SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE	DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

ProgramName:B. Tech	**Assignment Type: Lab**	**AcademicYear:**2025-2026

CourseCoordinatorName	Venkataramana Veeramsetty
Instructor(s)Name	Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)

CourseCode	24CS002PC215	**CourseTitle**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week4 - Tuesday	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	

AssignmentNumber:8.2(Present assignment number)/**24**(Total number of assignments)

Q.No.	**Question**	***ExpectedTime to complete***
1	Lab 8: Test-Driven Development with AI – Generating and Working with Test Cases **Lab Objectives:** • To introduce students to test-driven development (TDD) using AI code generation tools. • To enable the generation of test cases before writing code implementations.	Week4 - Wednesday

- To reinforce the importance of testing, validation, and error handling.
- To encourage writing clean and reliable code based on AI-generated test expectations.

Lab Outcomes (LOs):
After completing this lab, students will be able to:

- Use AI tools to write test cases for Python functions and classes.
- Implement functions based on test cases in a test-first development style.
- Use unittest or pytest to validate code correctness.
- Analyze the completeness and coverage of AI-generated tests.
- Compare AI-generated and manually written test cases for quality and logic

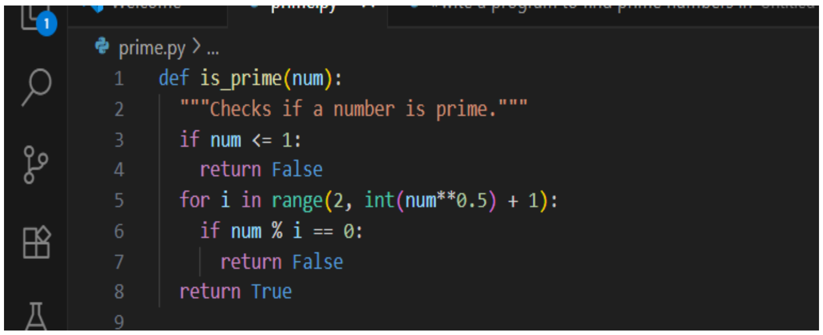
Task Description#1

Use AI to generate test cases for a function is_prime(n) and then implement the function.

Requirements:

- Only integers > 1 can be prime.
- Check edge cases: 0, 1, 2, negative numbers, and large primes.

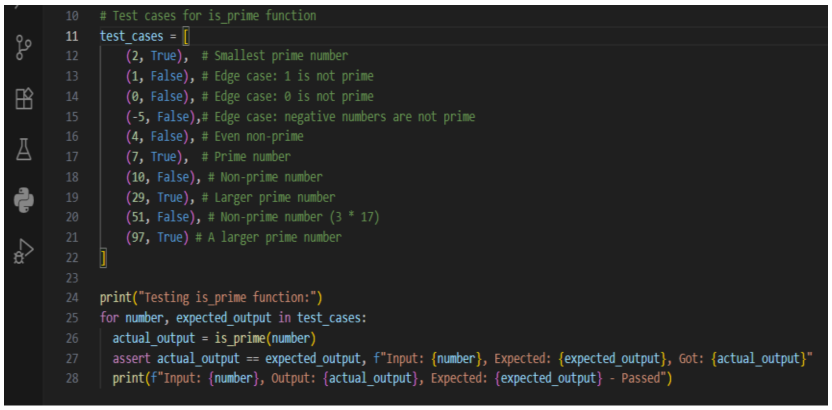
- **Prompt : Generate a program to find prime using function.**



prime.py > ...

```python
def is_prime(num):
  """Checks if a number is prime."""
  if num <= 1:
    return False
  for i in range(2, int(num**0.5) + 1):
    if num % i == 0:
      return False
  return True
```

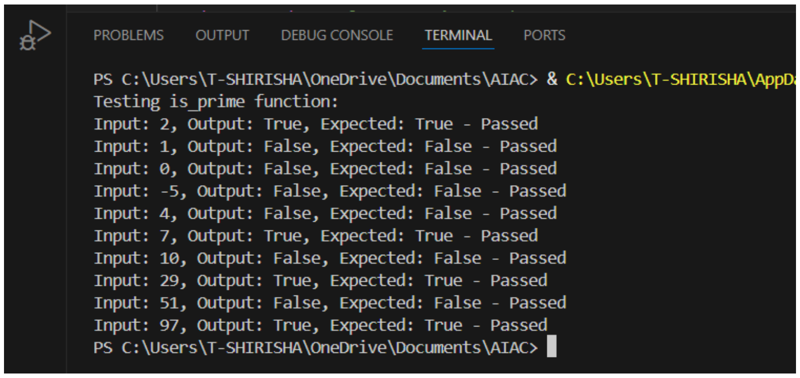
Test cases :



```python
# Test cases for is_prime function
test_cases = [
    (2, True),  # Smallest prime number
    (1, False), # Edge case: 1 is not prime
    (0, False), # Edge case: 0 is not prime
    (-5, False),# Edge case: negative numbers are not prime
    (4, False), # Even non-prime
    (7, True),  # Prime number
    (10, False), # Non-prime number
    (29, True), # Larger prime number
    (51, False), # Non-prime number (3 * 17)
    (97, True) # A larger prime number
]

print("Testing is_prime function:")
for number, expected_output in test_cases:
  actual_output = is_prime(number)
  assert actual_output == expected_output, f"Input: {number}, Expected: {expected_output}, Got: {actual_output}"
  print(f"Input: {number}, Output: {actual_output}, Expected: {expected_output} - Passed")
```

Expected Output#1

- A working prime checker that passes AI-generated tests using edge coverage.



```
PS C:\Users\T-SHIRISHA\OneDrive\Documents\AIAC> & C:\Users\T-SHIRISHA\AppD
Testing is_prime function:
Input: 2, Output: True, Expected: True - Passed
Input: 1, Output: False, Expected: False - Passed
Input: 0, Output: False, Expected: False - Passed
Input: -5, Output: False, Expected: False - Passed
Input: 4, Output: False, Expected: False - Passed
Input: 7, Output: True, Expected: True - Passed
Input: 10, Output: False, Expected: False - Passed
Input: 29, Output: True, Expected: True - Passed
Input: 51, Output: False, Expected: False - Passed
Input: 97, Output: True, Expected: True - Passed
PS C:\Users\T-SHIRISHA\OneDrive\Documents\AIAC>
```

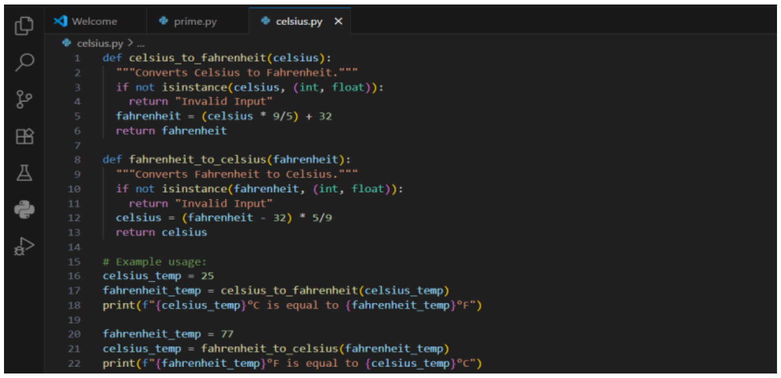
Task Description#2 (Loops)

- Ask AI to generate test cases for celsius_to_fahrenheit(c) and fahrenheit_to_celsius(f).

Requirements

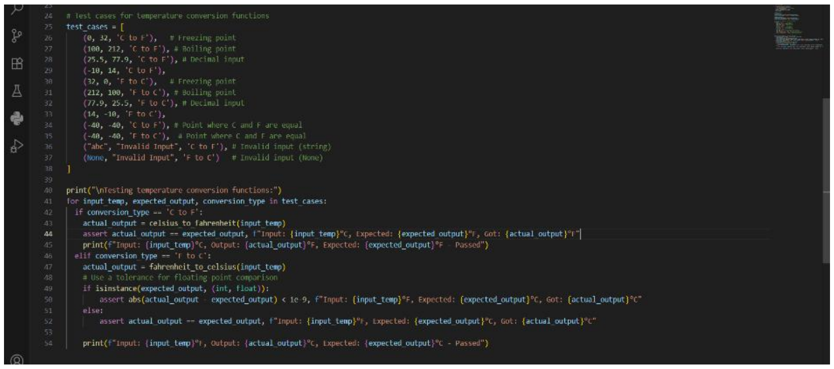
- Validate known pairs: 0°C = 32°F, 100°C = 212°F.
- Include decimals and invalid inputs like strings or None

- **Prompt : Generate a program to convert for celsius_to_fahrenheit(c) and fahrenheit_to_celsius(f).**



```python
def celsius_to_fahrenheit(celsius):
    """Converts Celsius to Fahrenheit."""
    if not isinstance(celsius, (int, float)):
        return "Invalid Input"
    fahrenheit = (celsius * 9/5) + 32
    return fahrenheit

def fahrenheit_to_celsius(fahrenheit):
    """Converts Fahrenheit to Celsius."""
    if not isinstance(fahrenheit, (int, float)):
        return "Invalid Input"
    celsius = (fahrenheit - 32) * 5/9
    return celsius

# Example usage:
celsius_temp = 25
fahrenheit_temp = celsius_to_fahrenheit(celsius_temp)
print(f"{celsius_temp}°C is equal to {fahrenheit_temp}°F")

fahrenheit_temp = 77
celsius_temp = fahrenheit_to_celsius(fahrenheit_temp)
print(f"{fahrenheit_temp}°F is equal to {celsius_temp}°C")
```

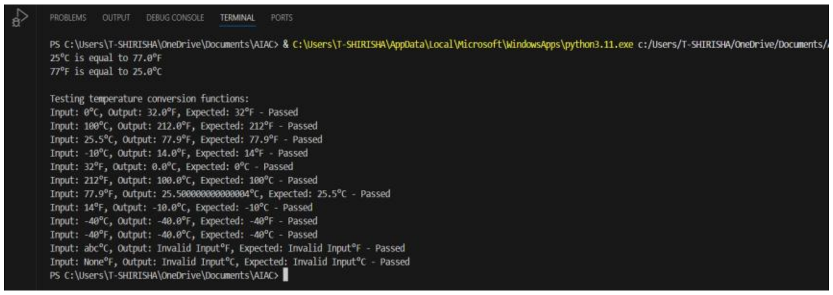
Test cases :



```python
# Test cases for temperature conversion functions
test_cases = [
    (0, 32, 'C to F'),   # Freezing point
    (100, 212, 'C to F'), # Boiling point
    (25.5, 77.9, 'C to F'), # Decimal input
    (-10, 14, 'C to F'),
    (32, 0, 'F to C'),   # Freezing point
    (212, 100, 'F to C'), # Boiling point
    (77.9, 25.5, 'F to C'), # Decimal input
    (14, -10, 'F to C'),
    (-40, -40, 'C to F'), # Point where C and F are equal
    (-40, -40, 'F to C'),  # Point where C and F are equal
    ("abc", "Invalid Input", 'C to F'), # Invalid input (string)
    (None, "Invalid Input", 'F to C')   # Invalid input (None)
]

print("\nTesting temperature conversion functions:")
for input_temp, expected_output, conversion_type in test_cases:
  if conversion_type == 'C to F':
    actual_output = celsius_to_fahrenheit(input_temp)
    assert actual_output == expected_output, f"Input: {input_temp}°C, Expected: {expected_output}°F, Got: {actual_output}°F"
    print(f"Input: {input_temp}°C, Output: {actual_output}°F, Expected: {expected_output}°F - Passed")
  elif conversion_type == 'F to C':
    actual_output = fahrenheit_to_celsius(input_temp)
    # Use a tolerance for floating point comparison
    if isinstance(expected_output, (int, float)):
        assert abs(actual_output - expected_output) < 1e-9, f"Input: {input_temp}°F, Expected: {expected_output}°C, Got: {actual_output}°C"
    else:
        assert actual_output == expected_output, f"Input: {input_temp}°F, Expected: {expected_output}°C, Got: {actual_output}°C"

    print(f"Input: {input_temp}°F, Output: {actual_output}°C, Expected: {expected_output}°C - Passed")
```

Expected Output#2

Dual conversion functions with complete test coverage and safe type handling



```
PS C:\Users\T-SHIRISHA\OneDrive\Documents\AIAC> & C:\Users\T-SHIRISHA\AppData\Local\Microsoft\WindowsApps\python3.11.exe c:/Users/T-SHIRISHA/OneDrive/Documents/
25°C is equal to 77.0°F
77°F is equal to 25.0°C

Testing temperature conversion functions:
Input: 0°C, Output: 32.0°F, Expected: 32°F - Passed
Input: 100°C, Output: 212.0°F, Expected: 212°F - Passed
Input: 25.5°C, Output: 77.9°F, Expected: 77.9°F - Passed
Input: -10°C, Output: 14.0°F, Expected: 14°F - Passed
Input: 32°F, Output: 0.0°C, Expected: 0°C - Passed
Input: 212°F, Output: 100.0°C, Expected: 100°C - Passed
Input: 77.9°F, Output: 25.500000000000004°C, Expected: 25.5°C - Passed
Input: 14°F, Output: -10.0°C, Expected: -10°C - Passed
Input: -40°C, Output: -40.0°F, Expected: -40°F - Passed
Input: -40°F, Output: -40.0°C, Expected: -40°C - Passed
Input: abc°C, Output: Invalid Input°F, Expected: Invalid Input°F - Passed
Input: None°F, Output: Invalid Input°C, Expected: Invalid Input°C - Passed
PS C:\Users\T-SHIRISHA\OneDrive\Documents\AIAC>
```

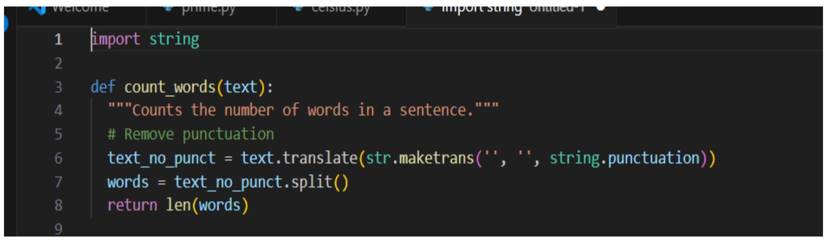
Task Description#3

- Use AI to write test cases for a function count_words(text) that returns the number of words in a sentence.

Requirement

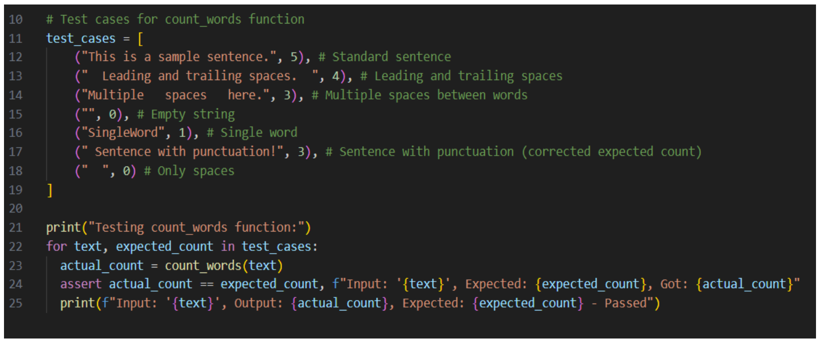
- Handle normal text, multiple spaces, punctuation, and empty strings.

- **Prompt : Generate a python function to count_words(text) that returns the number of words in a sentence.**



```python
import string

def count_words(text):
  """Counts the number of words in a sentence."""
  # Remove punctuation
  text_no_punct = text.translate(str.maketrans('', '', string.punctuation))
  words = text_no_punct.split()
  return len(words)
```

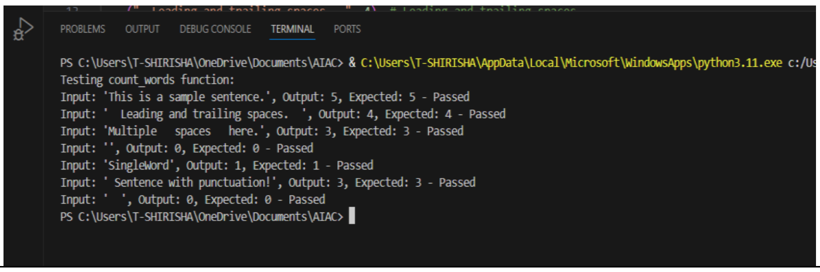
Test Cases :



```python
# Test cases for count_words function
test_cases = [
    ("This is a sample sentence.", 5), # Standard sentence
    ("  Leading and trailing spaces.  ", 4), # Leading and trailing spaces
    ("Multiple   spaces   here.", 3), # Multiple spaces between words
    ("", 0), # Empty string
    ("SingleWord", 1), # Single word
    (" Sentence with punctuation!", 3), # Sentence with punctuation (corrected expected count)
    ("  ", 0) # Only spaces
]

print("Testing count_words function:")
for text, expected_count in test_cases:
  actual_count = count_words(text)
  assert actual_count == expected_count, f"Input: '{text}', Expected: {expected_count}, Got: {actual_count}"
  print(f"Input: '{text}', Output: {actual_count}, Expected: {expected_count} - Passed")
```

Expected Output#3

Accurate word count with robust test case validation.



```
PS C:\Users\T-SHIRISHA\OneDrive\Documents\AIAC> & C:\Users\T-SHIRISHA\AppData\Local\Microsoft\WindowsApps\python3.11.exe c:/Us
Testing count_words function:
Input: 'This is a sample sentence.', Output: 5, Expected: 5 - Passed
Input: '  Leading and trailing spaces.  ', Output: 4, Expected: 4 - Passed
Input: 'Multiple   spaces   here.', Output: 3, Expected: 3 - Passed
Input: '', Output: 0, Expected: 0 - Passed
Input: 'SingleWord', Output: 1, Expected: 1 - Passed
Input: ' Sentence with punctuation!', Output: 3, Expected: 3 - Passed
Input: '  ', Output: 0, Expected: 0 - Passed
PS C:\Users\T-SHIRISHA\OneDrive\Documents\AIAC>
```

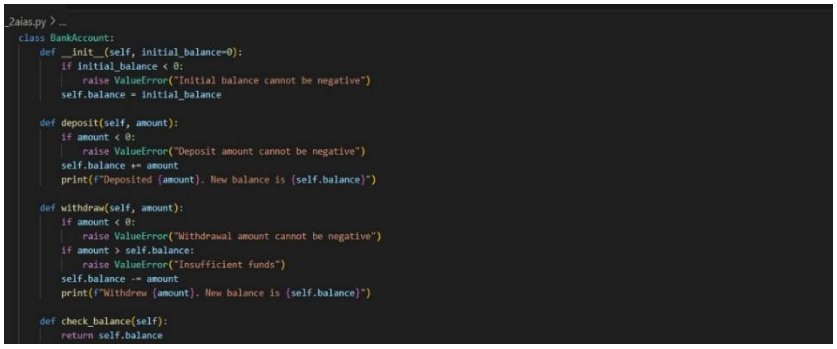
Task Description#4

- Generate test cases for a BankAccount class with:
  Methods:
  deposit(amount)
  withdraw(amount)
  check_balance()

Requirements:

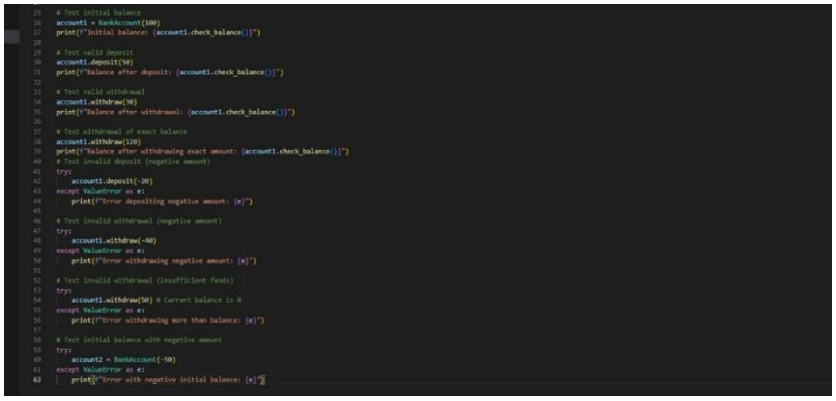
- Negative deposits/withdrawals should raise an error.
- Cannot withdraw more than balance.

- **Prompt : Generate a program for a bankAccount class using deposit(amount) withdraw(amount).**



```python
_2aias.py > ...
class BankAccount:
    def __init__(self, initial_balance=0):
        if initial_balance < 0:
            raise ValueError("Initial balance cannot be negative")
        self.balance = initial_balance

    def deposit(self, amount):
        if amount < 0:
            raise ValueError("Deposit amount cannot be negative")
        self.balance += amount
        print(f"Deposited {amount}. New balance is {self.balance}")

    def withdraw(self, amount):
        if amount < 0:
            raise ValueError("Withdrawal amount cannot be negative")
        if amount > self.balance:
            raise ValueError("Insufficient funds")
        self.balance -= amount
        print(f"Withdrew {amount}. New balance is {self.balance}")

    def check_balance(self):
        return self.balance
```

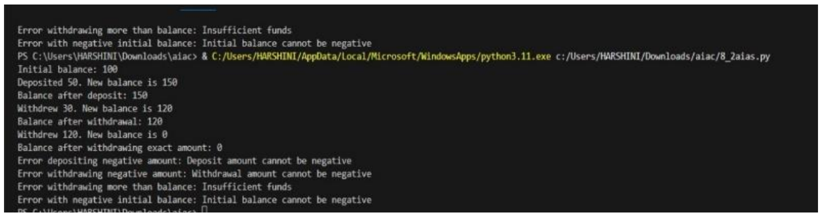
Testcases :



```python
# Test initial balance
account1 = BankAccount(100)
print(f"Initial balance: {account1.check_balance()}")

# Test valid deposit
account1.deposit(50)
print(f"Balance after deposit: {account1.check_balance()}")

# Test valid withdrawal
account1.withdraw(30)
print(f"Balance after withdrawal: {account1.check_balance()}")

# Test withdrawal of exact balance
account1.withdraw(120)
print(f"Balance after withdrawing exact amount: {account1.check_balance()}")
# Test invalid deposit (negative amount)
try:
    account1.deposit(-20)
except ValueError as e:
    print(f"Error depositing negative amount: {e}")

# Test invalid withdrawal (negative amount)
try:
    account1.withdraw(-40)
except ValueError as e:
    print(f"Error withdrawing negative amount: {e}")

# Test invalid withdrawal (insufficient funds)
try:
    account1.withdraw(50) # Current balance is 0
except ValueError as e:
    print(f"Error withdrawing more than balance: {e}")

# Test initial balance with negative amount
try:
    account2 = BankAccount(-50)
except ValueError as e:
    print(f"Error with negative initial balance: {e}")
```

Expected Output#4

- AI-generated test suite with a robust class that handles all test cases.



```
Error withdrawing more than balance: Insufficient funds
Error with negative initial balance: Initial balance cannot be negative
PS C:\Users\HARSHINI\Downloads\aiac> & C:/Users/HARSHINI/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/HARSHINI/Downloads/aiac/8_2aias.py
Initial balance: 100
Deposited 50. New balance is 150
Balance after deposit: 150
Withdrew 30. New balance is 120
Balance after withdrawal: 120
Withdrew 120. New balance is 0
Balance after withdrawing exact amount: 0
Error depositing negative amount: Deposit amount cannot be negative
Error withdrawing negative amount: Withdrawal amount cannot be negative
Error withdrawing more than balance: Insufficient funds
Error with negative initial balance: Initial balance cannot be negative
PS C:\Users\HARSHINI\Downloads\aiac>
```

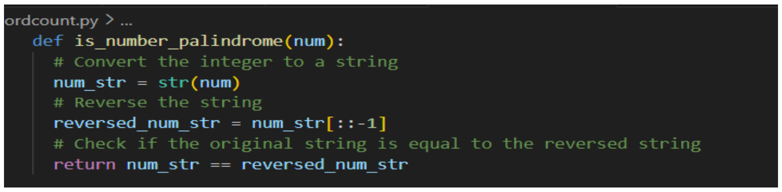
Task Description#5

• Generate test cases for is_number_palindrome(num), which checks if an integer reads the same backward.

Examples:

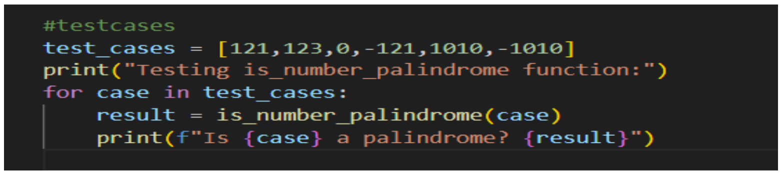
121 → True
123 → False
0, negative numbers → handled gracefully

• **Prompt : Generate a program to check is number_palindrome(num) and if an integer reads the same backward.**



```
ordcount.py > ...
  def is_number_palindrome(num):
      # Convert the integer to a string
      num_str = str(num)
      # Reverse the string
      reversed_num_str = num_str[::-1]
      # Check if the original string is equal to the reversed string
      return num_str == reversed_num_str
```

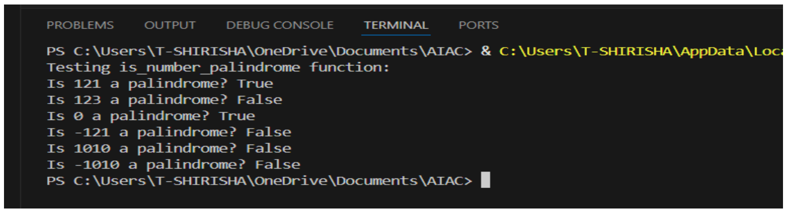
Testcases :



```
#testcases
test_cases = [121,123,0,-121,1010,-1010]
print("Testing is_number_palindrome function:")
for case in test_cases:
    result = is_number_palindrome(case)
    print(f"Is {case} a palindrome? {result}")
```

Expected Output#5

- Number-based palindrome checker function validated against test cases.



```
PS C:\Users\T-SHIRISHA\OneDrive\Documents\AIAC> & C:\Users\T-SHIRISHA\AppData\Loc
Testing is_number_palindrome function:
Is 121 a palindrome? True
Is 123 a palindrome? False
Is 0 a palindrome? True
Is -121 a palindrome? False
Is 1010 a palindrome? False
Is -1010 a palindrome? False
PS C:\Users\T-SHIRISHA\OneDrive\Documents\AIAC>
```

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
Task #1	0.5
Task #2	0.5
Task #3	0.5
Task #4	0.5
Task #5	0.5
Total	**2.5 Marks**