**Coding Challenge**

Total Duration: 2 Hours

Sections:

1. Python Programming & OOP (40 mins)  
2. Data Structures & Algorithms (30 mins)  
3. SQL with Python Integration (30 mins)  
4. Version Control with Git (10 mins)  
5. Bonus/Stretch Task: Unit Testing with PyUnit (10 mins)

**Section 1: Python Programming & OOP (40 mins)**

Q1. Functional Coding Challenge – Movie Booking System (20 mins)  
- Show available movies (stored in a list)  
- Allow user to select movie & number of tickets  
- Calculate and show total amount (use a dictionary to store movie:price)  
- Use functions for showing movies, booking logic, and calculating amount

ANS: movie\_prices = {

    "Avatar": 150,

    "KGF": 120,

    "Inside out": 180,

    "Jawan": 100

}

def show\_movies():

    print("Available Movies:")

    for movie in movie\_prices:

        print(f"{movie} - Rs. {movie\_prices[movie]}")

def calculate\_amount(movie, tickets):

    return movie\_prices[movie] \* tickets

def book\_ticket():

    show\_movies()

    movie = input("\nEnter movie name: ")

    if movie not in movie\_prices:

        print(" Movie not available. Please choose from the list.")

        return

    try:

        tickets = int(input("Enter number of tickets: "))

        if tickets <= 0:

            print("Invalid number of tickets.")

            return

    except ValueError:

        print("Please enter a valid number.")

        return

    total = calculate\_amount(movie, tickets)

    print(f"Total amount for {tickets} tickets to {movie} is Rs. {total}. Thank you!")

if \_\_name\_\_ == "\_\_main\_\_":

    book\_ticket()

A computer screen with white text

AI-generated content may be incorrect.

Q2. OOP Implementation – Library Management (20 mins)  
- Create classes Book, Library, and User  
- Library contains a collection of books  
- User can borrow/return/view books  
- Use class, constructor, inheritance, method overriding

ANS:

class Book:

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

        self.title = title

        self.author = author

        self.is\_borrowed = False

    def \_\_str\_\_(self):

        status = "Not Available" if self.is\_borrowed else "Available"

        return f"{self.title} by {self.author} - {status}"

class Library:

    def \_\_init\_\_(self):

        self.books = []

    def add\_book(self, book):

        self.books.append(book)

    def view\_books(self):

        print("\nBooks in Library:")

        for book in self.books:

            print(book)

    def borrow\_book(self, title):

        for book in self.books:

            if book.title.lower() == title.lower() and not book.is\_borrowed:

                book.is\_borrowed = True

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

                return

        print("\nBook not available or already borrowed.")

    def return\_book(self, title):

        for book in self.books:

            if book.title.lower() == title.lower() and book.is\_borrowed:

                book.is\_borrowed = False

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

                return

        print("\nInvalid return or book was not borrowed.")

class User(Library):

    def \_\_init\_\_(self, name):

        super().\_\_init\_\_()

        self.name = name

    def view\_books(self):

        print(f"\n{self.name}'s View of Library:")

        super().view\_books()

# Main Program

library = Library()

library.add\_book(Book("Kids story", "Ravi"))

library.add\_book(Book("Jataka Tales", "Varma"))

library.add\_book(Book("Codind guide", "Mani"))

user1 = User("Janalyn")

user1.books = library.books

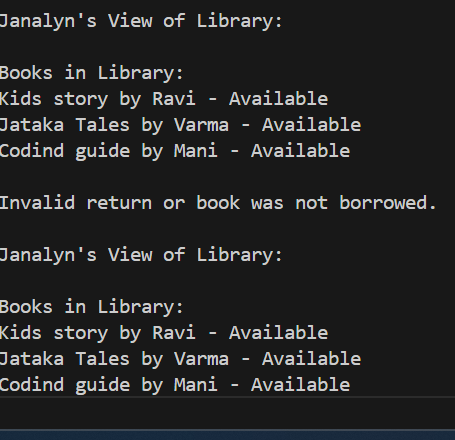
user1.view\_books()

user1.borrow\_book("Python Basics")

user1.view\_books()

user1.return\_book("Python Basics")

user1.view\_books()



**Section 2: Data Structures & Algorithms (30 mins)**

Q3. Algorithm Problem – Minimize Coins (Greedy) (15 mins)  
- Find minimum number of coins needed for a given amount  
- Denominations: [1, 2, 5, 10, 20, 50, 100, 200, 500]

ANS:

def minimize\_coins(amount):

    denominations = [500, 200, 100, 50, 20, 10, 5, 2, 1]

    result = []

    for coin in denominations:

        while amount >= coin:

            amount -= coin

            result.append(coin)

    return result

if \_\_name\_\_ == "\_\_main\_\_":

    try:

        amt = int(input("Enter the amount: "))

        if amt <= 0:

            print("Please enter a positive amount.")

        else:

            coins = minimize\_coins(amt)

            print(f"\nCoins used: {coins}")

            print(f"Total coins used: {len(coins)}")

    except ValueError:

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

A screenshot of a computer screen

AI-generated content may be incorrect.

Q4. Data Structure Usage (15 mins)  
- Stack: Evaluate postfix expression '231\*+9-'

ANS:

def evaluate\_postfix(expression):

    stack = []

    for char in expression:

        if char.isdigit():

            stack.append(int(char))

        else:

            right = stack.pop()

            left = stack.pop()

            if char == '+':

                stack.append(left + right)

            elif char == '-':

                stack.append(left - right)

            elif char == '\*':

                stack.append(left \* right)

            elif char == '/':

                stack.append(left // right)

    return stack.pop()

if \_\_name\_\_ == "\_\_main\_\_":

    expr = '231\*+9-'

    result = evaluate\_postfix(expr)

    print(f"Result of postfix '{expr}' is: {result}")

  
- Linked List class: append(), display(), reverse()

ANS:

class Node:

    def \_\_init\_\_(self, data):

        self.data = data

        self.next = None

class LinkedList:

    def \_\_init\_\_(self):

        self.head = None

    def append(self, data):

        new\_node = Node(data)

        if not self.head:

            self.head = new\_node

        else:

            curr = self.head

            while curr.next:

                curr = curr.next

            curr.next = new\_node

    def display(self):

        curr = self.head

        print("\nLinked List:", end=" ")

        while curr:

            print(curr.data, end=" -> ")

            curr = curr.next

        print("None")

    def reverse(self):

        prev = None

        curr = self.head

        while curr:

            next\_node = curr.next

            curr.next = prev

            prev = curr

            curr = next\_node

        self.head = prev

if \_\_name\_\_ == "\_\_main\_\_":

    ll = LinkedList()

    ll.append(10)

    ll.append(20)

    ll.append(30)

    ll.display()

    ll.reverse()

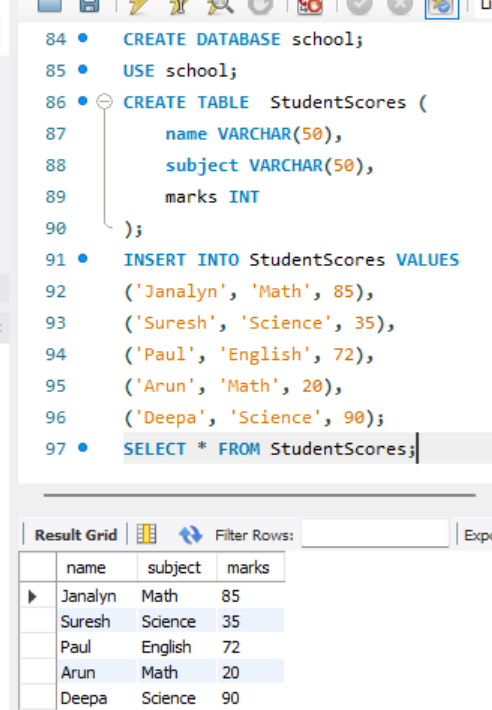
    ll.display()

A black background with white text

AI-generated content may be incorrect.

**Section 3: SQL with Python Integration (30 mins)**

Q5. SQL + Python – Student Scores Table  
- Create table StudentScores(name VARCHAR, subject VARCHAR, marks INT)  
- Insert sample data

  
- Use Python to display records, show average marks, list students scoring <40

ANS: import mysql.connector

def connect\_db():

    return mysql.connector.connect(

        host="localhost",

        user="root",

        password="janalyn@6205",

        database="school"

    )

def show\_all\_scores(cursor):

    cursor.execute("SELECT \* FROM StudentScores")

    print("\n All Student Records:")

    for row in cursor.fetchall():

        print(row)

def show\_average(cursor):

    cursor.execute("SELECT AVG(marks) FROM StudentScores")

    avg = cursor.fetchone()[0]

    print(f"\nAverage Marks: {avg:.2f}")

def show\_failed\_students(cursor):

    cursor.execute("SELECT name, subject, marks FROM StudentScores WHERE marks < 40")

    print("\n Students Scoring Below 40:")

    results = cursor.fetchall()

    if results:

        for row in results:

            print(row)

    else:

        print("None")

if \_\_name\_\_ == "\_\_main\_\_":

    try:

        conn = connect\_db()

        cursor = conn.cursor()

        show\_all\_scores(cursor)

        show\_average(cursor)

        show\_failed\_students(cursor)

        cursor.close()

        conn.close()

    except mysql.connector.Error as err:

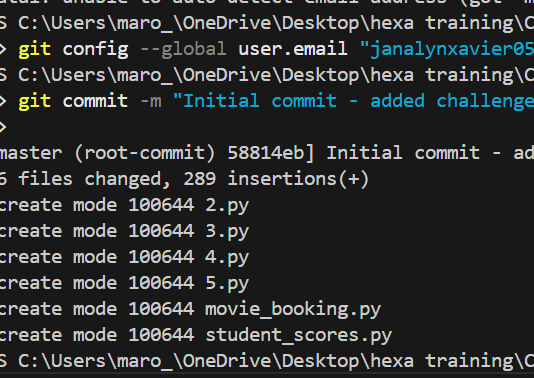
        print("Error:", err)

A screenshot of a computer

AI-generated content may be incorrect.

**Section 4: Version Control with Git (10 mins)**

Q6. Git Challenge  
- Initialize Git repository  
- Create and switch to branch feature/students  
- Add and commit your Python code  
- Merge feature/students into main  
- Provide Git commands



**Bonus Section: PyUnit Test Case (10 mins)**

Q7. PyUnit test cases for Q1 (Booking System)  
- 1 test case for calculate\_amount()  
- 1 test case for booking() using mocks if needed  
- Use unittest.TestCase, setUp(), tearDown()

# test\_movie\_booking.py

import unittest

from movie\_booking import calculate\_amount, book\_movie

class TestMovieBooking(unittest.TestCase):

    def test\_calculate\_amount(self):

        self.assertEqual(calculate\_amount("KGF", 2), 240)

        self.assertEqual(calculate\_amount("Jawan", 1), 100)

    def test\_book\_movie\_valid(self):

        result = book\_movie("Inside out", 2)

        self.assertIn("Booked 2 ticket(s) for Inside out", result)

    def test\_book\_movie\_invalid\_name(self):

        result = book\_movie("Barbie", 3)

        self.assertEqual(result, "Movie not available.")

    def test\_book\_movie\_invalid\_tickets(self):

        result = book\_movie("KGF", 0)

        self.assertEqual(result, "Invalid number of tickets.")

if \_\_name\_\_ == "\_\_main\_\_":

    unittest.main()

A screen shot of a computer

AI-generated content may be incorrect.