PREPARED BY DAUD EID

AUGUST 12, 2025

# Q7

# Performance Report

SIMPLE MOVIE TICKET BOOKING SYSTEM

ARMYMAN1024.WIXSITE.COM/MY-SITE-4 252-65-9308740 ARMYMAN10249@GMAIL.COM

Alright, here's your **full written evaluation** for your Cinema_name movie ticket program.

---

**Evaluation of Your Movie Ticket Program**

**1. Variable Naming**

- ✅ Cinema_name, user_name, movies_name, ticket_prices are clear and descriptive.
- ⚠ Cinema_name should follow lowercase naming conventions (cinema_name) for Python best practices.
- ✅ You consistently used snake_case, except for the capital letter in Cinema_name.

---

**2. String Methods**

- ✅ .upper() used correctly to make output more formal and visible.
- ⚠ .lower() was used for user_choice, but your list movies_name has values in lowercase and uppercase mix — this will cause **ValueError** if user types a lowercase version, because .index() requires exact match.

---

**3. Lists Usage**

- ✅ You used **two parallel lists** (movies_name and ticket_prices).
- ⚠ Parallel lists can get tricky — if items are removed/added separately, they can go out of sync.
  Better to use a **dictionary** for future improvements (though you avoided it this time as per your request).

---

**4. Getting Movie Price**

python

CopyEdit

movie_index = movies_name.index(user_choice)

price_of_the_movie = ticket_prices[movie_index]

- ⚠ Problem: .index() will fail if the movie name typed by the user does not exactly match the case/spelling in movies_name.
- **Solution**: Either .title() or make all list items lowercase before comparing.

---

**5. Calculating Cost**

python

CopyEdit

total_cost = price_of_the_movie * user_number_of_tickets

- ✅ Correct logic, no issues.
- ✅ Correct use of multiplication with integers and floats.

---

**6. Inputs & Data Types**

- ✅ Correctly casted ticket number to int.
- ⚠ No input validation — if the user types text instead of numbers, the program will crash.

---

**7. Seat Choice**

python

CopyEdit

user_seat_choice = str(input(...)).lower()

- ✅ Good use of .lower() for consistent matching.
- ⚠ No check to ensure only "standard seat" or "vip seat" is accepted.

---

**8. Output & Formatting**

- ✅ Used f-strings for cleaner output.
- ✅ Receipt is well-structured with headers.
- ⚠ Typo: RECIEPT should be RECEIPT.

---

**9. Program Flow**

- ✅ Logical steps: Welcome → Show movies → Get choice → Calculate price → Ask seat → Show receipt.
- ⚠ If movie not found, program should give an error message and not crash.

---

**10. Best Practices Missing**

- No error handling (try/except).
- No loop for retrying on wrong input.
- No data structure to prevent movie-price mismatch.
- Could improve user experience with numbering movies instead of typing names.

---

**Final Evaluation**

| Criteria | Score (out of 10) |
| --- | --- |
| Variable Naming | 8 |
| Input Handling | 6 |
| Data Structure Usage | 7 |
| Logic & Flow | 8 |
| User Experience | 7 |
| Error Prevention | 5 |

**Overall Score: 6.8 / 10**

---

**Strengths**

- Clear structure.
- Good use of variables, lists, and f-strings.
- Simple and easy to read.

**Weaknesses**

- Lacks input validation.
- Sensitive to case/spelling errors.
- No retry loop for wrong inputs.

**Recommendation**

For your current learning level (no functions, no loops beyond while), this is a solid program.
If you want to improve:

1. Convert all movie names to lowercase before matching.
2. Add an if statement to check if the movie exists before using .index().
3. Add error handling for ticket input.
4. Replace parallel lists with a dictionary in future learning.