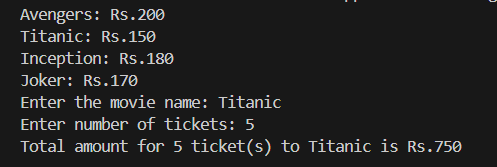
# Coding Challenge (27-06-2025)

## 1. Movie Booking System

movies = {  
 "Avengers": 200,  
 "Titanic": 150,  
 "Inception": 180,  
 "Joker": 170  
}  
  
def show\_movies():  
 for movie, price in movies.items():  
 print(f"{movie}: Rs.{price}")  
  
def calculate\_amount(movie, tickets):  
 return movies[movie] \* tickets  
  
def book\_movie():  
 show\_movies()  
 movie = input("Enter the movie name: ")  
 if movie not in movies:  
 print("Movie not available")  
 return  
 tickets = int(input("Enter number of tickets: "))  
 total = calculate\_amount(movie, tickets)  
 print(f"Total amount for {tickets} ticket(s) to {movie} is Rs.{total}")  
  
book\_movie()

Output:



## 2. Library Management

class Book:

def \_\_init\_\_(self, title, author):

self.title = title

self.author = author

class Library:

def \_\_init\_\_(self, books=[]):

self.books = books

def display\_books(self):

if not self.books:

print("\nNo books available in the library.")

else:

print("\nAvailable Books:")

for idx, book in enumerate(self.books, start=1):

print(f"{idx}. {book.title} by {book.author}")

def borrow\_book(self, index):

if 0 <= index < len(self.books):

book = self.books.pop(index)

print(f"\nYou have borrowed '{book.title}'")

else:

print("\nInvalid choice or Book not available")

def return\_book(self, book):

self.books.append(book)

print(f"\nYou have returned '{book.title}'")

class User(Library):

def \_\_init\_\_(self, books=[]):

super().\_\_init\_\_(books)

books = [

Book("Harry Potter and the Sorcerer's Stone", "J.K. Rowling"),

Book("Tom & Jerry Adventures", "Hanna-Barbera"),

Book("SpongeBob SquarePants", "Stephen Hillenburg"),

Book("Dora the Explorer", "Chris Gifford"),

Book("Pokemon Journey", "Satoshi Tajiri")

]

library = User(books)

while True:

print("\n=== Library Menu ===")

print("1. View Books")

print("2. Borrow Book")

print("3. Return Book")

print("4. Exit")

choice = input("Enter your choice (1-4): ")

if choice == '1':

library.display\_books()

elif choice == '2':

library.display\_books()

if library.books:

try:

book\_choice = int(input("Enter the book number you want to borrow: ")) - 1

library.borrow\_book(book\_choice)

except ValueError:

print("Invalid input. Please enter a valid number.")

elif choice == '3':

title = input("Enter the book title you want to return: ")

author = input("Enter the author name: ")

returned\_book = Book(title, author)

library.return\_book(returned\_book)

elif choice == '4':

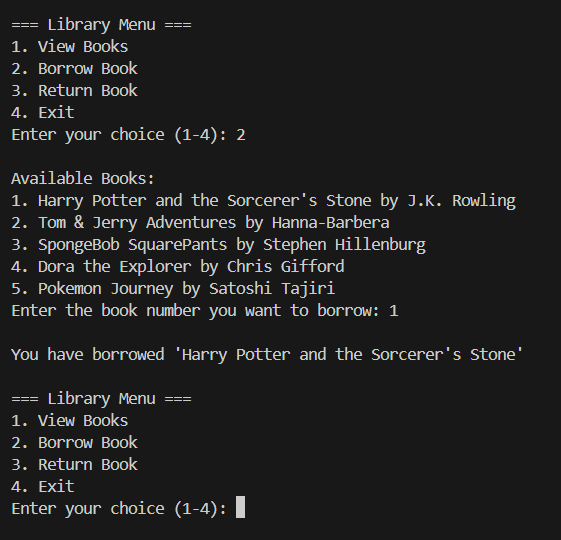
print("Thank you for visiting the library. Goodbye!")

break

else:

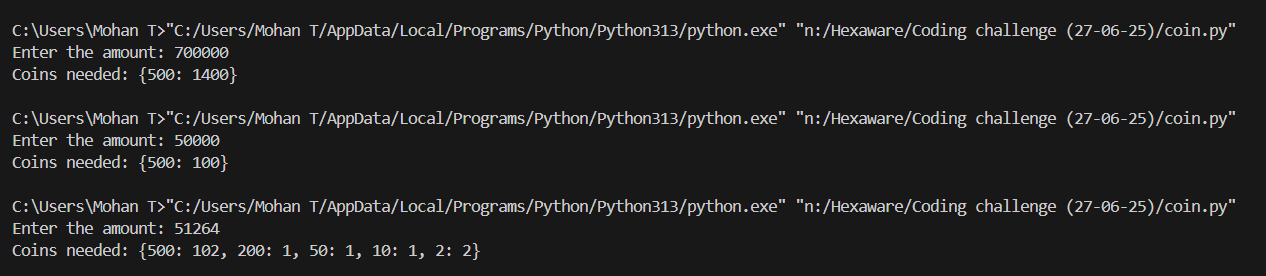
print("Invalid choice. Please select from 1 to 4.")

Output:



## 3. Minimize Coins

def minimize\_coins(amount):  
 denominations = [500, 200, 100, 50, 20, 10, 5, 2, 1]  
 result = {}  
  
 for coin in denominations:  
 count = amount // coin  
 if count > 0:  
 result[coin] = count  
 amount -= coin \* count  
  
 return result  
amount = int(input("Enter the amount: "))  
coins = minimize\_coins(amount)  
print("Coins needed:", coins)

output:

## 4. Stack - Postfix Evaluation

def evaluate\_postfix(expression):

stack = []

for char in expression:

if char.isdigit():

stack.append(int(char))

else:

b = stack.pop()

a = stack.pop()

if char == '+':

stack.append(a + b)

elif char == '-':

stack.append(a - b)

elif char == '\*':

stack.append(a \* b)

elif char == '/':

stack.append(a / b)

return stack[0]

expression = input("Enter postfix expression: ")

result = evaluate\_postfix(expression)

print("Result:", result)

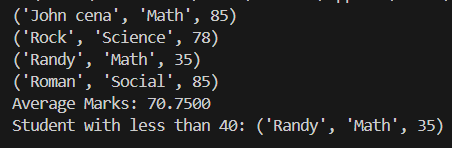
output:



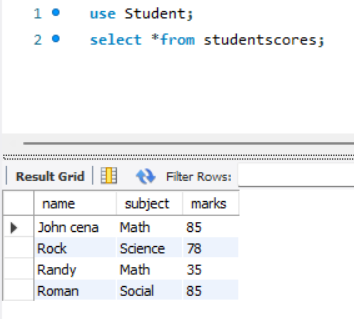
## 5. SQL with Python Integration

import mysql.connector  
  
conn = mysql.connector.connect(  
 host="localhost",  
 user="root",  
 password="Mathavan@003",  
 database="Student"  
)  
  
cursor = conn.cursor()  
  
cursor.execute("CREATE TABLE IF NOT EXISTS StudentScores (name VARCHAR(50), subject VARCHAR(50), marks INT)")  
  
cursor.execute("INSERT INTO StudentScores VALUES ('John cena', 'Math', 85), ('Rock', 'Science', 78), ('Randy', 'Math', 35), ('Roman', 'Social', 85)")  
  
conn.commit()  
  
cursor.execute("SELECT \* FROM StudentScores")  
for row in cursor.fetchall():  
 print(row)  
  
cursor.execute("SELECT AVG(marks) FROM StudentScores")  
print("Average Marks:", cursor.fetchone()[0])  
  
cursor.execute("SELECT \* FROM StudentScores WHERE marks < 40")  
for row in cursor.fetchall():  
 print("Student with less than 40:", row)  
  
conn.close()

output:



Database output:



## 6. Git Commands

git init  
git branch feature/students  
git checkout feature/students  
git add .  
git commit -m "Add student feature"  
git checkout main  
git merge feature/students

## 7. PyUnit Test for Movie Booking

import unittest  
  
movies = {"Avengers": 200, "Titanic": 150}  
  
def calculate\_amount(movie, tickets):  
 return movies[movie] \* tickets  
  
class TestMovieBooking(unittest.TestCase):  
 def setUp(self):  
 pass  
  
 def tearDown(self):  
 pass  
  
 def test\_calculate\_amount(self):  
 self.assertEqual(calculate\_amount("Avengers", 2), 400)  
  
 def test\_booking(self):  
 self.assertTrue("Avengers" in movies)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

output:

