

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	<p>Lab 5: Ethical Foundations – Responsible AI Coding Practices</p> <p>Lab Objectives:</p> <ul style="list-style-type: none"> 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. <p>Lab Outcomes (LOs): After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> 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 4.py TASK 5.py TASK1.py X def count_lines_in_file(file_path): Untitled ... CHA
C: > Users > rimsha > OneDrive > Desktop > Mohammed Farnas Ali Mudabbir > LAB 5 > TASK1.py > ...
1  import hashlib
2  import secrets
3  import sqlite3
4  from pathlib import Path
5  from typing import Optional, Tuple
6
7  class SecureLoginSystem:
8      def __init__(self, db_path: str = "users.db"):
9          """Initialize login system with SQLite database"""
10         self.db_path = Path(db_path)
11         self._init_database()
12
13     def _init_database(self) -> None:
14         """Create users table with hashed passwords and salts"""
15         with sqlite3.connect(self.db_path) as conn:
16             conn.execute("""
17                 CREATE TABLE IF NOT EXISTS users (
18                     username TEXT PRIMARY KEY,
19                     password_hash TEXT NOT NULL,
20                     salt TEXT NOT NULL,
21                     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
22                 )
23             """)
24
25     def _hash_password(self, password: str, salt: Optional[str] = None)
26     """
27     Hash password using SHA-256 with random salt
28     Returns (hash, salt) tuple
29     """

```

```
TASK 4.py TASK 5.py TASK1.py X def count_lines_in_file(file_path): Untitled CHAT
C: > Users > rimsha > OneDrive > Desktop > Mohammed Farnas Ali Mudabbir > LAB 5 > TASK1.py > ...
7 class SecureLoginSystem:
25 def _hash_password(self, password: str, salt: Optional[str] = None)
27     """Hash password using SHA-256 with random salt
28     Returns (hash, salt) tuple
29     """
30     if not salt:
31         salt = secrets.token_hex(16) # Generate random salt
32
33     # Combine password and salt, then hash
34     hash_input = (password + salt).encode('utf-8')
35     password_hash = hashlib.sha256(hash_input).hexdigest()
36
37     return password_hash, salt
38
39 def register(self, username: str, password: str) -> bool:
40     """Register new user with hashed password"""
41     if not username or not password:
42         return False
43
44     try:
45         password_hash, salt = self._hash_password(password)
46         with sqlite3.connect(self.db_path) as conn:
47             conn.execute(
48                 "INSERT INTO users (username, password_hash, salt) VALUES
49                 (username, password_hash, salt)
50             )
51         return True
52     except sqlite3.IntegrityError:
53         return False # Username already exists
```

```
TASK 4.py TASK 5.py TASK1.py X def count_lines_in_file(file_path): Untitled CHAT
C: > Users > rimsha > OneDrive > Desktop > Mohammed Farnas Ali Mudabbir > LAB 5 > TASK1.py > ...
53         return False # Username already exists
54
55 def login(self, username: str, password: str) -> bool:
56     """Verify login credentials using secure comparison"""
57     if not username or not password:
58         return False
59
60     try:
61         with sqlite3.connect(self.db_path) as conn:
62             cursor = conn.execute(
63                 "SELECT password_hash, salt FROM users WHERE username
64                 (username,)
65             )
66             result = cursor.fetchone()
67
68             if not result:
69                 return False
70
71             stored_hash, salt = result
72             test_hash, _ = self._hash_password(password, salt)
73
74             # Use constant-time comparison
75             return secrets.compare_digest(test_hash, stored_hash)
76
77     except Exception:
78         return False
79
80 # Example usage
```

```
TASK 4.py TASK 5.py TASK1.py X def count_lines_in_file(file_path): Untitled
C: > Users > rimsha > OneDrive > Desktop > Mohammed Farnas Ali Mudabbir > LAB 5 > TASK1.py > ...
/9
80 # Example usage
81 def main():
82     login_system = SecureLoginSystem()
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:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
> v TERMINAL
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1> python -u "
c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1\tempCodeRunnerF
ile.python"
• Register new user:
Username: FARNAS7
Password: FARNAS@123
Registration successful!

Login:
Username: FARNAS7
Password: FARNAS@123
Login successful!
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1>

Create a secure P
hardcoded passw
and explain how

> Used 1 reference

Here's a secure login
password hashing a
explanations:

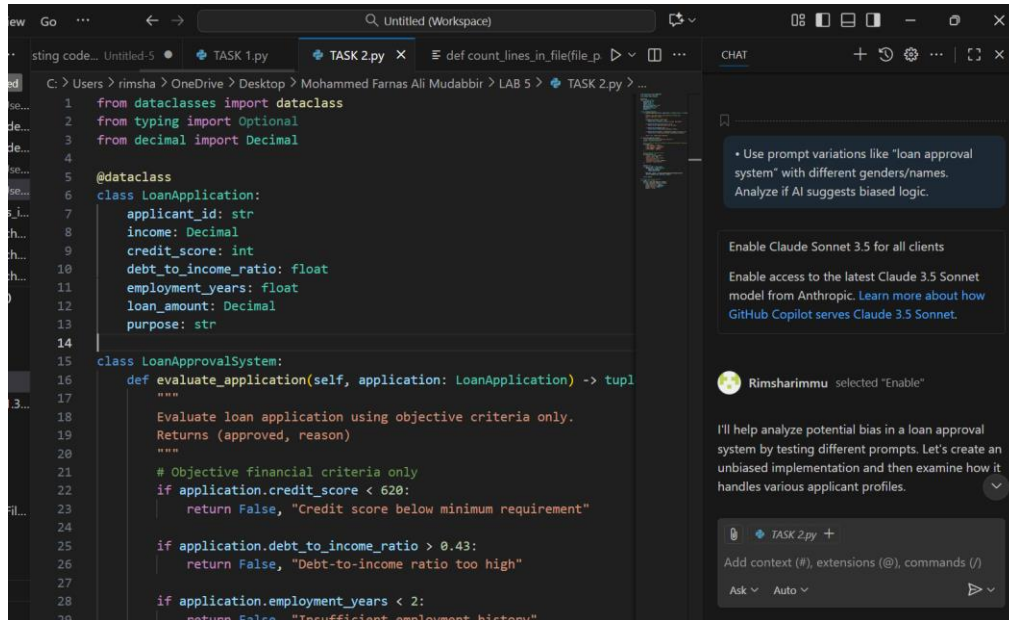
import hashlib
import secrets
import sqlite3
from nathlih i

import hashl
```

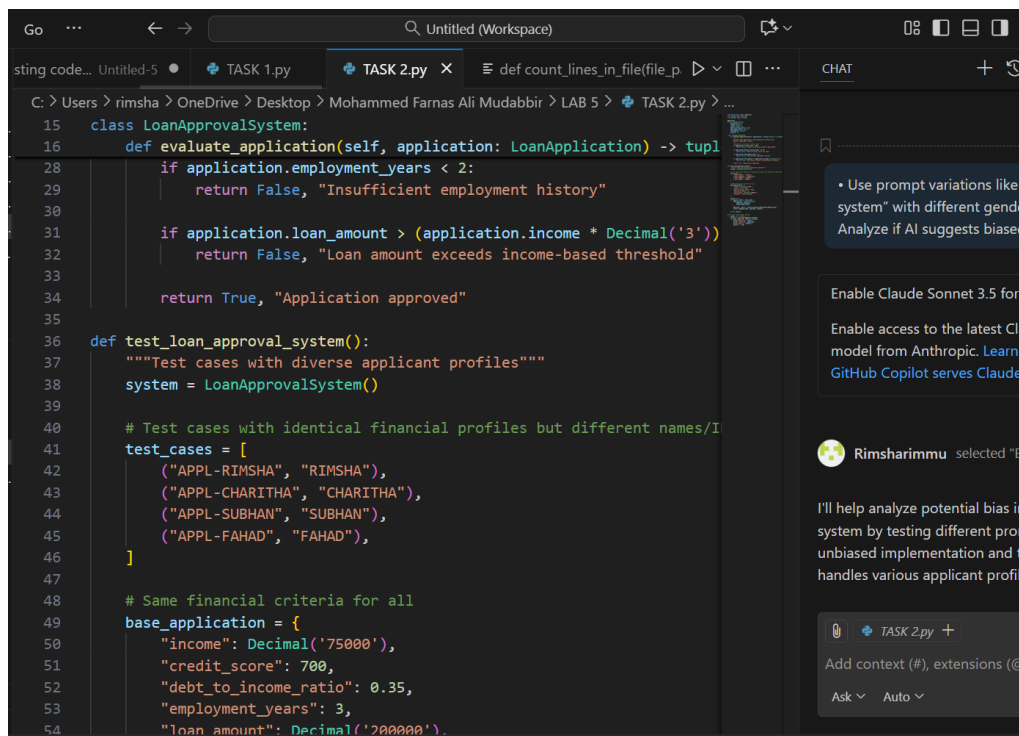
Task Description#2 (Bias)

- 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.”



```
1 from dataclasses import dataclass
2 from typing import Optional
3 from decimal import Decimal
4
5 @dataclass
6 class LoanApplication:
7     applicant_id: str
8     income: Decimal
9     credit_score: int
10    debt_to_income_ratio: float
11    employment_years: float
12    loan_amount: Decimal
13    purpose: str
14
15 class LoanApprovalSystem:
16     def evaluate_application(self, application: LoanApplication) -> tuple:
17         """
18         Evaluate loan application using objective criteria only.
19         Returns (approved, reason)
20         """
21         # Objective financial criteria only
22         if application.credit_score < 620:
23             return False, "Credit score below minimum requirement"
24
25         if application.debt_to_income_ratio > 0.43:
26             return False, "Debt-to-income ratio too high"
27
28         if application.employment_years < 2:
29             return False, "Insufficient employment history"
```



```
15 class LoanApprovalSystem:
16     def evaluate_application(self, application: LoanApplication) -> tuple:
17         """
18         Evaluate loan application using objective criteria only.
19         Returns (approved, reason)
20         """
21         # Objective financial criteria only
22         if application.credit_score < 620:
23             return False, "Credit score below minimum requirement"
24
25         if application.debt_to_income_ratio > 0.43:
26             return False, "Debt-to-income ratio too high"
27
28         if application.employment_years < 2:
29             return False, "Insufficient employment history"
30
31         if application.loan_amount > (application.income * Decimal('3')):
32             return False, "Loan amount exceeds income-based threshold"
33
34         return True, "Application approved"
35
36 def test_loan_approval_system():
37     """Test cases with diverse applicant profiles"""
38     system = LoanApprovalSystem()
39
40     # Test cases with identical financial profiles but different names/IDs
41     test_cases = [
42         ("APPL-RIMSHA", "RIMSHA"),
43         ("APPL-CHARITHA", "CHARITHA"),
44         ("APPL-SUBHAN", "SUBHAN"),
45         ("APPL-FAHAD", "FAHAD"),
46     ]
47
48     # Same financial criteria for all
49     base_application = {
50         "income": Decimal('75000'),
51         "credit_score": 700,
52         "debt_to_income_ratio": 0.35,
53         "employment_years": 3,
54         "loan_amount": Decimal('200000'),
```

```
sting code... Untitled-5 • TASK 1.py TASK 2.py X def count_lines_in_file(file_p
C: > Users > rimsha > OneDrive > Desktop > Mohammed Farnas Ali Mudabbir > LAB 5 > TASK 2.py > ...
36 def test_loan_approval_system():
53     "employment_years": 3,
54     "loan_amount": Decimal('200000'),
55     "purpose": "Home purchase"
56 }
57
58 results = []
59 for app_id, name in test_cases:
60     application = LoanApplication(
61         applicant_id=app_id,
62         **base_application
63     )
64     approved, reason = system.evaluate_application(application)
65     results.append((name, approved, reason))
66
67     return results
68
69 # Run tests and display results
70 if __name__ == "__main__":
71     results = test_loan_approval_system()
72     for name, approved, reason in results:
73         print(f"Applicant: {name}")
74         print(f"Approved: {approved}")
75         print(f"Reason: {reason}")
76         print("-" * 40)
```

Expected Output#2

- Identification of bias (if any) and mitigation ideas

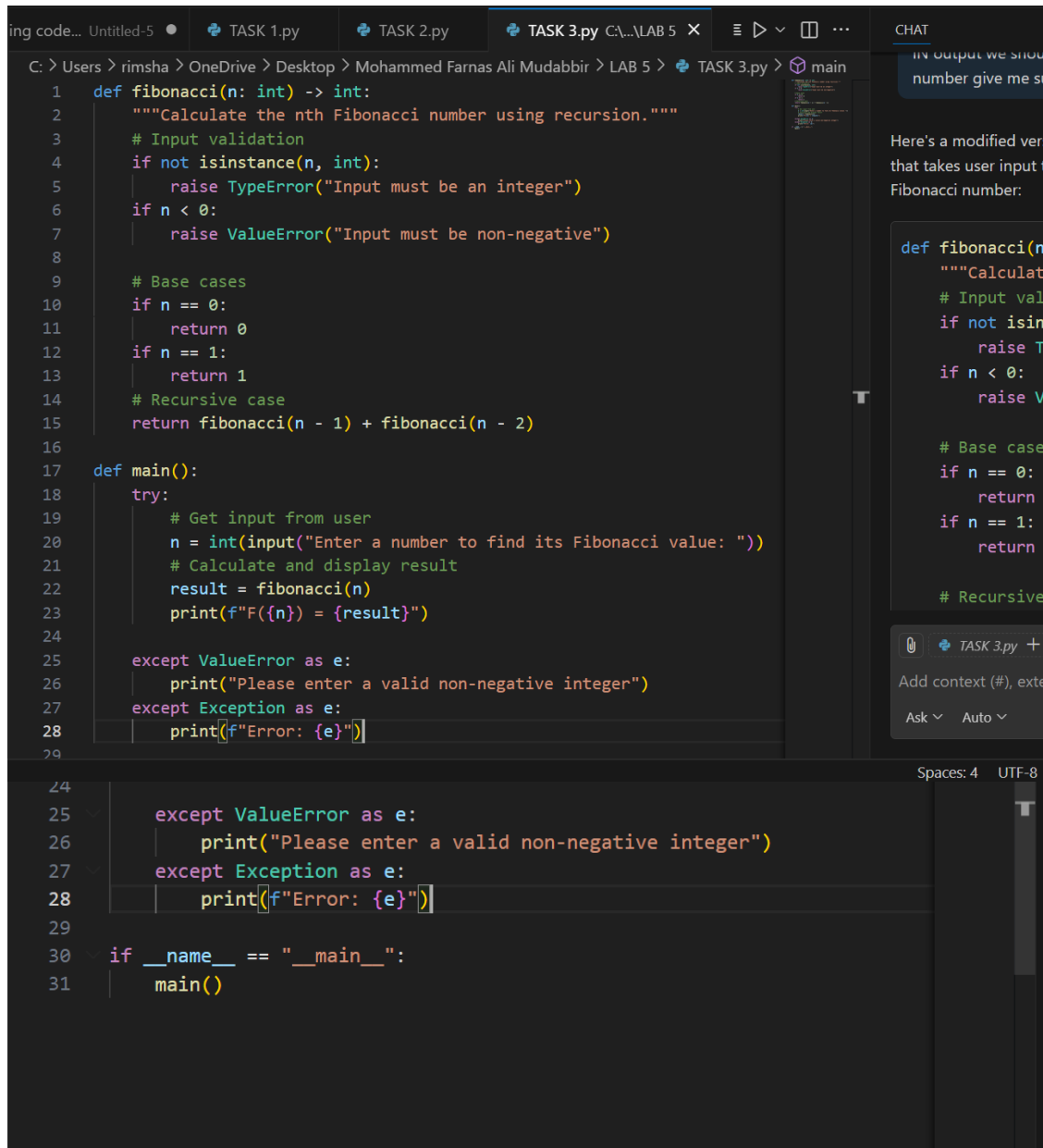
Practical Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
> TERMINAL
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1> python -u "
c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1\tempCodeRunnerF
ile.python"
Applicant: RIMSHA
Approved: True
Reason: Application approved
-----
Applicant: CHARITHA
Approved: True
Reason: Application approved
-----
Applicant: SUBHAN
Approved: True
Reason: Application approved
-----
Applicant: FAHAD
Approved: True
Reason: Application approved
-----
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1>
```

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.



```
1 def fibonacci(n: int) -> int:
2     """Calculate the nth Fibonacci number using recursion."""
3     # Input validation
4     if not isinstance(n, int):
5         raise TypeError("Input must be an integer")
6     if n < 0:
7         raise ValueError("Input must be non-negative")
8
9     # Base cases
10    if n == 0:
11        return 0
12    if n == 1:
13        return 1
14    # Recursive case
15    return fibonacci(n - 1) + fibonacci(n - 2)
16
17 def main():
18     try:
19         # Get input from user
20         n = int(input("Enter a number to find its Fibonacci value: "))
21         # Calculate and display result
22         result = fibonacci(n)
23         print(f"F({n}) = {result}")
24
25     except ValueError as e:
26         print("Please enter a valid non-negative integer")
27     except Exception as e:
28         print(f"Error: {e}")
29
30 if __name__ == "__main__":
31     main()
```

Here's a modified version of the code that takes user input to find the Fibonacci number:

```
def fibonacci(n):
    """Calculate the nth Fibonacci number using recursion."""
    # Input validation
    if not isinstance(n, int):
        raise TypeError("Input must be an integer")
    if n < 0:
        raise ValueError("Input must be non-negative")

    # Base cases
    if n == 0:
        return 0
    if n == 1:
        return 1
    # Recursive case
    return fibonacci(n - 1) + fibonacci(n - 2)
```

Spaces: 4 UTF-8

Expected Output#3

- Code with explanation

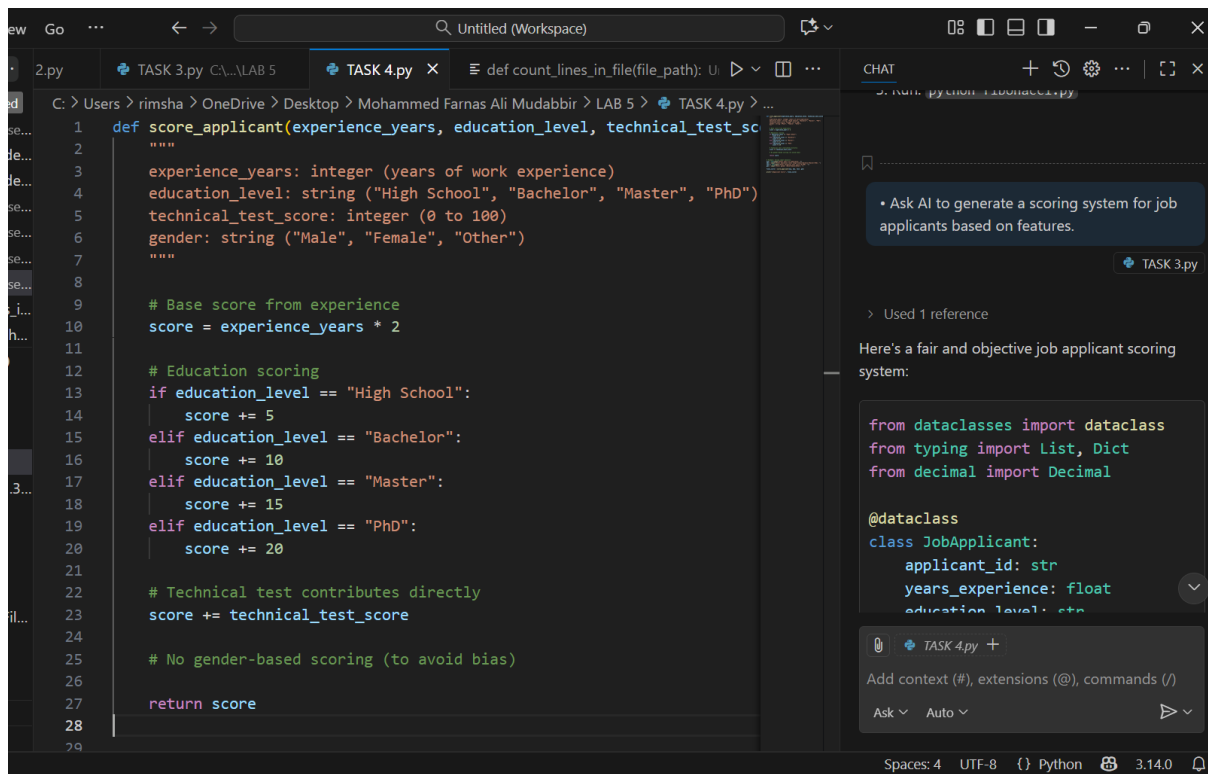
Practical Output:

```
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1\tempCodeRunnerFile.python"
Enter a number to find its Fibonacci value: 8
F(8) = 21
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1>
```

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.



```
def score_applicant(experience_years, education_level, technical_test_score):
    # Technical test contributes directly
    score += technical_test_score

    # 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)
```

CHAT

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

> Used 1 reference

Here's a fair and objective job applicant scoring system:

```
from dataclasses import dataclass
from typing import List, Dict
from decimal import Decimal

@dataclass
class JobApplicant:
    applicant_id: str
    years_experience: float
    education_level: str
```

Expected Output#4

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

Practical Output:

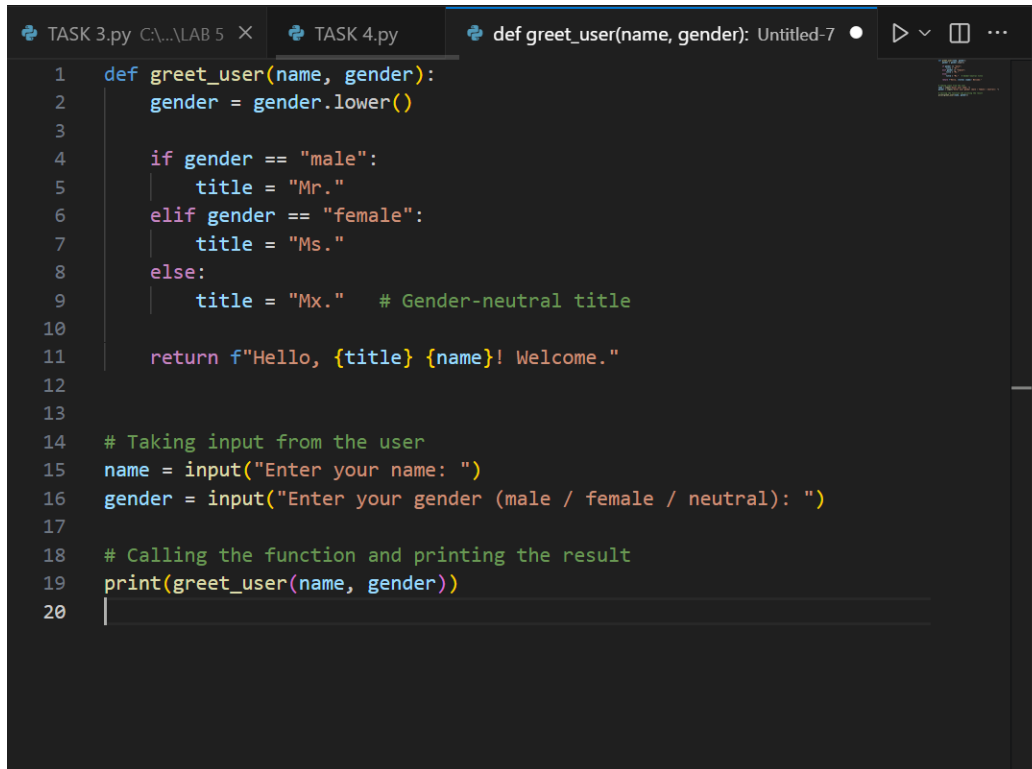
```
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1\tempCodeRunnerFile.python"
Enter years of experience: 2
Enter education level (High School/Bachelor/Master/PhD): Master
Enter technical test score (0-100): 80
Enter gender (Male/Female/Other): Male

Applicant Score: 99
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1>
```

Task Description#5 (Inclusiveness)

- Code Snippet

PROMPT: Regenerate code that includes gender-neutral also.

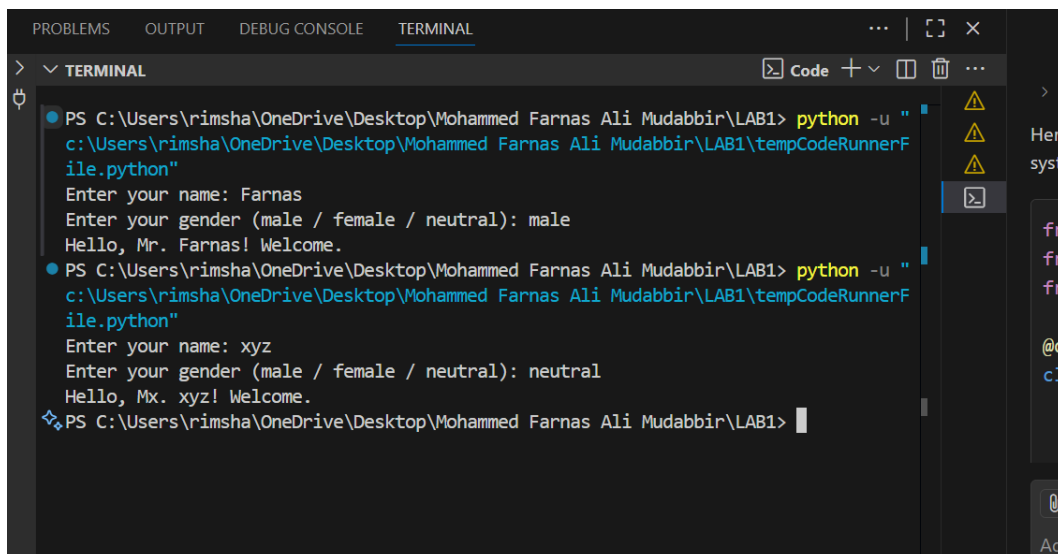


```
1 def greet_user(name, gender):
2     gender = gender.lower()
3
4     if gender == "male":
5         title = "Mr."
6     elif gender == "female":
7         title = "Ms."
8     else:
9         title = "Mx." # Gender-neutral title
10
11     return f"Hello, {title} {name}! Welcome."
12
13
14 # Taking input from the user
15 name = input("Enter your name: ")
16 gender = input("Enter your gender (male / female / neutral): ")
17
18 # Calling the function and printing the result
19 print(greet_user(name, gender))
20
```

Expected Output#5

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

Practical Output:



```
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1\tempCodeRunnerFile.python"
Enter your name: Farnas
Enter your gender (male / female / neutral): male
Hello, Mr. Farnas! Welcome.

PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1\tempCodeRunnerFile.python"
Enter your name: xyz
Enter your gender (male / female / neutral): neutral
Hello, Mx. xyz! Welcome.

PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1>
```

