SR UNIVERSITY

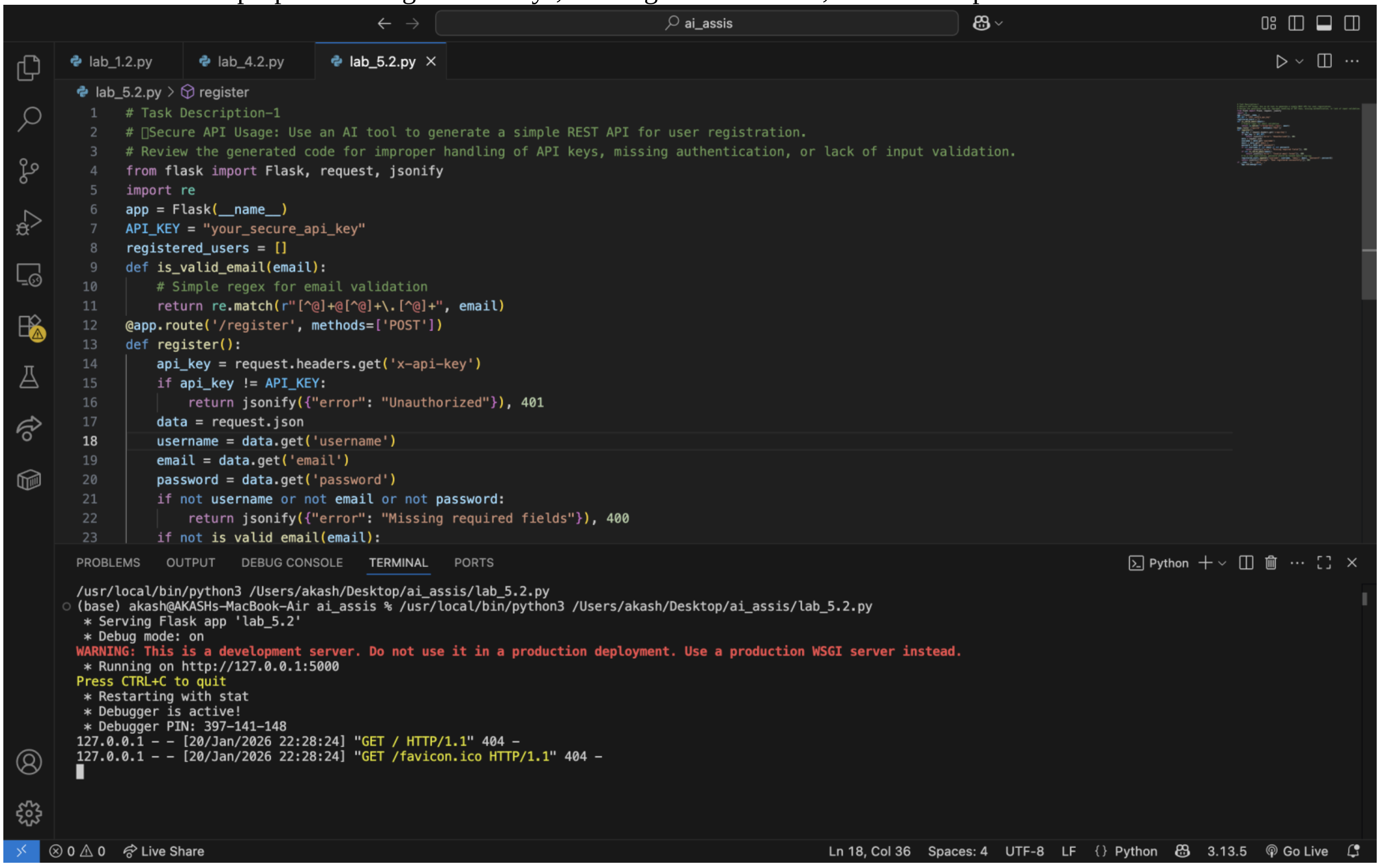
School of Computer Science and Artificial Intelligence

Lab Assignment # 5.2

Program	**: B. Tech (CSE)**
Specialization	**: --**
Course Title	**: AI Assisted coding**
Course Code	**:**
Semester	**II**
Academic Session	**: 2025-2026**
Name of Student	**: Akash Reddy**
Enrollment No.	**: 2403A51L30**
Batch No.	**: 51**
Date	**:20-01-2026**

Task Description-1

- **Secure API Usage:** Use an AI tool to generate a simple REST API for user registration. Review the generated code for improper handling of API keys, missing authentication, or lack of input validation.

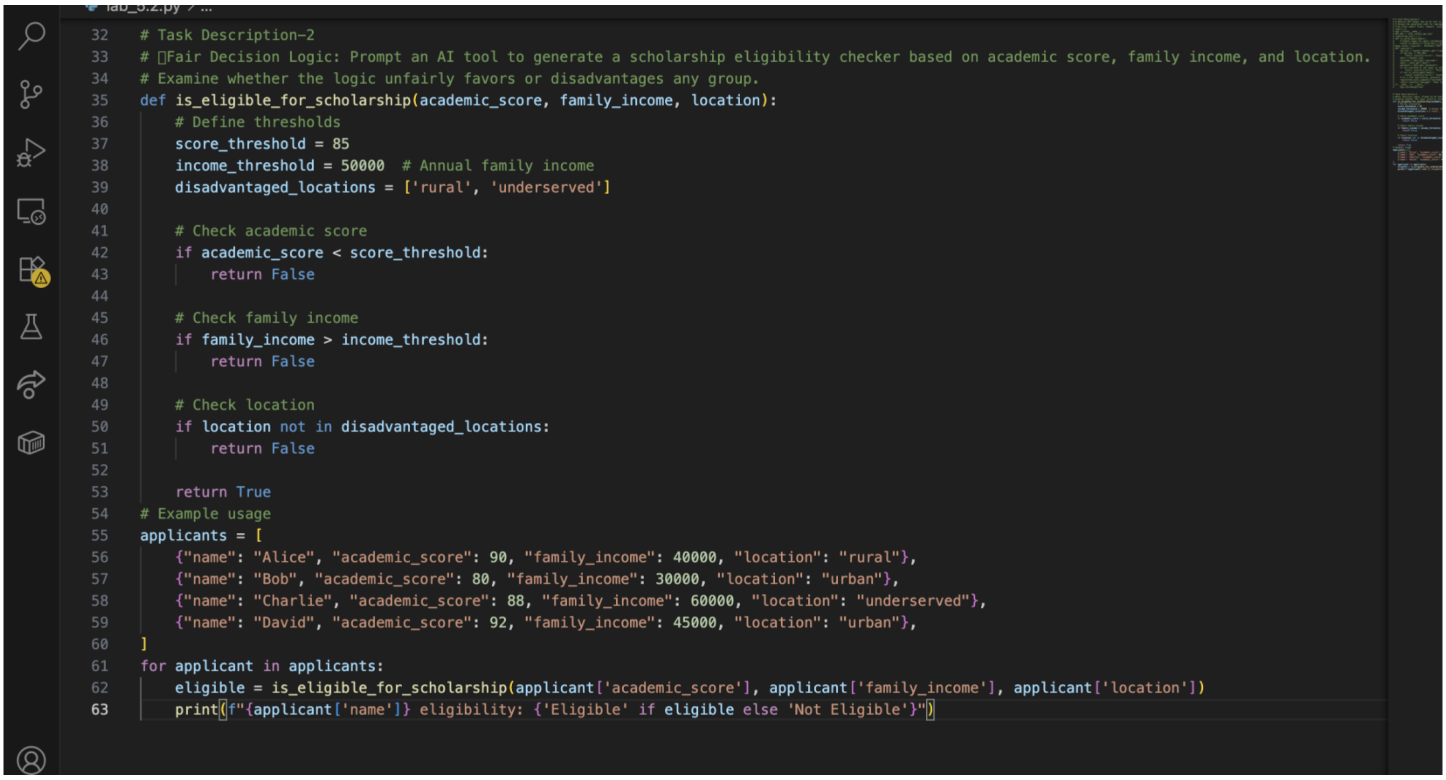


```python
# Task Description-1
# Secure API Usage: Use an AI tool to generate a simple REST API for user registration.
# Review the generated code for improper handling of API keys, missing authentication, or lack of input validation.
from flask import Flask, request, jsonify
import re
app = Flask(__name__)
API_KEY = "your_secure_api_key"
registered_users = []
def is_valid_email(email):
    # Simple regex for email validation
    return re.match(r"[^@]+@[^@]+\.[^@]+", email)
@app.route('/register', methods=['POST'])
def register():
    api_key = request.headers.get('x-api-key')
    if api_key != API_KEY:
        return jsonify({"error": "Unauthorized"}), 401
    data = request.json
    username = data.get('username')
    email = data.get('email')
    password = data.get('password')
    if not username or not email or not password:
        return jsonify({"error": "Missing required fields"}), 400
    if not is_valid_email(email):
```

```
/usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_5.2.py
(base) akash@AKASHs-MacBook-Air ai_assis % /usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_5.2.py
 * Serving Flask app 'lab_5.2'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 397-141-148
127.0.0.1 - - [20/Jan/2026 22:28:24] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [20/Jan/2026 22:28:24] "GET /favicon.ico HTTP/1.1" 404 -
```

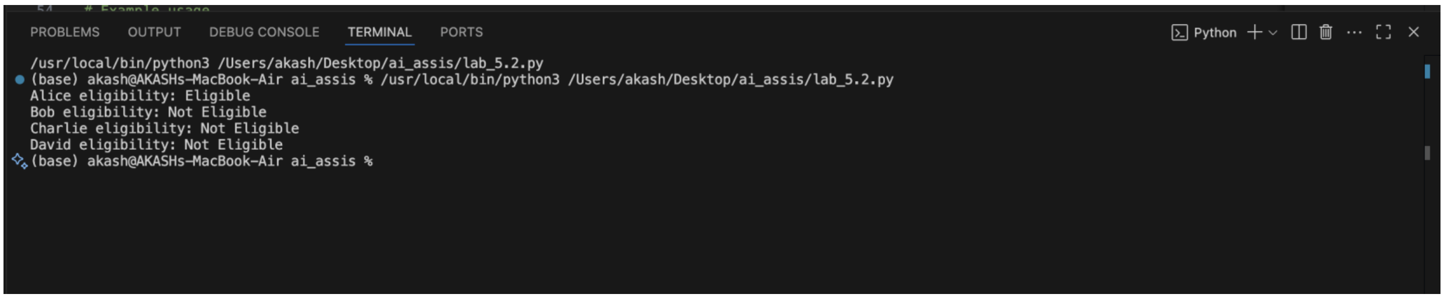
Task Description-2

- **Fair Decision Logic:** Prompt an AI tool to generate a scholarship eligibility checker based on academic score, family income, and location. Examine whether the logic unfairly favors or disadvantages any group.



```python
# Task Description-2
# □Fair Decision Logic: Prompt an AI tool to generate a scholarship eligibility checker based on academic score, family income, and location.
# Examine whether the logic unfairly favors or disadvantages any group.
def is_eligible_for_scholarship(academic_score, family_income, location):
    # Define thresholds
    score_threshold = 85
    income_threshold = 50000  # Annual family income
    disadvantaged_locations = ['rural', 'underserved']

    # Check academic score
    if academic_score < score_threshold:
        return False

    # Check family income
    if family_income > income_threshold:
        return False

    # Check location
    if location not in disadvantaged_locations:
        return False

    return True
# Example usage
applicants = [
    {"name": "Alice", "academic_score": 90, "family_income": 40000, "location": "rural"},
    {"name": "Bob", "academic_score": 80, "family_income": 30000, "location": "urban"},
    {"name": "Charlie", "academic_score": 88, "family_income": 60000, "location": "underserved"},
    {"name": "David", "academic_score": 92, "family_income": 45000, "location": "urban"},
]
for applicant in applicants:
    eligible = is_eligible_for_scholarship(applicant['academic_score'], applicant['family_income'], applicant['location'])
    print(f"{applicant['name']} eligibility: {'Eligible' if eligible else 'Not Eligible'}")
```

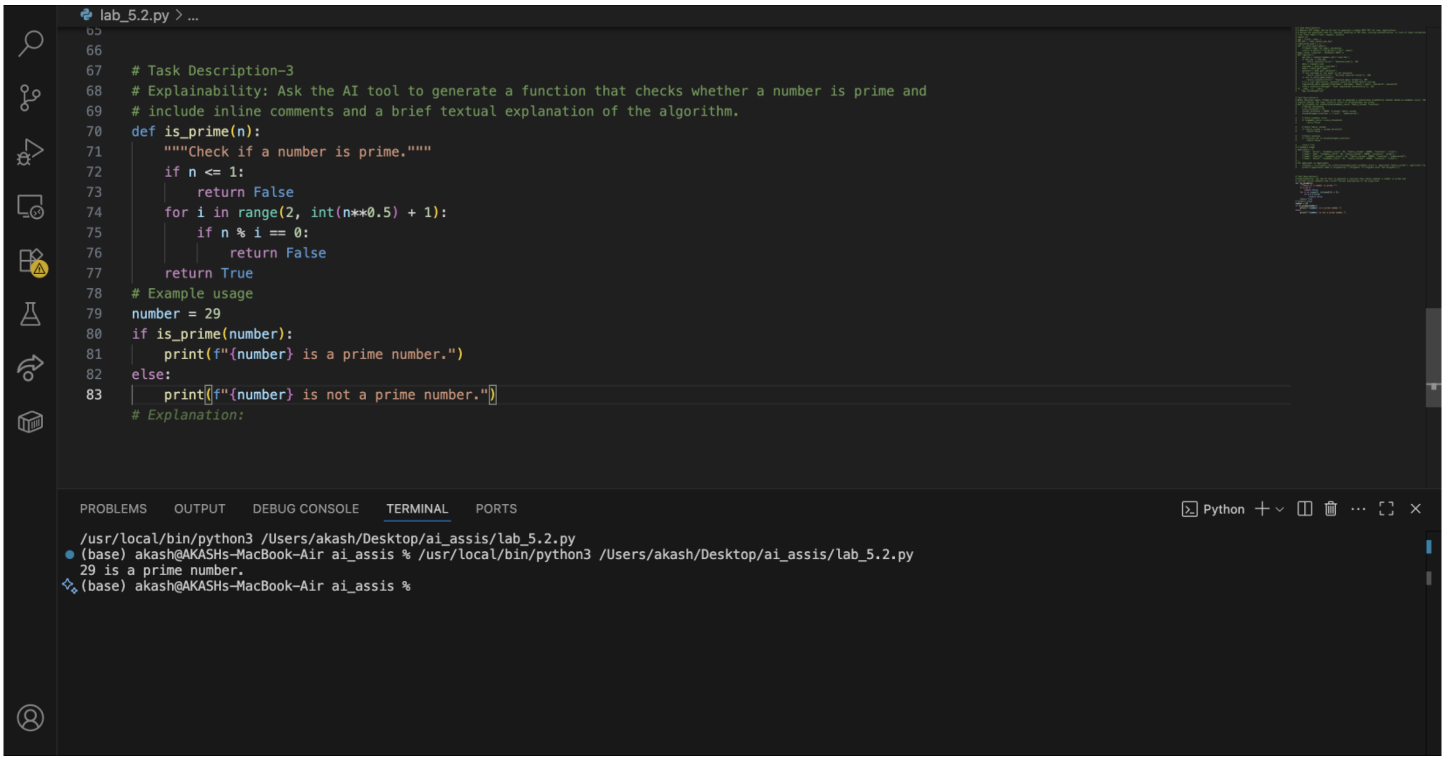
OUTPUT:



```
/usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_5.2.py
(base) akash@AKASHs-MacBook-Air ai_assis % /usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_5.2.py
Alice eligibility: Eligible
Bob eligibility: Not Eligible
Charlie eligibility: Not Eligible
David eligibility: Not Eligible
(base) akash@AKASHs-MacBook-Air ai_assis %
```

Task Description-3

Explainability: Ask the AI tool to generate a function that checks whether a number is prime and include inline comments and a brief textual explanation of the algorithm.

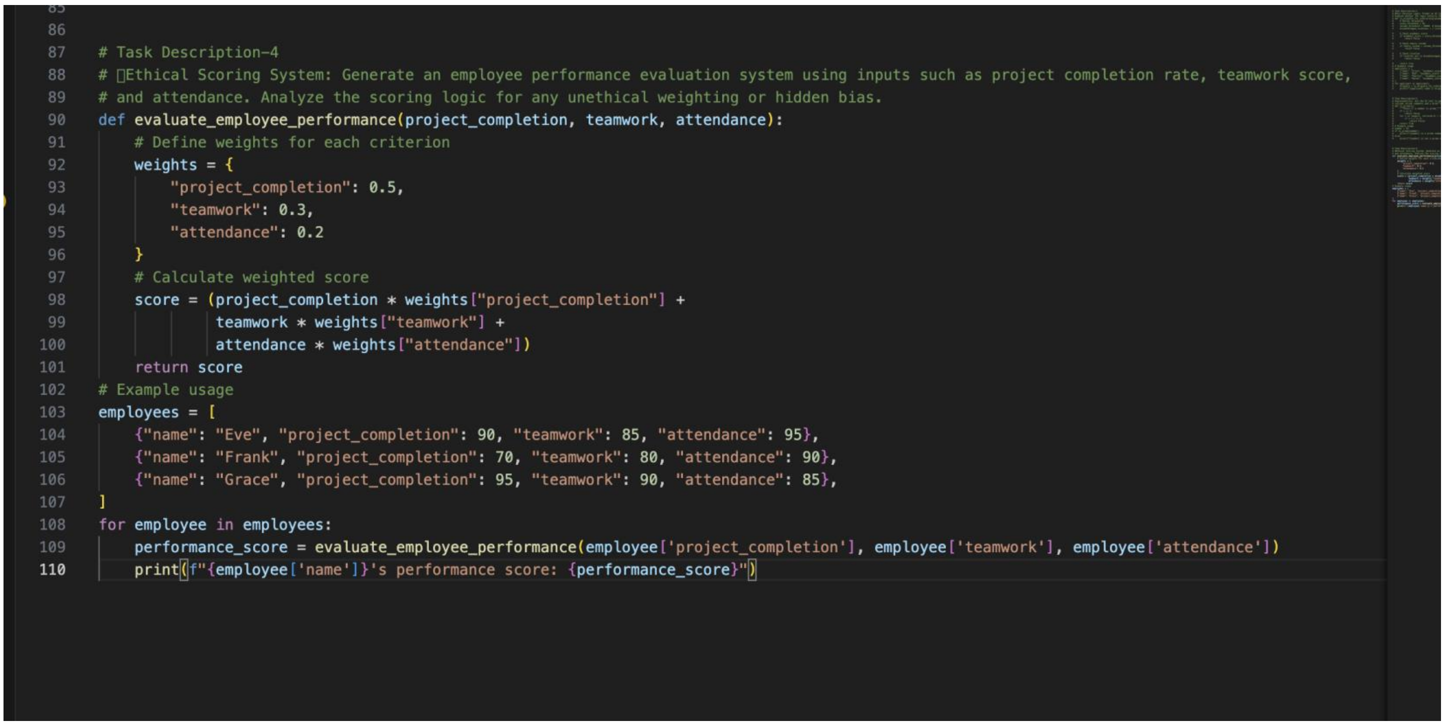


```python
# Task Description-3
# Explainability: Ask the AI tool to generate a function that checks whether a number is prime and
# include inline comments and a brief textual explanation of the algorithm.
def is_prime(n):
    """Check if a number is prime."""
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True
# Example usage
number = 29
if is_prime(number):
    print(f"{number} is a prime number.")
else:
    print(f"{number} is not a prime number.")
# Explanation:
```

```
/usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_5.2.py
(base) akash@AKASHs-MacBook-Air ai_assis % /usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_5.2.py
29 is a prime number.
(base) akash@AKASHs-MacBook-Air ai_assis %
```

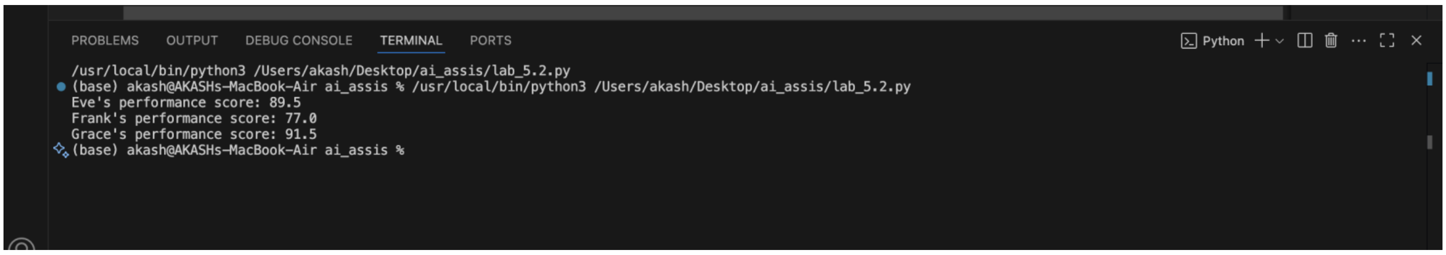
Task Description-4

- **Ethical Scoring System:** Generate an employee performance evaluation system using inputs such as project completion rate, teamwork score, and attendance. Analyze the scoring logic for any unethical weighting or hidden bias.



```python
# Task Description-4
# □Ethical Scoring System: Generate an employee performance evaluation system using inputs such as project completion rate, teamwork score,
# and attendance. Analyze the scoring logic for any unethical weighting or hidden bias.
def evaluate_employee_performance(project_completion, teamwork, attendance):
    # Define weights for each criterion
    weights = {
        "project_completion": 0.5,
        "teamwork": 0.3,
        "attendance": 0.2
    }
    # Calculate weighted score
    score = (project_completion * weights["project_completion"] +
             teamwork * weights["teamwork"] +
             attendance * weights["attendance"])
    return score
# Example usage
employees = [
    {"name": "Eve", "project_completion": 90, "teamwork": 85, "attendance": 95},
    {"name": "Frank", "project_completion": 70, "teamwork": 80, "attendance": 90},
    {"name": "Grace", "project_completion": 95, "teamwork": 90, "attendance": 85},
]
for employee in employees:
    performance_score = evaluate_employee_performance(employee['project_completion'], employee['teamwork'], employee['attendance'])
    print(f"{employee['name']}'s performance score: {performance_score}")
```

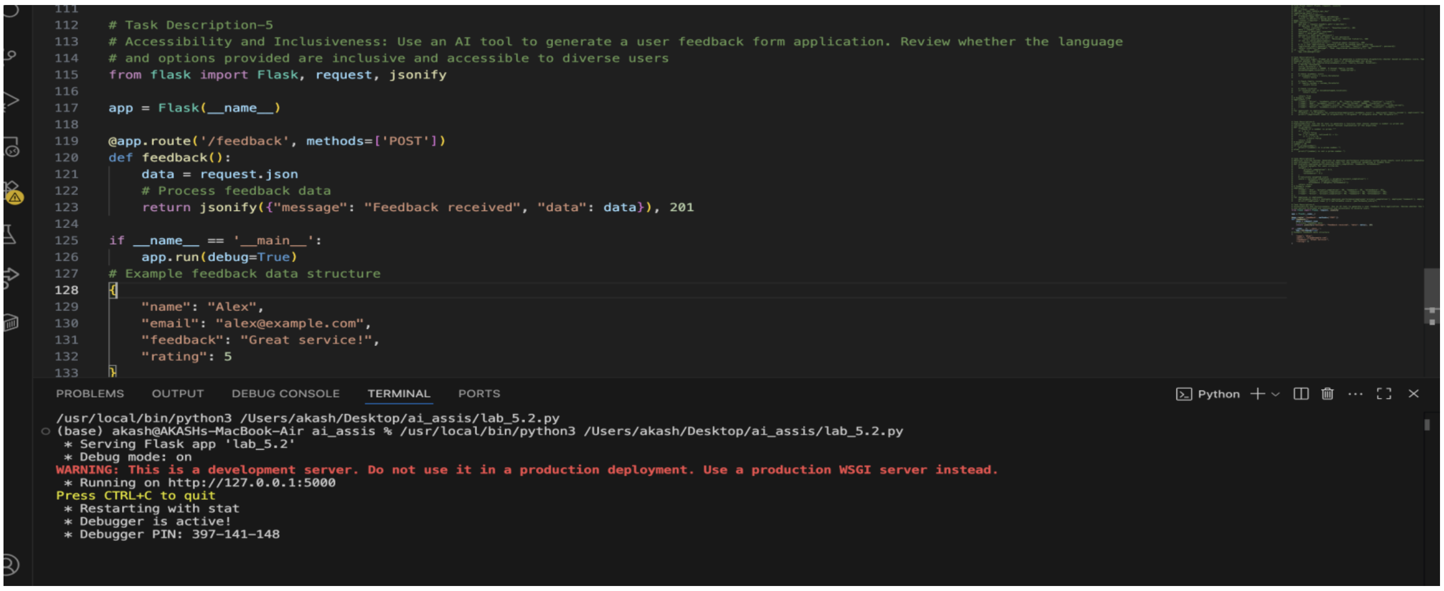
OUTPUT:



```
/usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_5.2.py
(base) akash@AKASHs-MacBook-Air ai_assis % /usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_5.2.py
Eve's performance score: 89.5
Frank's performance score: 77.0
Grace's performance score: 91.5
(base) akash@AKASHs-MacBook-Air ai_assis %
```

Task Description-5

- **Accessibility and Inclusiveness:** Use an AI tool to generate a user feedback form application. Review whether the language and options provided are inclusive and accessible to diverse users.



```python
# Task Description-5
# Accessibility and Inclusiveness: Use an AI tool to generate a user feedback form application. Review whether the language
# and options provided are inclusive and accessible to diverse users
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/feedback', methods=['POST'])
def feedback():
    data = request.json
    # Process feedback data
    return jsonify({"message": "Feedback received", "data": data}), 201

if __name__ == '__main__':
    app.run(debug=True)
# Example feedback data structure
{
    "name": "Alex",
    "email": "alex@example.com",
    "feedback": "Great service!",
    "rating": 5
}
```

```
/usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_5.2.py
(base) akash@AKASHs-MacBook-Air ai_assis % /usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_5.2.py
 * Serving Flask app 'lab_5.2'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 397-141-148
```