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



## 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



A white paper with writing on it

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## 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



## 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.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.