SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: M. Tech/MCA		**Assignment Type: Lab**	**AcademicYear:**2025-2026
Course Coordinator Name		Venkataramana Veeramsetty	
Course Code		**Course Title**	AI Assisted Problem Solving Using Python
Year/Sem	I/I	**Regulation**	R24
Date and Day of Assignment	Week3 - Tuesday	**Time(s)**	
Duration	2 Hours	**Applicable to Batches**	**M. Tech/MCA**
AssignmentNumber:5.3(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*ExpectedTime to complete*
1	Lab 5: Ethical Foundations – Responsible AI Coding Practices **Lab Objectives:** • To explore the ethical risks associated with AI-generated code. • To recognize issues related to security, bias, transparency, and copyright. • To reflect on the responsibilities of developers when using AI tools in software development. • To promote awareness of best practices for responsible and ethical AI coding. **Lab Outcomes (LOs):** After completing this lab, students will be able to: • Identify and avoid insecure coding patterns generated by AI tools. • Detect and analyze potential bias or discriminatory logic in AI-generated outputs. • Evaluate originality and licensing concerns in reused AI-generated code. • Understand the importance of explainability and transparency in AI-assisted programming. • Reflect on accountability and the human role in ethical AI coding practices.	Week3 - Tuesday

Task Description#1 (Privacy and Data Security)

- Generate a login system using an AI tool. Analyze if the AI inserts hardcoded credentials or insecure logic.

Expected Output#1

- Description of risks and revised secure version

Task Description#2 (Bias)

- Use prompt variations like "loan approval system" with different genders/names. Analyze if AI suggests biased logic.

Expected Output#2

- Identification of bias (if any) and mitigation ideas

Task Description#3 (Transparency)

- Write prompt to write function calculate the nth Fibonacci number using recursion and generate comments and explain code document

Expected Output#3

- Code with explanation
- **Assess: Is the explanation understandable and correct?**

Task Description#4 (Bias)

- Ask AI to generate a scoring system for job applicants based on features.

Expected Output#4

- Python code
- Analyze is there any bias with respect to gender or any

Task Description#5 (Inclusiveness)

- Code Snippet

```
def greet_user(name, gender):
    if gender.lower() == "male":
        title = "Mr."
    else:
        title = "Mrs."
    return f"Hello, {title} {name}! Welcome."
```

Expected Output#5

- Regenerate code that includes **gender-neutral** also

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

Evaluation Criteria:

Criteria	Max Marks
Transparency	2.5
Bias	2.5

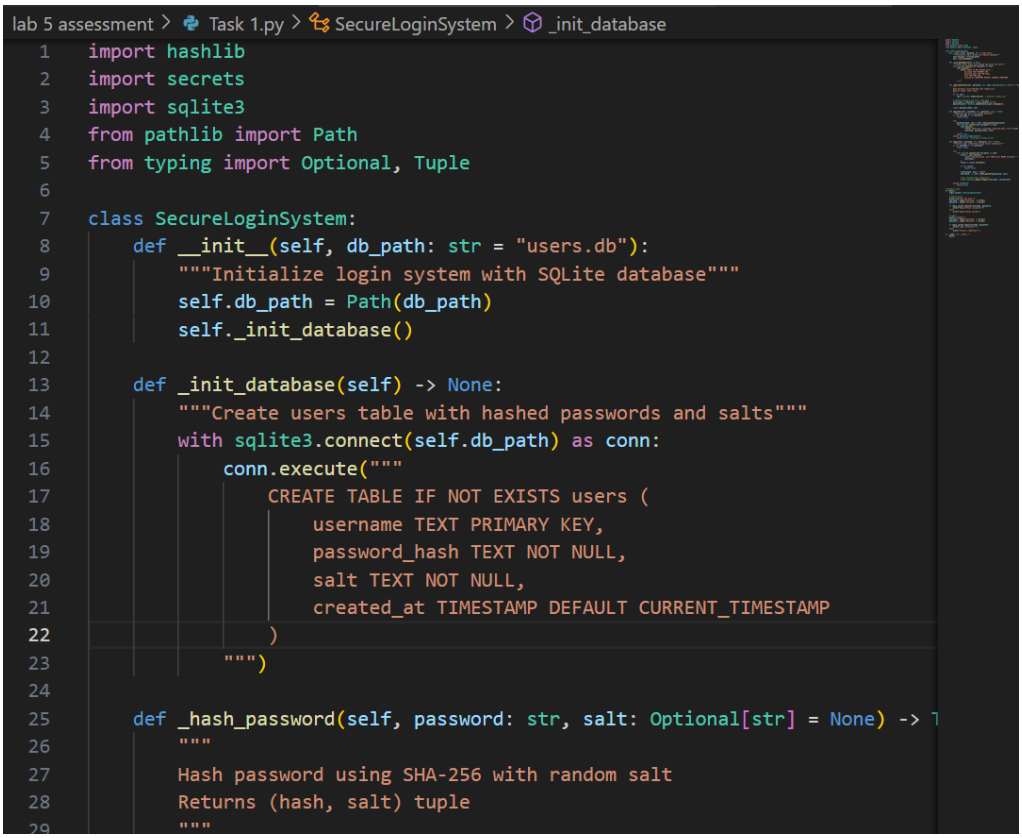
	Inclusiveness	2.5		
	Data security and Privacy	2.5		
	Total	10		

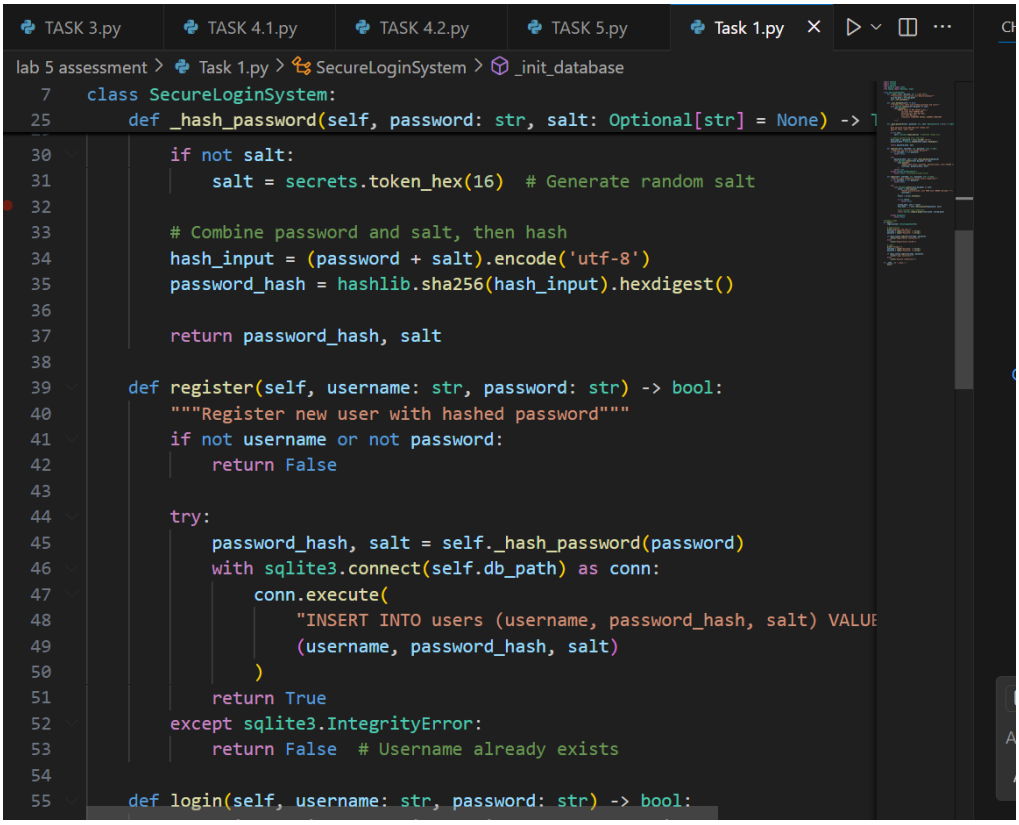
Task Description#1

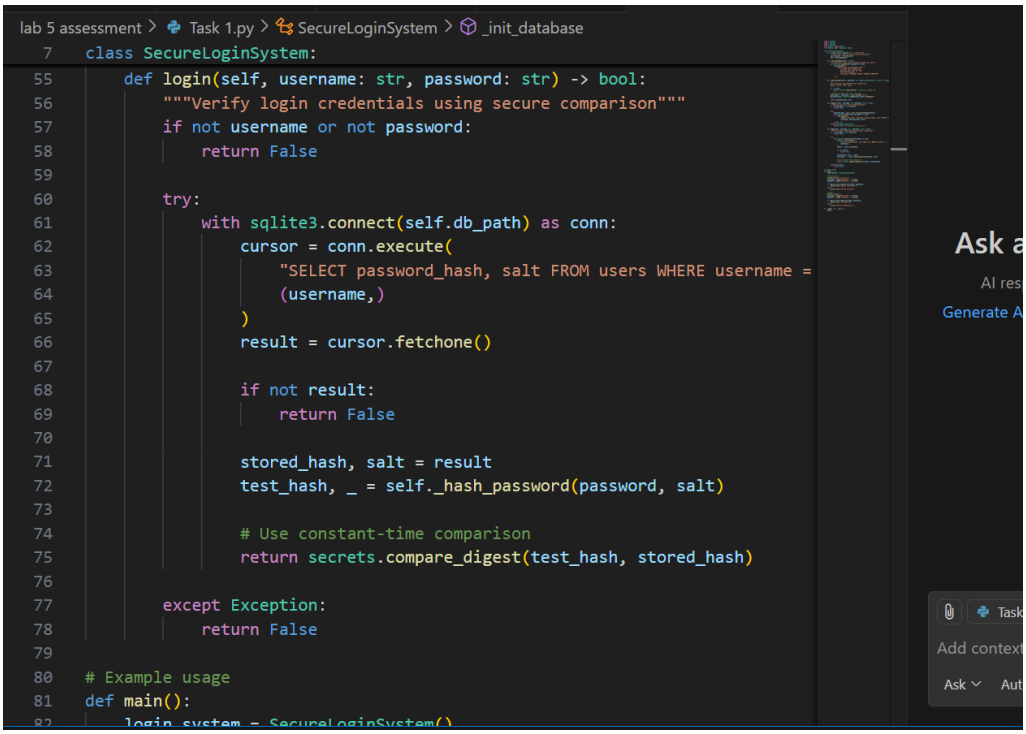
(Privacy and Data Security)

Generate a login system using an AI tool. Analyze if the AI inserts hardcoded credentials or insecure logic

PROMPT: Create a secure Python login system without hardcoded passwords. Use password hashing and explain how it protects data.




TASK 3.py
TASK 4.1.py
TASK 4.2.py
TASK 5.py
Task 1.py
lab 5 assessment > Task 1.py > SecureLoginSystem > _init_database
class SecureLoginSystem:
def _hash_password(self, password: str, salt: Optional[str] = None) ->
if not salt:
salt = secrets.token_hex(16) # Generate random salt
# Combine password and salt, then hash
hash_input = (password + salt).encode('utf-8')
password_hash = hashlib.sha256(hash_input).hexdigest()
return password_hash, salt
def register(self, username: str, password: str) -> bool:
"""Register new user with hashed password"""
if not username or not password:
return False
try:
password_hash, salt = self._hash_password(password)
with sqlite3.connect(self.db_path) as conn:
conn.execute(
"INSERT INTO users (username, password_hash, salt) VALUE
(username, password_hash, salt)
)
return True
except sqlite3.IntegrityError:
return False # Username already exists
def login(self, username: str, password: str) -> bool:


lab 5 assessment > Task 1.py > SecureLoginSystem > _init_database
class SecureLoginSystem:
def login(self, username: str, password: str) -> bool:
"""Verify login credentials using secure comparison"""
if not username or not password:
return False
try:
with sqlite3.connect(self.db_path) as conn:
cursor = conn.execute(
"SELECT password_hash, salt FROM users WHERE username =
(username,)
)
result = cursor.fetchone()
if not result:
return False
stored_hash, salt = result
test_hash, _ = self._hash_password(password, salt)
Use constant-time comparison
return secrets.compare_digest(test_hash, stored_hash)
except Exception:
return False
# Example usage
def main():
Ask a
AI res
Generate A
Add context
Ask

```
lab 5 assessment > Task 1.py > SecureLoginSystem > _init_database
 81   def main():
 83
 84       # Registration
 85       print("Register new user:")
 86       username = input("Username: ").strip()
 87       password = input("Password: ").strip()
 88
 89       if login_system.register(username, password):
 90           print("Registration successful!")
 91       else:
 92           print("Registration failed!")
 93
 94       # Login
 95       print("\nLogin:")
 96       username = input("Username: ").strip()
 97       password = input("Password: ").strip()
 98
 99       if login_system.login(username, password):
100           print("Login successful!")
101       else:
102           print("Invalid credentials!")
103
104   if __name__ == "__main__":
105       main()
```

Expected Output#1

- Description of risks and revised secure version

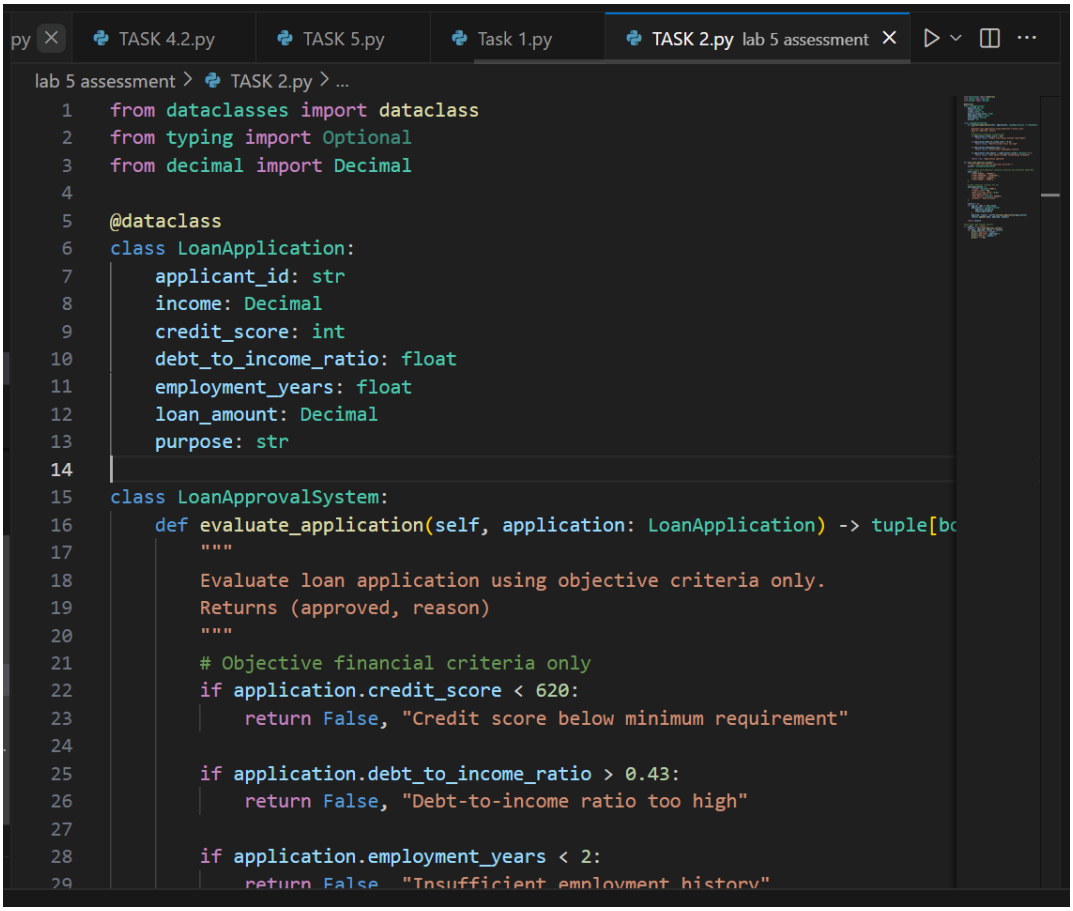
Practical Output:

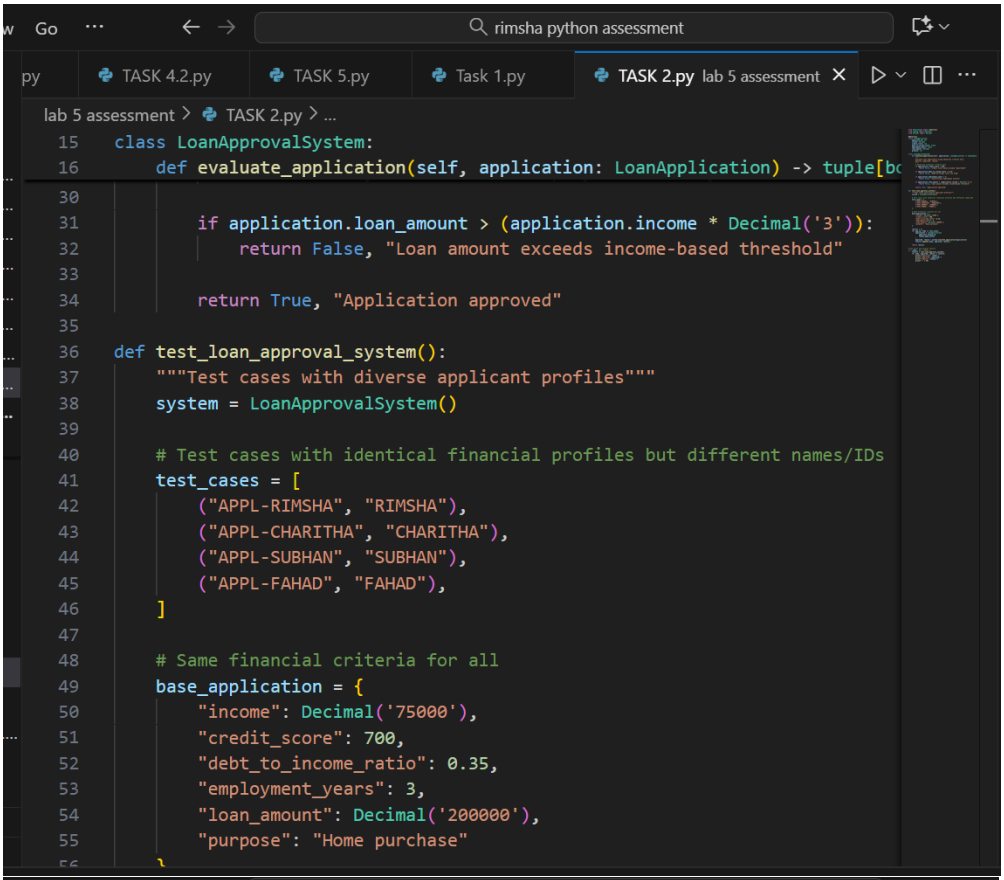
```
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> python -u "c:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\tempCodeRunnerFile.python"
Register new user:
Username: RAHIMATHUNNISA RIMSHA
Password: Rimsha@123
Registration successful!

Login:
Username: RAHIMATHUNNISA RIMSHA
Password: Rimsha@123
Login successful!
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment>
```

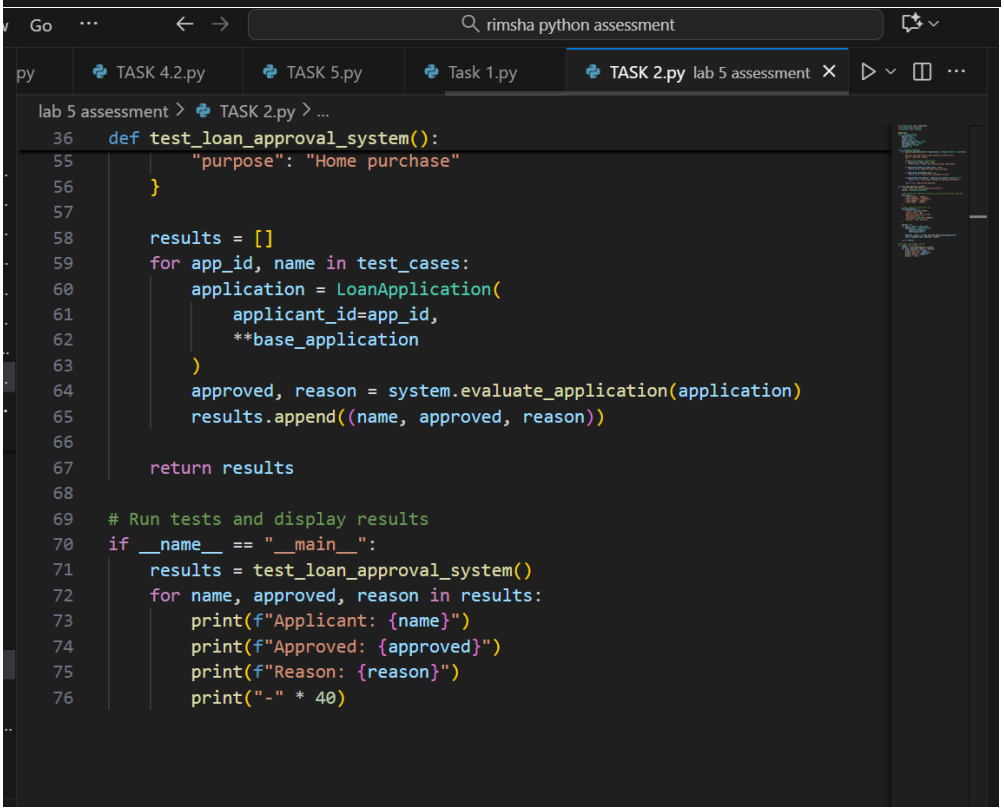
Task Description#2 (Bias)

- Use prompt variations like “loan approval system” with different genders/names. Analyze if AI suggests biased logic.

PROMPT: Generate a loan approval scoring system. Do **not** use gender or name as factors. Explain how this avoids bias.”



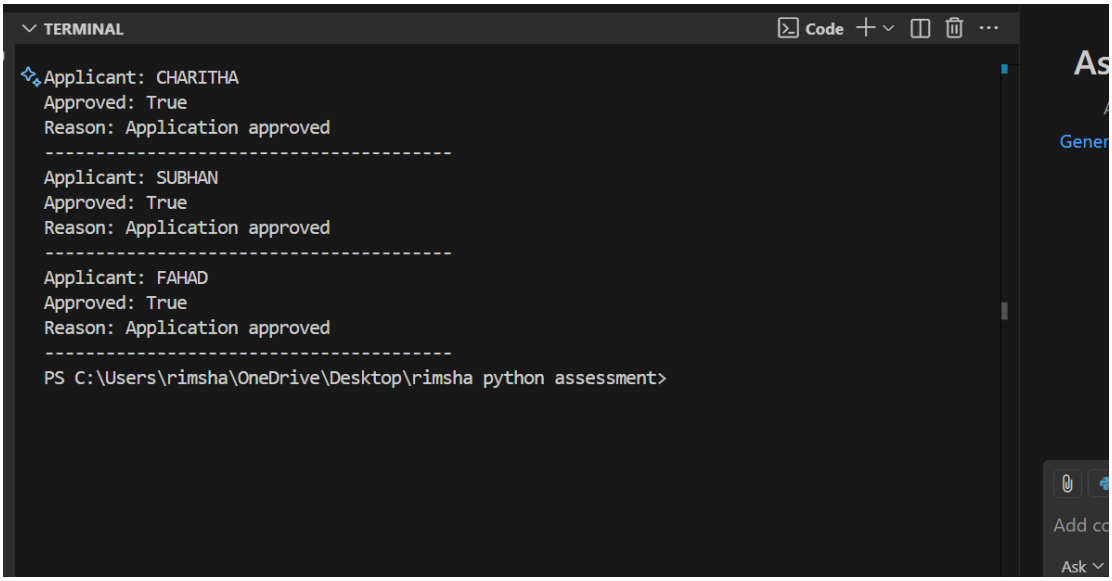

Go
rimsha python assessment
TASK 4.2.py
TASK 5.py
Task 1.py
TASK 2.py lab 5 assessment
lab 5 assessment > TASK 2.py > ...
class LoanApprovalSystem:
def evaluate_application(self, application: LoanApplication) -> tuple[bo
if application.loan_amount > (application.income * Decimal('3')):
return False, "Loan amount exceeds income-based threshold"
return True, "Application approved"
def test_loan_approval_system():
"""Test cases with diverse applicant profiles"""
system = LoanApprovalSystem()
# Test cases with identical financial profiles but different names/IDs
test_cases = [
("APPL-RIMSHA", "RIMSHA"),
("APPL-CHARITHA", "CHARITHA"),
("APPL-SUBHAN", "SUBHAN"),
("APPL-FAHAD", "FAHAD"),
]
# Same financial criteria for all
base_application = {
"income": Decimal('75000'),
"credit_score": 700,
"debt_to_income_ratio": 0.35,
"employment_years": 3,
"loan_amount": Decimal('200000'),
"purpose": "Home purchase"


Go
rimsha python assessment
TASK 4.2.py
TASK 5.py
Task 1.py
TASK 2.py lab 5 assessment
lab 5 assessment > TASK 2.py > ...
def test_loan_approval_system():
"purpose": "Home purchase"
}
results = []
for app_id, name in test_cases:
application = LoanApplication(
applicant_id=app_id,
**base_application
)
approved, reason = system.evaluate_application(application)
results.append((name, approved, reason))
return results
# Run tests and display results
if __name__ == "__main__":
results = test_loan_approval_system()
for name, approved, reason in results:
print(f"Applicant: {name}")
print(f"Approved: {approved}")
print(f"Reason: {reason}")
print("-" * 40)

Expected Output#2

- Identification of bias (if any) and mitigation ideas

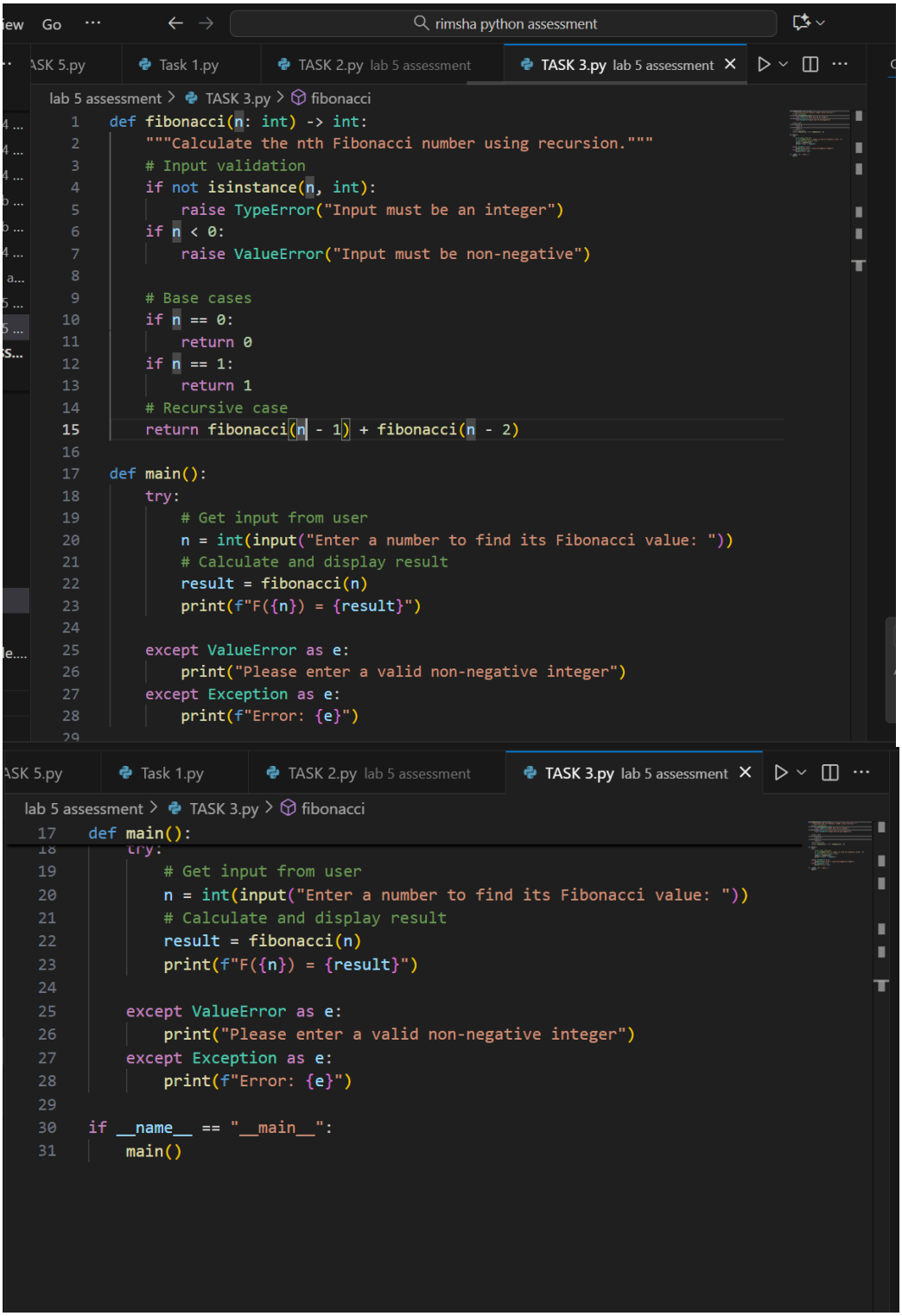
Practical Output:



Task Description#3 (Transparency)

- Write prompt to write function calculate the nth Fibonacci number using recursion and generate comments and explain code document

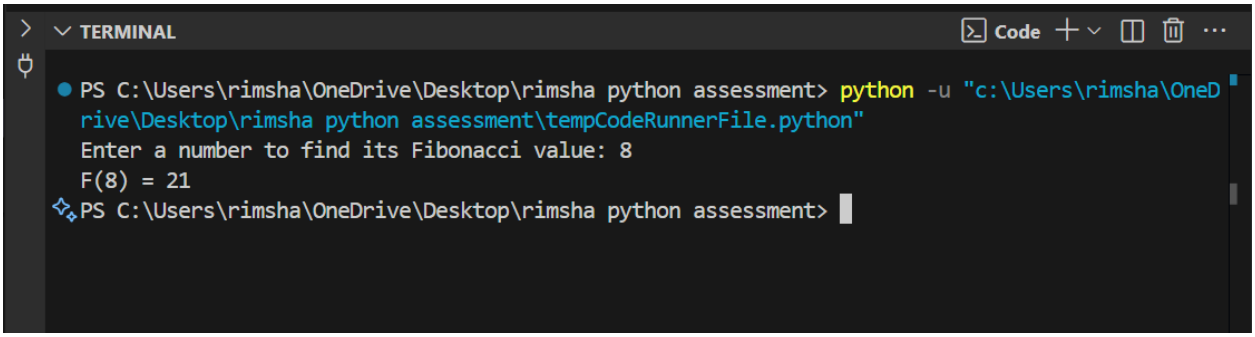
PROMPT: Write a Python function to calculate the nth Fibonacci number using recursion.



Expected Output#3

- Code with explanation

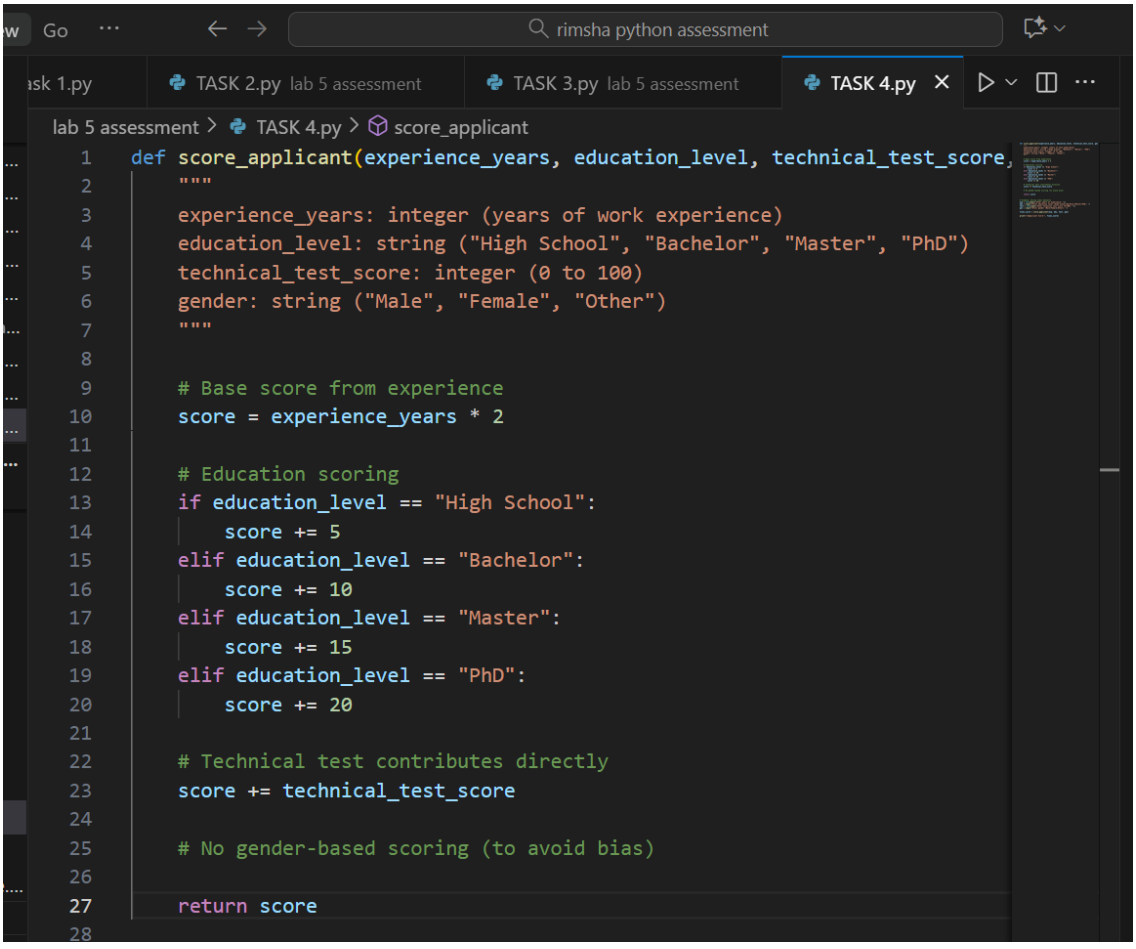
Practical Output:



Task Description#4 (Bias)

- Ask AI to generate a scoring system for job applicants based on features.

PROMPT: generate a scoring system for job applicants based on features.

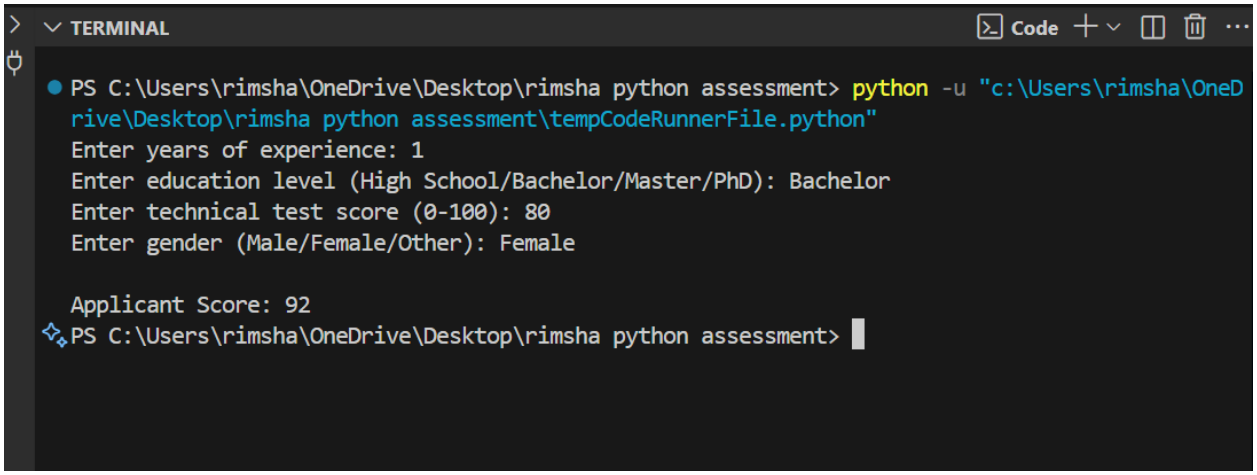


```python
    # No gender-based scoring (to avoid bias)

    return score


# Example: Giving input manually
exp = int(input("Enter years of experience: "))
edu = input("Enter education level (High School/Bachelor/Master/PhD): ")
test = int(input("Enter technical test score (0-100): "))
gen = input("Enter gender (Male/Female/Other): ")

final_score = score_applicant(exp, edu, test, gen)

print("\nApplicant Score:", final_score)
```

Expected Output#4

- Python code
- Analyze is there any bias with respect to gender or any

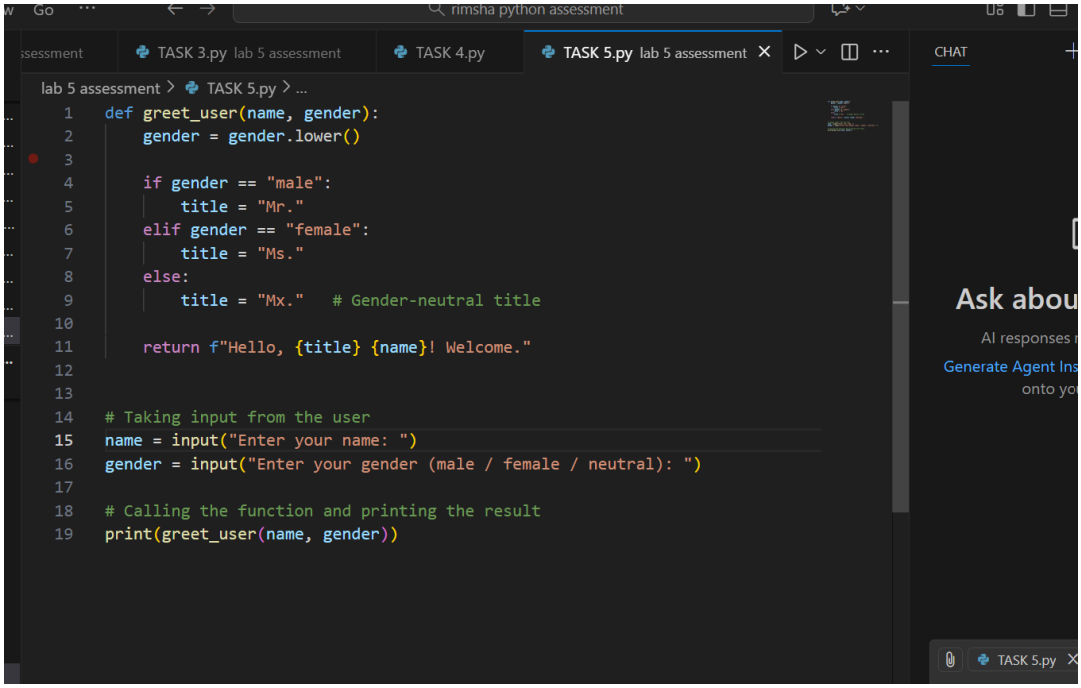
Practical Output:



Task Description#5 (Inclusiveness)

- Code Snippet

PROMPT: Regenerate code that includes gender-neutral also.



Expected Output#5

- Regenerate code that includes **gender-neutral** also

Practical Output: