) Set topic
3) Add student
4) Remove student
5) Add temperature reading
6) Show temperature stats
7) Show full report
8) Set capacity (optional)
9) Exit
"""
    while True:
        print(MENU_TEXT)
        choice = safe_input("Choose option (1-9): ")
        if choice is None:
```

```
    # user cancelled input (Ctrl+C / Ctrl+D)
    break
choice = choice.strip()
if choice == "1":
    toggle_projector()
elif choice == "2":
    set_topic()
elif choice == "3":
    add_student()
elif choice == "4":
    remove_student()
elif choice == "5":
    add_temperature()
elif choice == "6":
    show_temperature_stats()
elif choice == "7":
    report()
elif choice == "8":
    set_capacity()
elif choice == "9":
    print("Exiting Smart Classroom Monitor. Goodbye!")
    break
else:
    # allow text commands too
    cmd = choice.lower()
    if cmd in ("toggle", "projector"):
```

```
            toggle_projector()
        elif cmd == "exit":
            break
        else:
            print("Invalid choice — enter 1-9 or a command (toggle/exit).")
# -Program entry with top-level exception handling -
if __name__ == "__main__":
    try:
        main()
    except Exception as e:
        # print friendly error message and record full traceback to a log file
        print("An unexpected error occurred. A log file 'classroom_error.log' was created.")
        with open("classroom_error.log", "a", encoding="utf-8") as f:
            f.write(f"\n[{datetime.now().isoformat()}] Unhandled exception:\n")
            traceback.print_exc(file=f)
        # also print the traceback snippet to console for quick debugging
        traceback.print_exc()
        sys.exit(1)
```

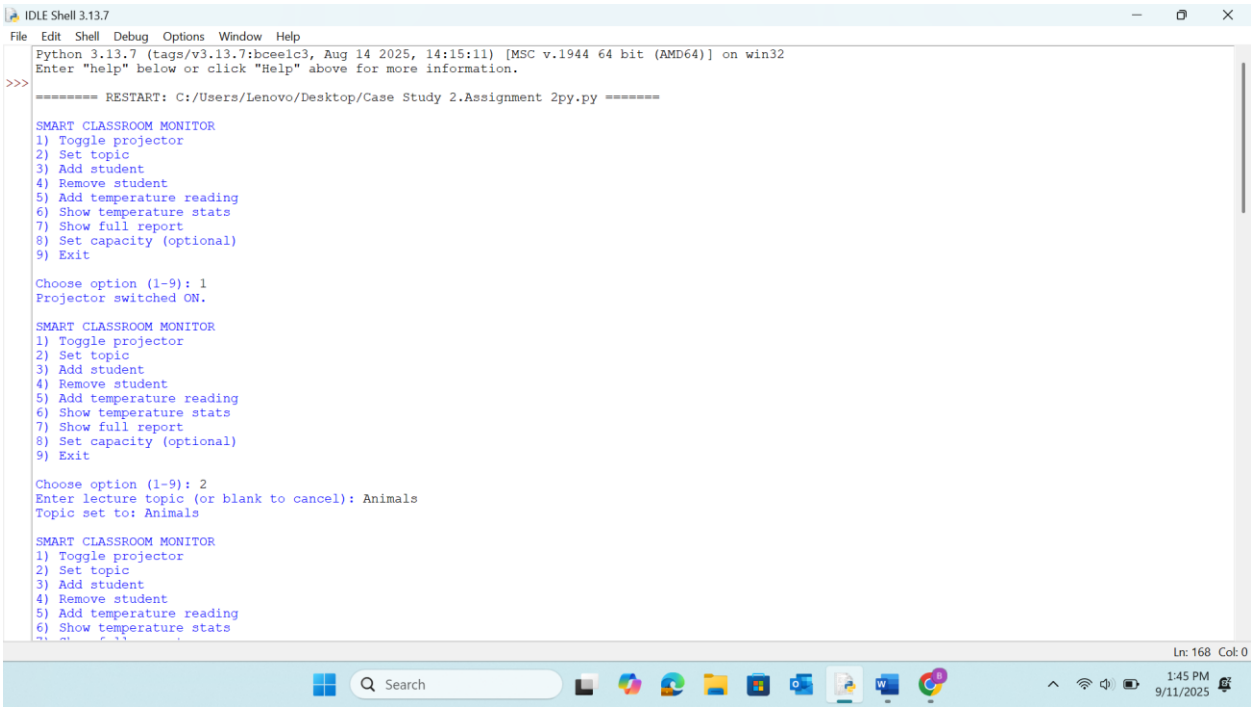
Step 6 -Testing: handwritten expected results + test runs & notes

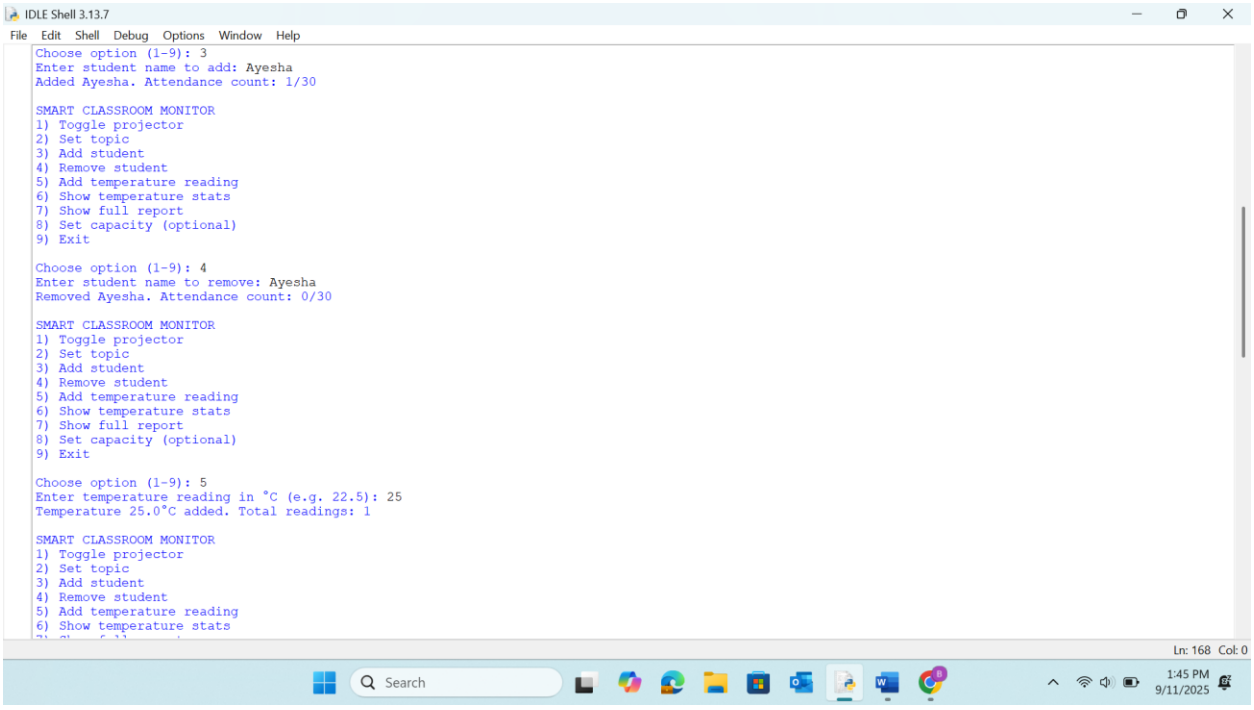
u3312671-Bazla Bilquees-Case study 2

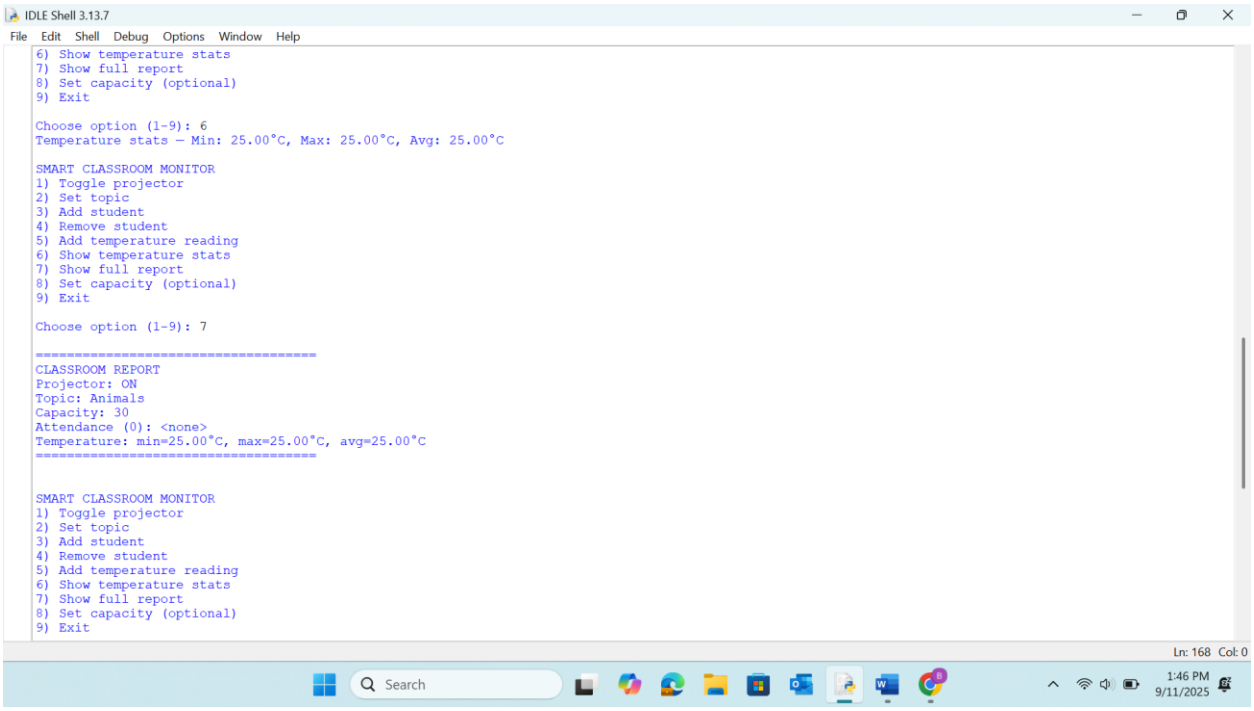
U3312671 - Bazia Bilqees IIT

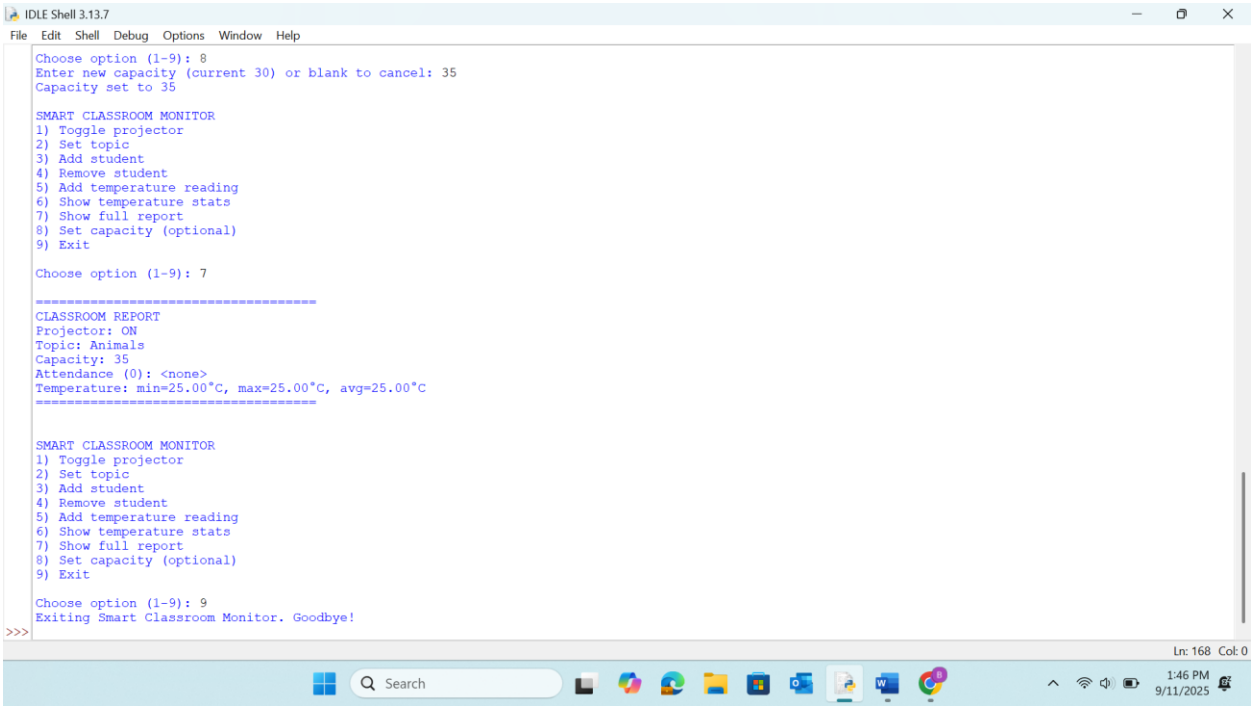
Case Study - II - Assignment 2

STEP#06 Test No.	Action / Input	Expected Result	Actual Result	Status.
1.	Start Program	Menu will displayed with 0-9 options	Displayed Correctly	Pass.
2.	Projector Status	1st press → ON, 2nd press → Off	Messages Correct	Pass.
3.	Enter Topic "Animals"	Output shows "Topic set to Animals".	Matches	Pass
4.	Add student name "Ayesha"	student list shows "Ayesha" added	Matches	Pass.
5.	Remove the name of student "Ayesha"	Name Removed	Matches	Pass.
6.	Add temperature reading '26'.	Value stored	Matches	Pass.
7.	Show statistic	Displays topic, projector status, total student, avg. temp	Matches.	Pass.
8.	Generate Complete Report	Formatted report	Matches	Pass.
9.	Exit	Program terminates without error	Matches	Pass.


IDLE Shell 3.13.7
File Edit Shell Debug Options Window Help
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
======== RESTART: C:/Users/Lenovo/Desktop/Case Study 2.Assignment 2py.py ========
SMART CLASSROOM MONITOR
1) Toggle projector
2) Set topic
3) Add student
4) Remove student
5) Add temperature reading
6) Show temperature stats
7) Show full report
8) Set capacity (optional)
9) Exit
Choose option (1-9): 1
Projector switched ON.
SMART CLASSROOM MONITOR
1) Toggle projector
2) Set topic
3) Add student
4) Remove student
5) Add temperature reading
6) Show temperature stats
7) Show full report
8) Set capacity (optional)
9) Exit
Choose option (1-9): 2
Enter lecture topic (or blank to cancel): Animals
Topic set to: Animals
SMART CLASSROOM MONITOR
1) Toggle projector
2) Set topic
3) Add student
4) Remove student
5) Add temperature reading
6) Show temperature stats
Ln: 168 Col: 0
Search
1:45 PM
9/11/2025


IDLE Shell 3.13.7
File Edit Shell Debug Options Window Help
Choose option (1-9): 3
Enter student name to add: Ayesha
Added Ayesha. Attendance count: 1/30
SMART CLASSROOM MONITOR
1) Toggle projector
2) Set topic
3) Add student
4) Remove student
5) Add temperature reading
6) Show temperature stats
7) Show full report
8) Set capacity (optional)
9) Exit
Choose option (1-9): 4
Enter student name to remove: Ayesha
Removed Ayesha. Attendance count: 0/30
SMART CLASSROOM MONITOR
1) Toggle projector
2) Set topic
3) Add student
4) Remove student
5) Add temperature reading
6) Show temperature stats
7) Show full report
8) Set capacity (optional)
9) Exit
Choose option (1-9): 5
Enter temperature reading in °C (e.g. 22.5): 25
Temperature 25.0°C added. Total readings: 1
SMART CLASSROOM MONITOR
1) Toggle projector
2) Set topic
3) Add student
4) Remove student
5) Add temperature reading
6) Show temperature stats
Ln: 168 Col: 0
Search
1:45 PM
9/11/2025


IDLE Shell 3.13.7
File Edit Shell Debug Options Window Help
6) Show temperature stats
7) Show full report
8) Set capacity (optional)
9) Exit
Choose option (1-9): 6
Temperature stats — Min: 25.00°C, Max: 25.00°C, Avg: 25.00°C
SMART CLASSROOM MONITOR
1) Toggle projector
2) Set topic
3) Add student
4) Remove student
5) Add temperature reading
6) Show temperature stats
7) Show full report
8) Set capacity (optional)
9) Exit
Choose option (1-9): 7
==========
CLASSROOM REPORT
Projector: ON
Topic: Animals
Capacity: 30
Attendance (0): <none>
Temperature: min=25.00°C, max=25.00°C, avg=25.00°C
==========
SMART CLASSROOM MONITOR
1) Toggle projector
2) Set topic
3) Add student
4) Remove student
5) Add temperature reading
6) Show temperature stats
7) Show full report
8) Set capacity (optional)
9) Exit
Ln: 168 Col: 0
Search
1:46 PM
9/11/2025


IDLE Shell 3.13.7
File Edit Shell Debug Options Window Help
Choose option (1-9): 8
Enter new capacity (current 30) or blank to cancel: 35
Capacity set to 35
SMART CLASSROOM MONITOR
1) Toggle projector
2) Set topic
3) Add student
4) Remove student
5) Add temperature reading
6) Show temperature stats
7) Show full report
8) Set capacity (optional)
9) Exit
Choose option (1-9): 7
==========
CLASSROOM REPORT
Projector: ON
Topic: Animals
Capacity: 35
Attendance (0): <none>
Temperature: min=25.00°C, max=25.00°C, avg=25.00°C
==========
SMART CLASSROOM MONITOR
1) Toggle projector
2) Set topic
3) Add student
4) Remove student
5) Add temperature reading
6) Show temperature stats
7) Show full report
8) Set capacity (optional)
9) Exit
Choose option (1-9): 9
Exiting Smart Classroom Monitor. Goodbye!
>>>
Ln: 168 Col: 0
Search
1:46 PM
9/11/2025

Step 7- Truth Table & Boolean Expression:

The classroom is “ready to generate a valid report” only when:

- Projector is ON (P)
- Topic is set (T)
- At least one student is present (S)
- At least one temperature reading exists (R)

Therefore:

The Boolean expression is:

M= P.T.S.R (Where “·” means logical **AND**)

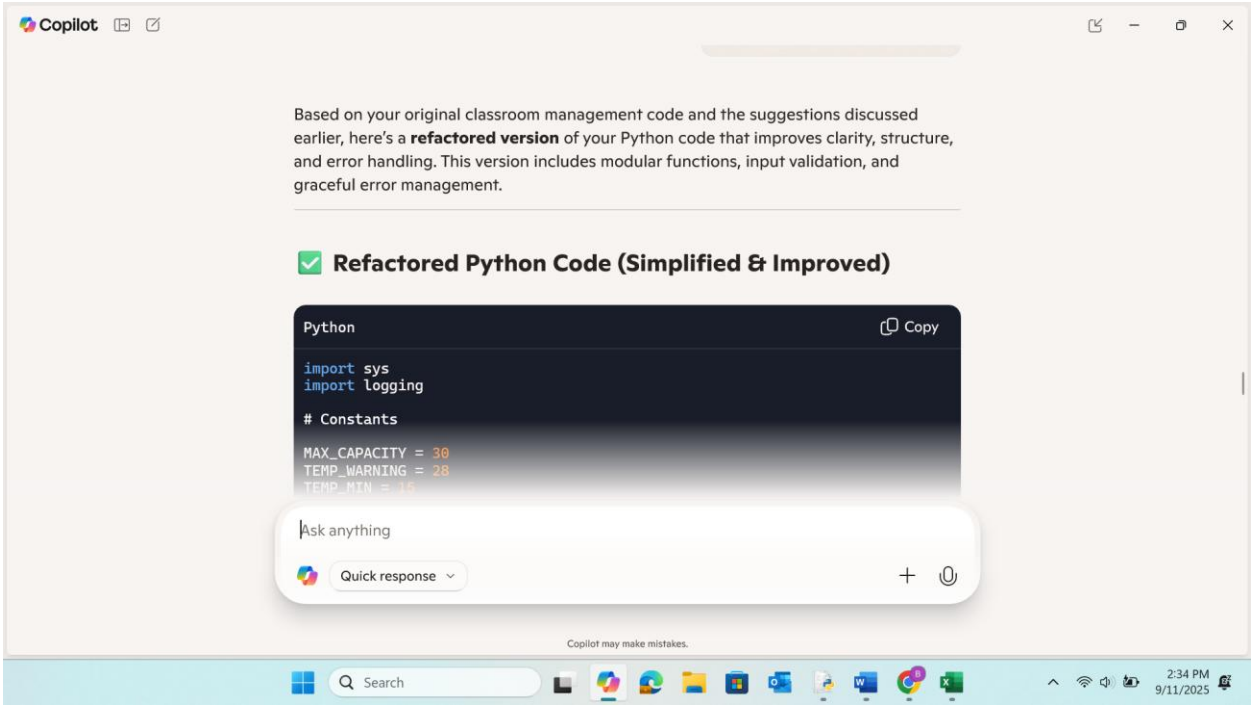
P	T	S	R	MON
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

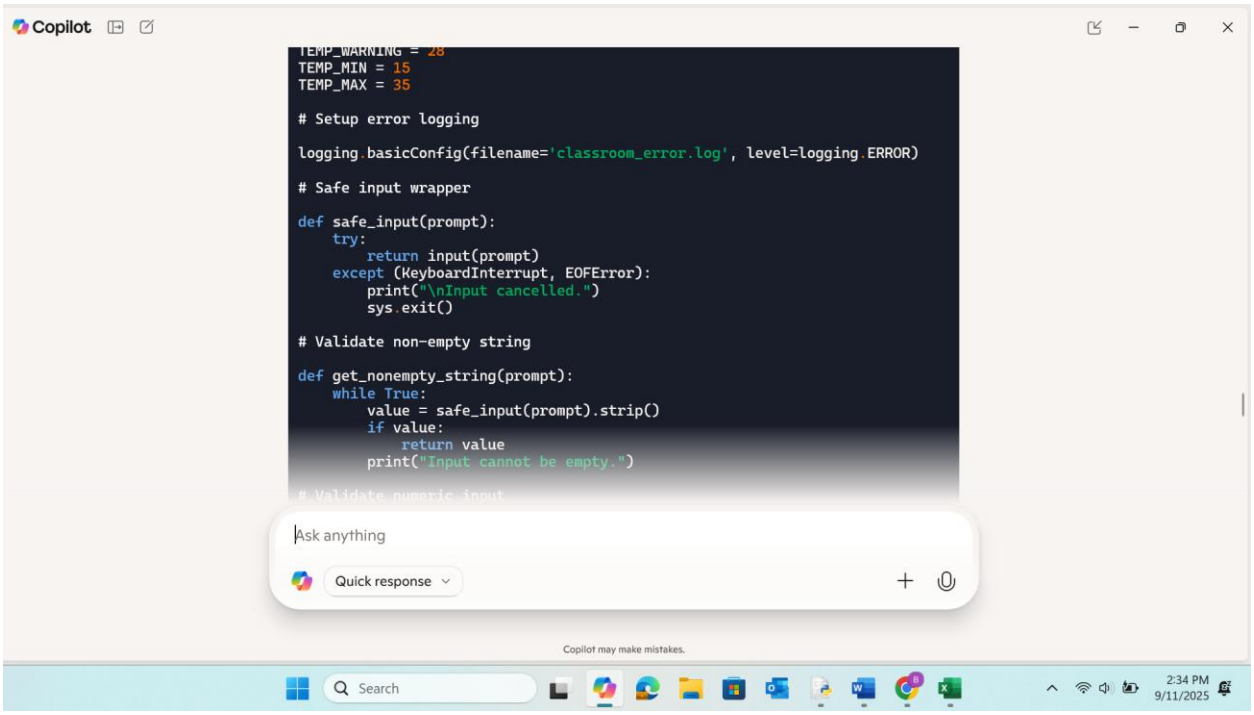
Step 8 – LOGIC DIAGRAM

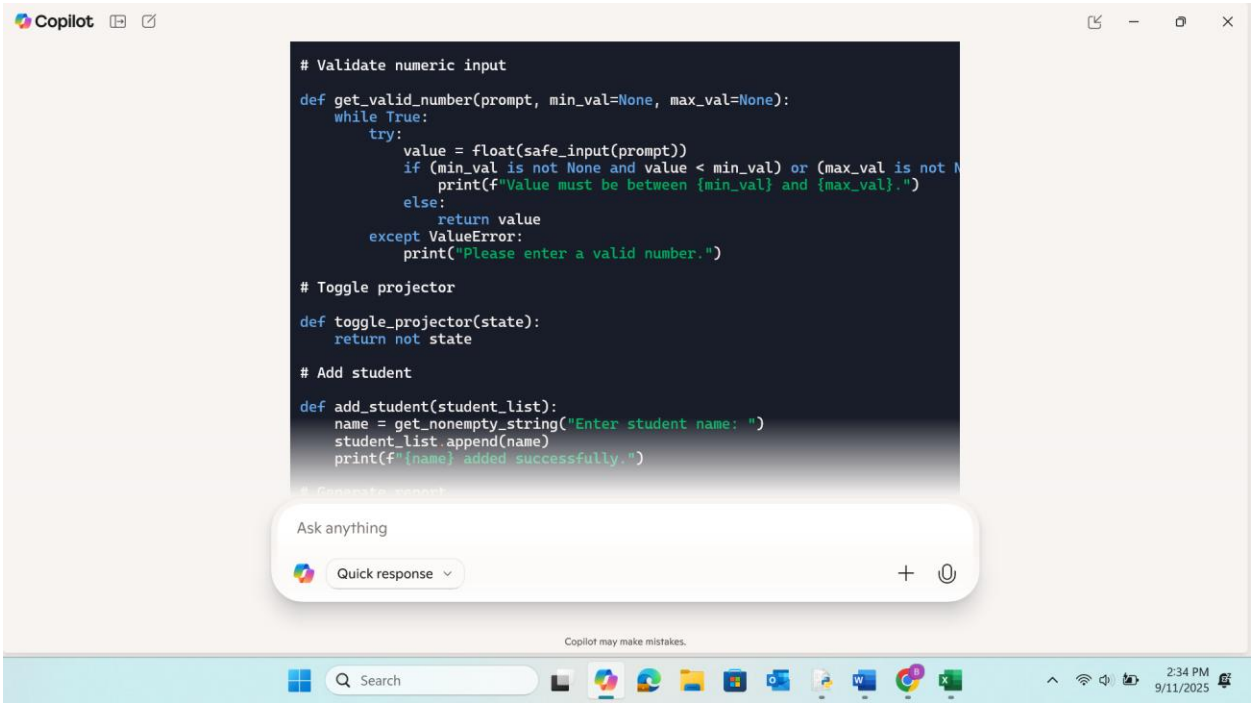
U3312671- Bazla Bilquees- LOGIC DIAG

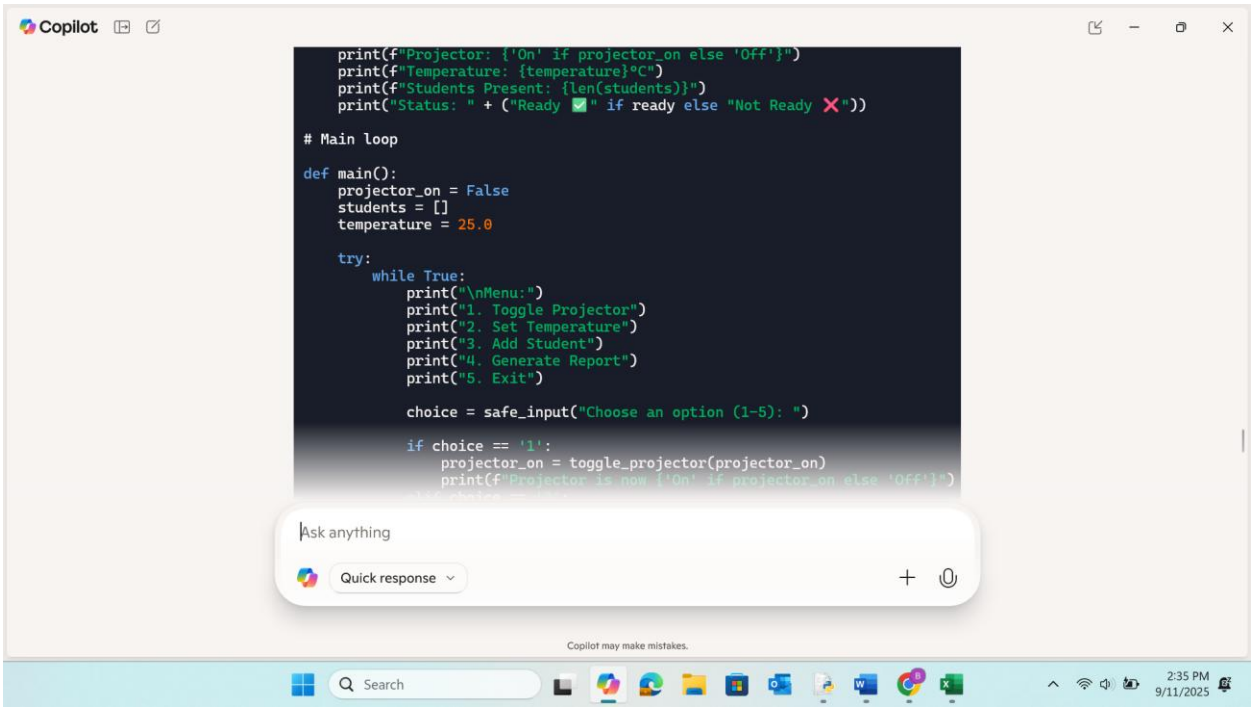
Step 9 - Refinement via GenAI: prompt(s), what changed, justification

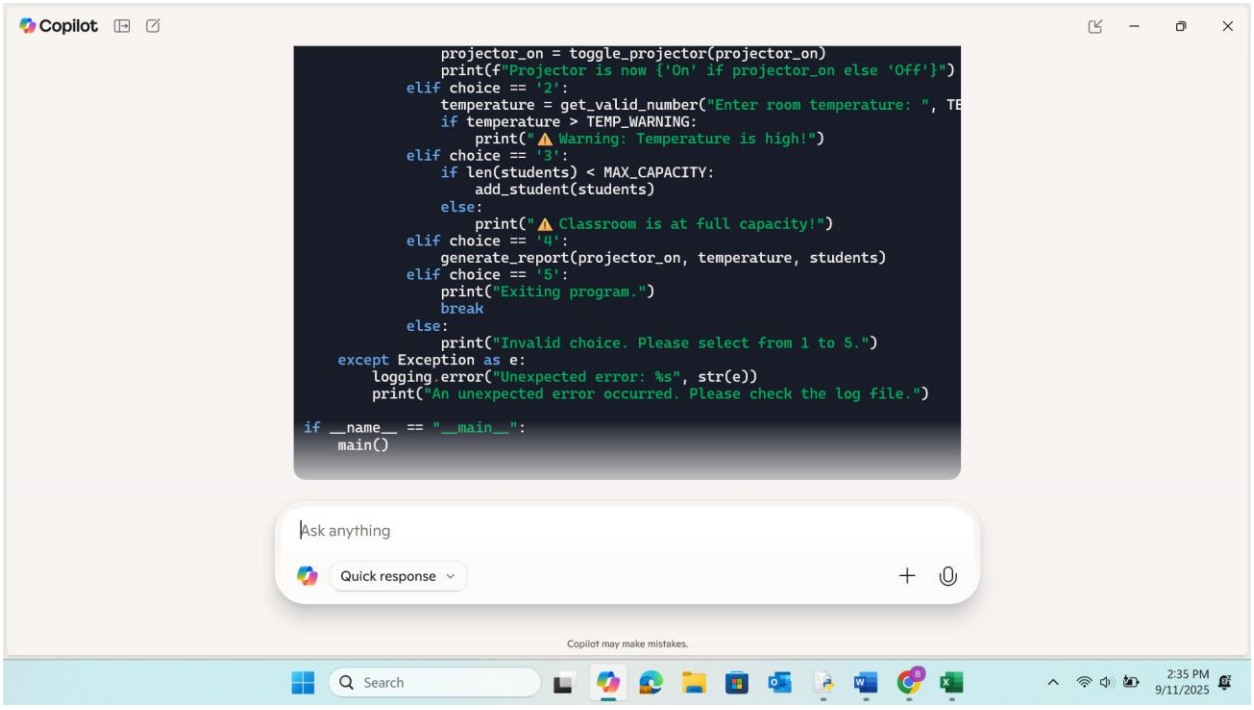
GenAI helped me to make my classroom management code clearer, structured, and strong, to refine it. I requested assistance with the logic modularisation, improved input validation, and graceful behaviour of the user interruptions. Consequently, I rearranged the code into clear functions like adding students, switching projectors and creating report to enhance readability and maintainability. To avoid unexpected exits (such as Ctrl+C or Ctrl+D), I used safe input() and I implemented checks to avoid empty or invalid entries, particularly student names and numeric inputs such as temperature and capacity. Threshold values were made constant so that crashes later may be more easily updated, and a top-level exception handler was introduced to record errors without terminating the program. These are refinements that make user interaction smoother, the data more resilient and the standard of the code more professional. The advice of GenAI has enabled me to develop a practical code into a more formal, trustworthy system that is easier to debug, expand and present with confidence in real-life scenario.

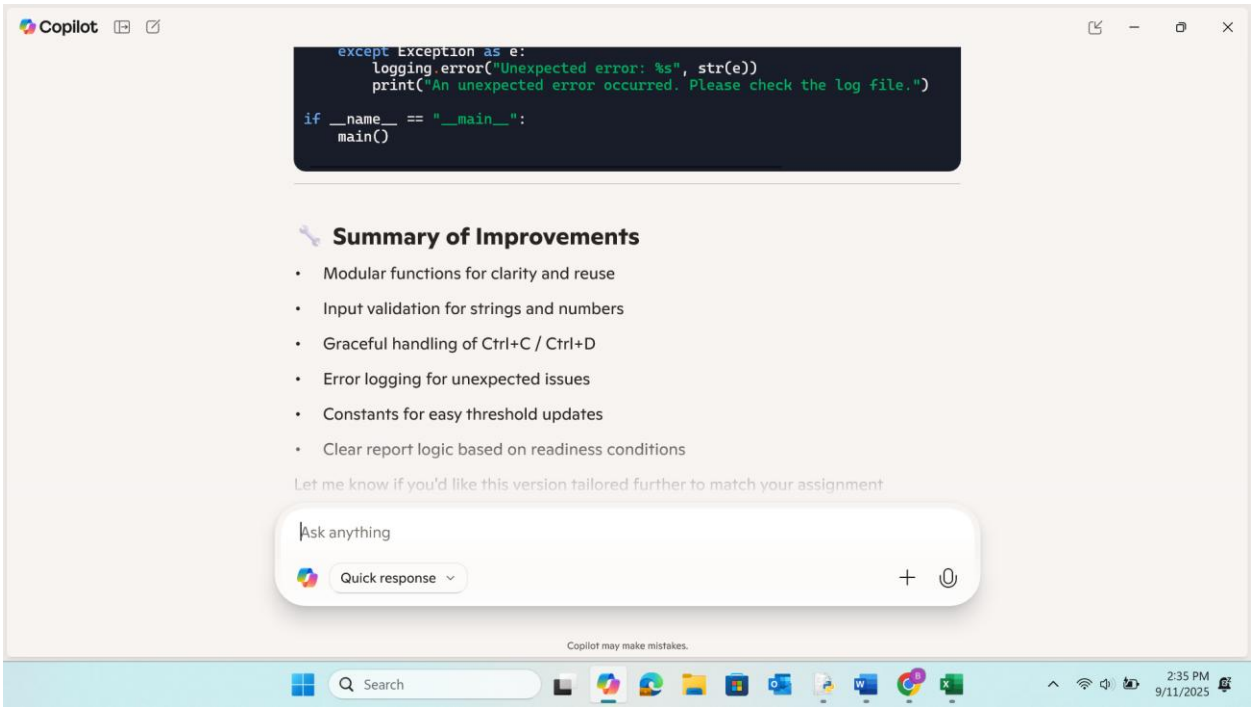



Copilot
TEMP_WARNING = 28
TEMP_MIN = 15
TEMP_MAX = 35
# Setup error logging
logging.basicConfig(filename='classroom_error.log', level=logging.ERROR)
# Safe input wrapper
def safe_input(prompt):
try:
return input(prompt)
except (KeyboardInterrupt, EOFError):
print("\nInput cancelled.")
sys.exit()
# Validate non-empty string
def get_nonempty_string(prompt):
while True:
value = safe_input(prompt).strip()
if value:
return value
print("Input cannot be empty.")
# Validate numeric input
Ask anything
Quick response
Copilot may make mistakes.
Search
2:34 PM
9/11/2025


Copilot
# Validate numeric input
def get_valid_number(prompt, min_val=None, max_val=None):
while True:
try:
value = float(safe_input(prompt))
if (min_val is not None and value < min_val) or (max_val is not N
print(f"Value must be between {min_val} and {max_val}.")
else:
return value
except ValueError:
print("Please enter a valid number.")
# Toggle projector
def toggle_projector(state):
return not state
# Add student
def add_student(student_list):
name = get_nonempty_string("Enter student name: ")
student_list.append(name)
print(f"{name} added successfully.")
# Generate report
Ask anything
Quick response
Copilot may make mistakes.
Search
2:34 PM
9/11/2025


Copilot
print(f"Projector: {'On' if projector_on else 'Off'}")
print(f"Temperature: {temperature}°C")
print(f"Students Present: {len(students)}")
print("Status: " + ("Ready ✅" if ready else "Not Ready ❌"))
# Main loop
def main():
projector_on = False
students = []
temperature = 25.0
try:
while True:
print("\nMenu:")
print("1. Toggle Projector")
print("2. Set Temperature")
print("3. Add Student")
print("4. Generate Report")
print("5. Exit")
choice = safe_input("Choose an option (1-5): ")
if choice == '1':
projector_on = toggle_projector(projector_on)
print(f"Projector is now {'On' if projector_on else 'Off'}")
Ask anything
Quick response
Copilot may make mistakes.
Search
2:35 PM
9/11/2025


Copilot
projector_on = toggle_projector(projector_on)
print(f"Projector is now {'On' if projector_on else 'Off'}")
elif choice == '2':
temperature = get_valid_number("Enter room temperature: ", TE
if temperature > TEMP_WARNING:
print("⚠ Warning: Temperature is high!")
elif choice == '3':
if len(students) < MAX_CAPACITY:
add_student(students)
else:
print("⚠ Classroom is at full capacity!")
elif choice == '4':
generate_report(projector_on, temperature, students)
elif choice == '5':
print("Exiting program.")
break
else:
print("Invalid choice. Please select from 1 to 5.")
except Exception as e:
logging.error("Unexpected error: %s", str(e))
print("An unexpected error occurred. Please check the log file.")
if __name__ == "__main__":
main()
Ask anything
Quick response
Copilot may make mistakes.
Search
2:35 PM
9/11/2025


Copilot
except Exception as e:
logging.error("Unexpected error: %s", str(e))
print("An unexpected error occurred. Please check the log file.")
if __name__ == "__main__":
main()
Summary of Improvements
Modular functions for clarity and reuse
Input validation for strings and numbers
Graceful handling of Ctrl+C / Ctrl+D
Error logging for unexpected issues
Constants for easy threshold updates
Clear report logic based on readiness conditions
Let me know if you'd like this version tailored further to match your assignment
Ask anything
Quick response
Copilot may make mistakes.
Search
2:35 PM
9/11/2025