

LAB NAME : AI ASSISTED CODING
ROLL NO :2503A51L16
BRANCH : CSE
NAME : K. JASHUVA

TASK 1

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

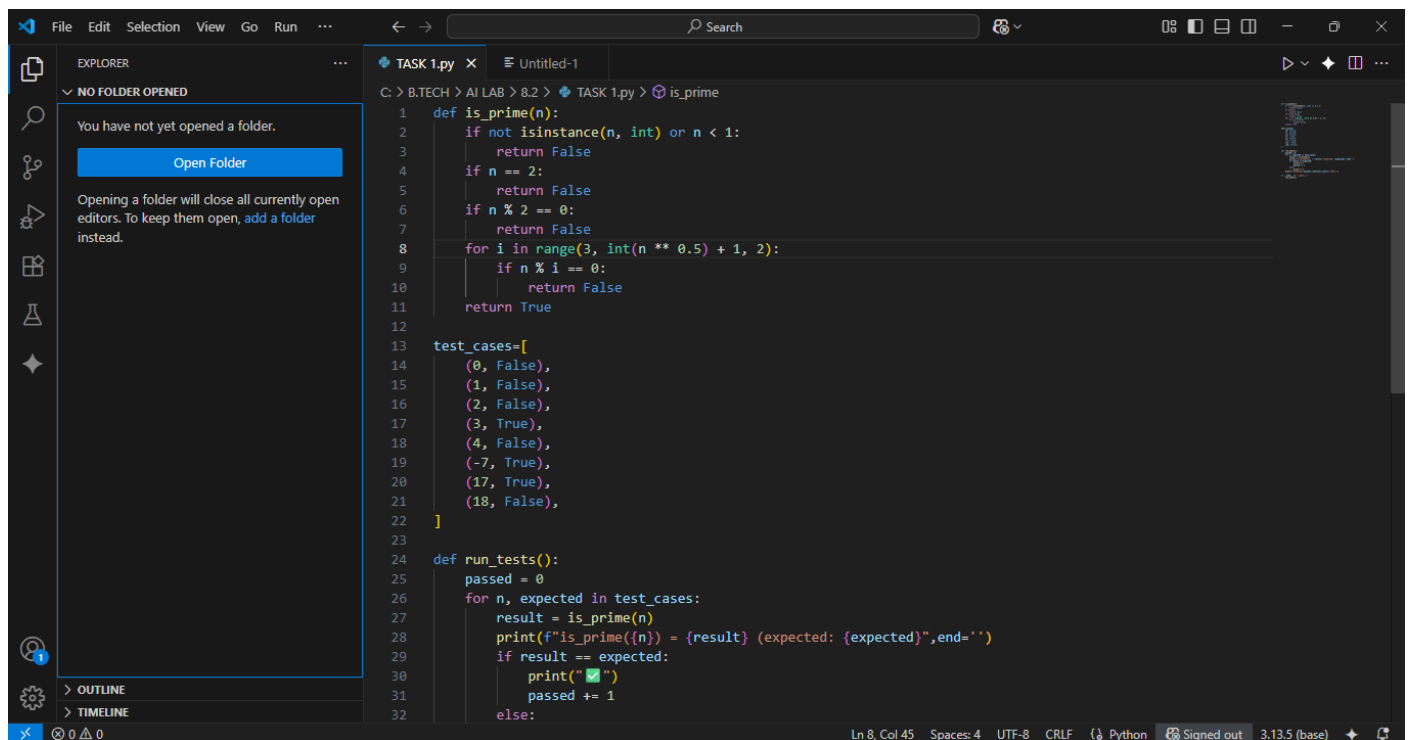
Requirements:

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

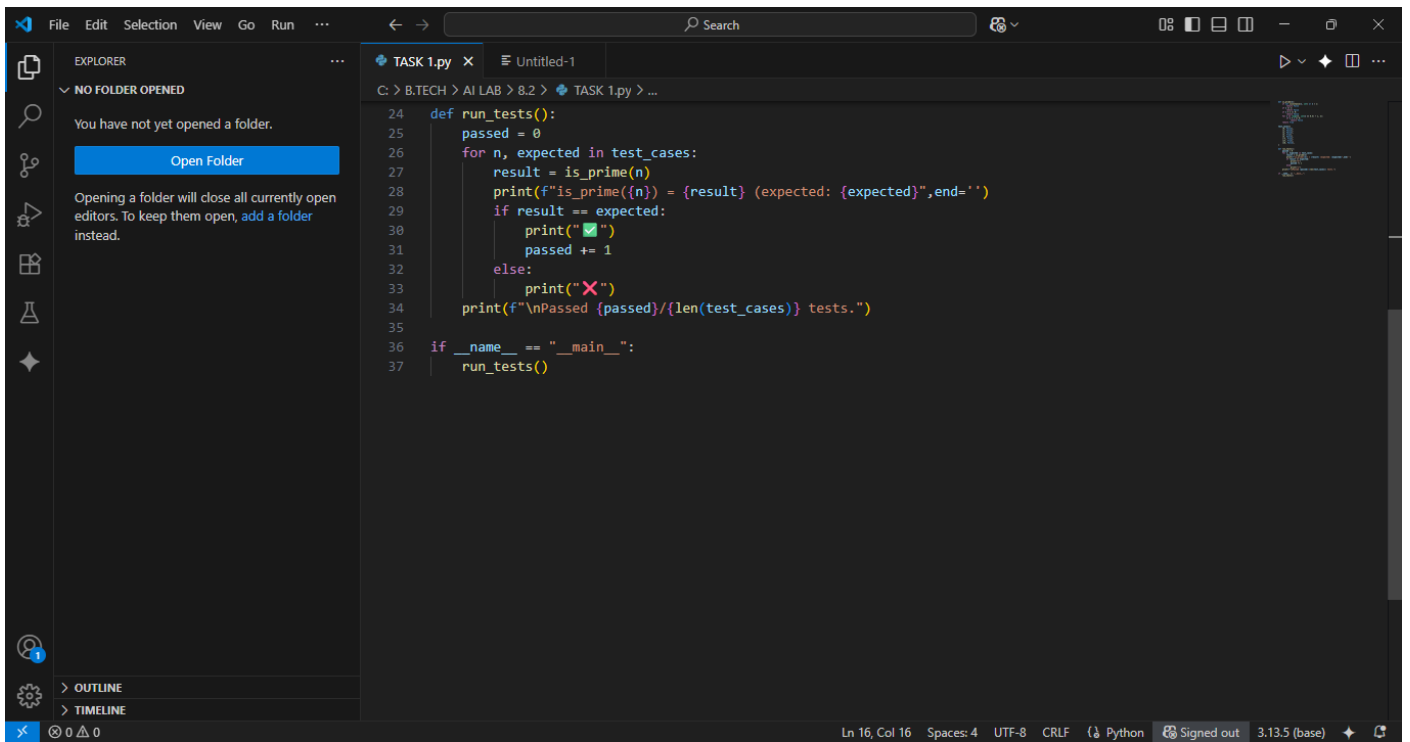
PROMPT:

Please write a Python function `is_prime(n)` and a corresponding suite of test cases to validate it. The function should be implemented first, followed by the tests.

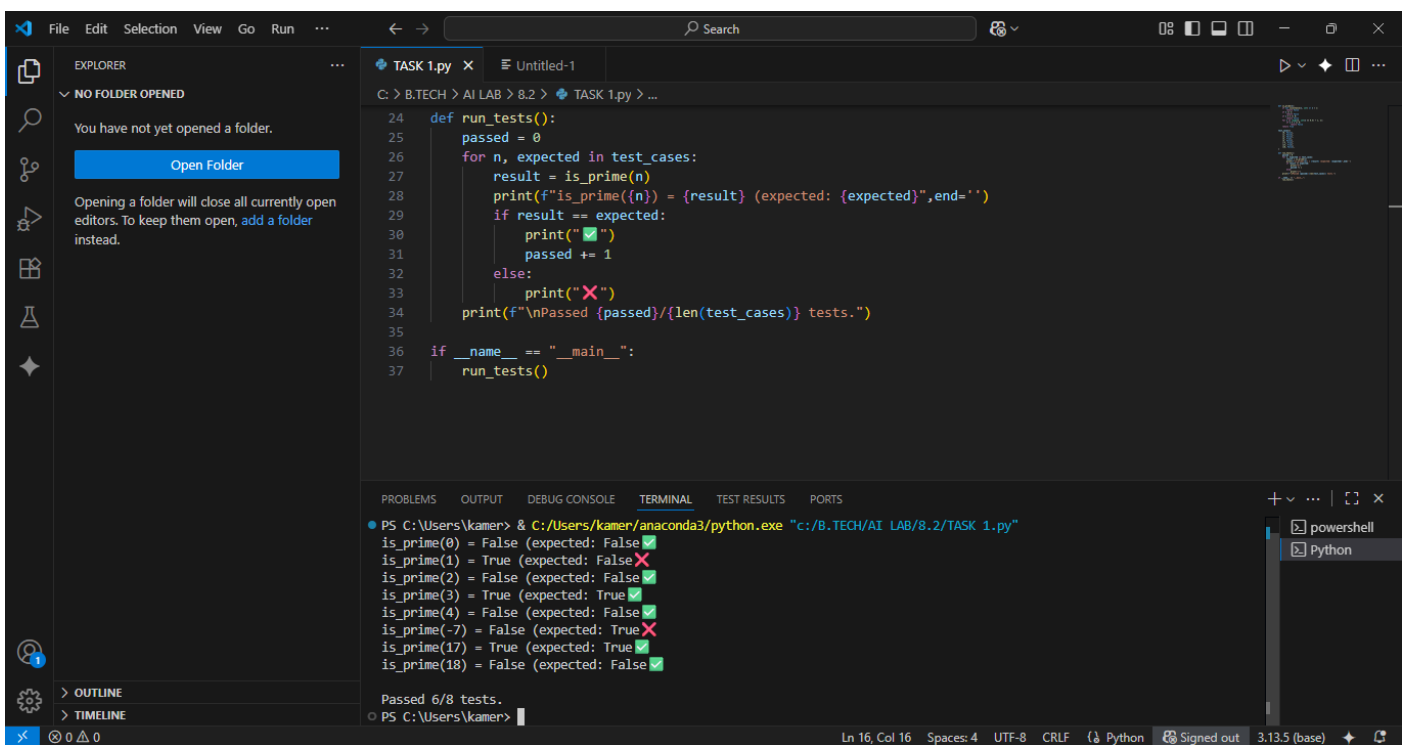
CODE:



```
1 def is_prime(n):
2     if not isinstance(n, int) or n < 1:
3         return False
4     if n == 2:
5         return True
6     if n % 2 == 0:
7         return False
8     for i in range(3, int(n ** 0.5) + 1, 2):
9         if n % i == 0:
10            return False
11    return True
12
13 test_cases=[
14     (0, False),
15     (1, False),
16     (2, False),
17     (3, True),
18     (4, False),
19     (-7, True),
20     (17, True),
21     (18, False),
22 ]
23
24 def run_tests():
25     passed = 0
26     for n, expected in test_cases:
27         result = is_prime(n)
28         print(f"is_prime({n}) = {result} (expected: {expected})",end='')
29         if result == expected:
30             print("✅")
31             passed += 1
32         else:
```



OUTPUT:



TASK 2

Task Description: Ask AI to generate test cases for `celsius_to_fahrenheit(c)` and `fahrenheit_to_celsius(f)`.

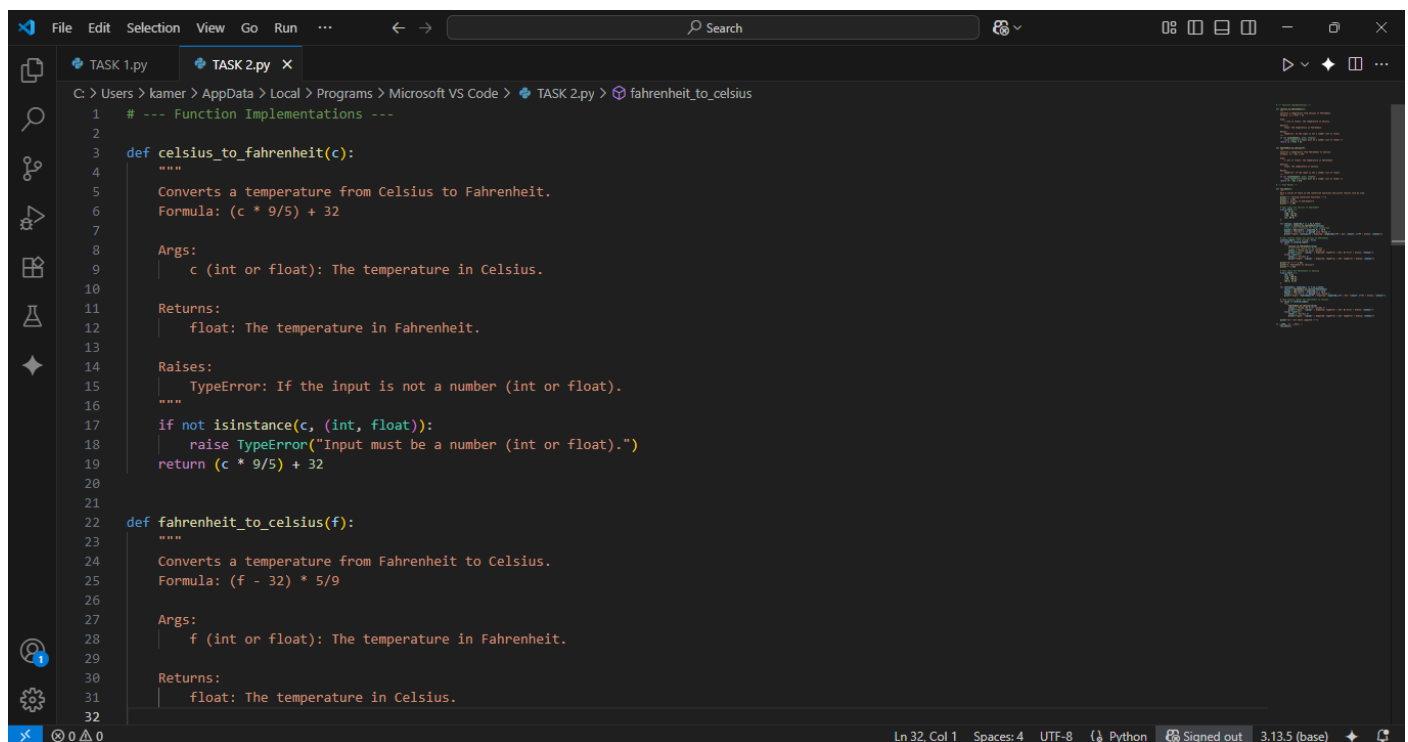
Requirements:

- Validate known pairs: $0^{\circ}\text{C} = 32^{\circ}\text{F}$, $100^{\circ}\text{C} = 212^{\circ}\text{F}$.
- Include decimals and invalid inputs like strings or None.

PROMPT:

Please write two Python functions for temperature conversion, `celsius_to_fahrenheit(c)` and `fahrenheit_to_celsius(f)`, and a corresponding suite of test cases to validate them.

CODE:



```
1  # --- Function Implementations ---
2
3  def celsius_to_fahrenheit(c):
4      """
5      Converts a temperature from Celsius to Fahrenheit.
6      Formula: (c * 9/5) + 32
7
8      Args:
9      | c (int or float): The temperature in Celsius.
10
11      Returns:
12      | float: The temperature in Fahrenheit.
13
14      Raises:
15      | TypeError: If the input is not a number (int or float).
16      """
17      if not isinstance(c, (int, float)):
18          raise TypeError("Input must be a number (int or float).")
19      return (c * 9/5) + 32
20
21
22  def fahrenheit_to_celsius(f):
23      """
24      Converts a temperature from Fahrenheit to Celsius.
25      Formula: (f - 32) * 5/9
26
27      Args:
28      | f (int or float): The temperature in Fahrenheit.
29
30      Returns:
31      | float: The temperature in Celsius.
32
33      """
34      if not isinstance(f, (int, float)):
35          raise TypeError("Input must be a number (int or float).")
36      return (f - 32) * 5/9
37
38
39  # --- Test Cases ---
40
41  def test_celsius_to_fahrenheit():
42      """Test cases for celsius_to_fahrenheit function"""
43      # Known pairs
44      assert celsius_to_fahrenheit(0) == 32
45      assert celsius_to_fahrenheit(100) == 212
46      # Decimals
47      assert celsius_to_fahrenheit(37.5) == 99.5
48      assert celsius_to_fahrenheit(-10) == 14
49      # Invalid inputs
50      assert isinstance(celsius_to_fahrenheit(100), float)
51      assert isinstance(celsius_to_fahrenheit(37.5), float)
52      assert isinstance(celsius_to_fahrenheit(-10), float)
53      # None input
54      assert celsius_to_fahrenheit(None) is None
55
56
57  def test_fahrenheit_to_celsius():
58      """Test cases for fahrenheit_to_celsius function"""
59      # Known pairs
60      assert fahrenheit_to_celsius(32) == 0
61      assert fahrenheit_to_celsius(212) == 100
62      # Decimals
63      assert fahrenheit_to_celsius(99.5) == 37.5
64      assert fahrenheit_to_celsius(14) == -10
65      # Invalid inputs
66      assert isinstance(fahrenheit_to_celsius(100), float)
67      assert isinstance(fahrenheit_to_celsius(37.5), float)
68      assert isinstance(fahrenheit_to_celsius(-10), float)
69      # None input
70      assert fahrenheit_to_celsius(None) is None
```



```
File Edit Selection View Go Run ... Search
TASK 1.py TASK 2.py x
C:\Users\kamer> AppData\Local\Programs\Microsoft VS Code> TASK 2.py > fahrenheit_to_celsius
42 def run_tests():
86     (-40, -40.0),
87     (98.6, 37.0)
88 ]
89
90 for fahrenheit, expected_c in f_to_c_tests:
91     result = fahrenheit_to_celsius(fahrenheit)
92     passed = abs(result - expected_c) < 1e-9
93     status = "Correct ✓" if passed else "Wrong ✗"
94     print(f"Input: {fahrenheit}°F | Expected: {expected_c}°C | Got: {result:.2f}°C | Status: {status}")
95
96 # Test invalid inputs for Fahrenheit to Celsius
97 for value in invalid_inputs:
98     try:
99         fahrenheit_to_celsius(value)
100         status = "Wrong (No error raised) ✗"
101         print(f"Input: '{value}' | Expected: TypeError | Got: No Error | Status: {status}")
102     except TypeError:
103         status = "Correct ✓"
104         print(f"Input: '{value}' | Expected: TypeError | Got: TypeError | Status: {status}")
105
106 print("\n--- All tests complete ---")
107
108 if __name__ == '__main__':
109     run_tests()
```

Ln 32, Col 1 Spaces: 4 UTF-8 Python Signed out 3.13.5 (base)

OUTPUT:

```
File Edit Selection View Go Run ... Search
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS
PS C:\Users\kamer> & C:\Users\kamer\anaconda3\python.exe "c:/Users/kamer/AppData/Local/Programs/Microsoft VS Code/TASK 2.py"
--- Testing Conversion Functions ---
1. Celsius to Fahrenheit
Input: 0°C | Expected: 32.0°F | Got: 32.00°F | Status: Correct ✓
Input: 100°C | Expected: 212.0°F | Got: 212.00°F | Status: Correct ✓
Input: -40°C | Expected: -40.0°F | Got: -40.00°F | Status: Correct ✓
Input: 37°C | Expected: 98.6°F | Got: 98.60°F | Status: Correct ✓
Input: 'a string' | Expected: TypeError | Got: TypeError | Status: Correct ✓
Input: 'None' | Expected: TypeError | Got: TypeError | Status: Correct ✓
2. Fahrenheit to Celsius
Input: 32°F | Expected: 0.0°C | Got: 0.00°C | Status: Correct ✓
Input: 212°F | Expected: 100.0°C | Got: 100.00°C | Status: Correct ✓
Input: -40°F | Expected: -40.0°C | Got: -40.00°C | Status: Correct ✓
Input: 98.6°F | Expected: 37.0°C | Got: 37.00°C | Status: Correct ✓
Input: 'a string' | Expected: TypeError | Got: TypeError | Status: Correct ✓
Input: 'None' | Expected: TypeError | Got: TypeError | Status: Correct ✓
--- All tests complete ---
PS C:\Users\kamer> []
```

Ln 32, Col 1 Spaces: 4 UTF-8 Python Signed out 3.13.5 (base)

TASK 3

Task Description:

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

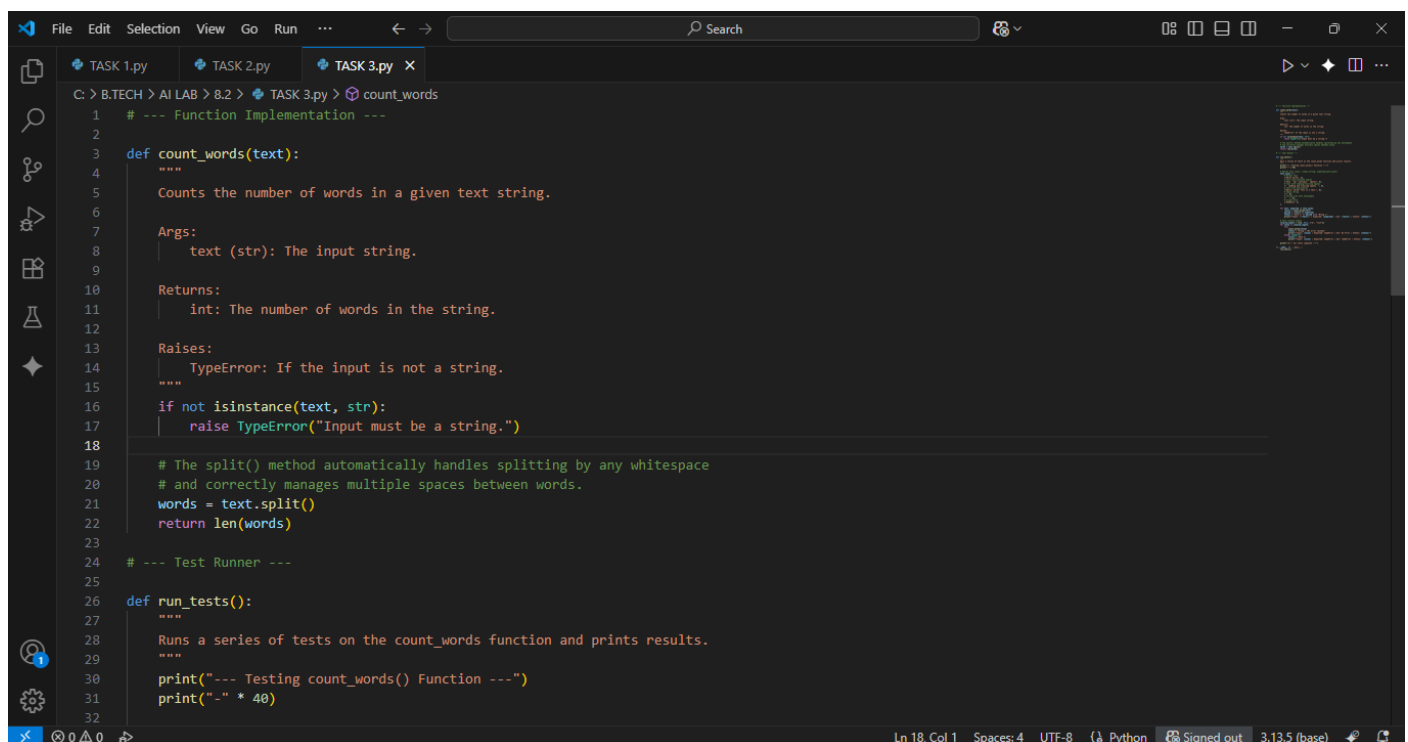
Requirements

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

PROMPT:

Please write a Python function `count_words(text)` and a corresponding suite of test cases to validate it.

CODE:

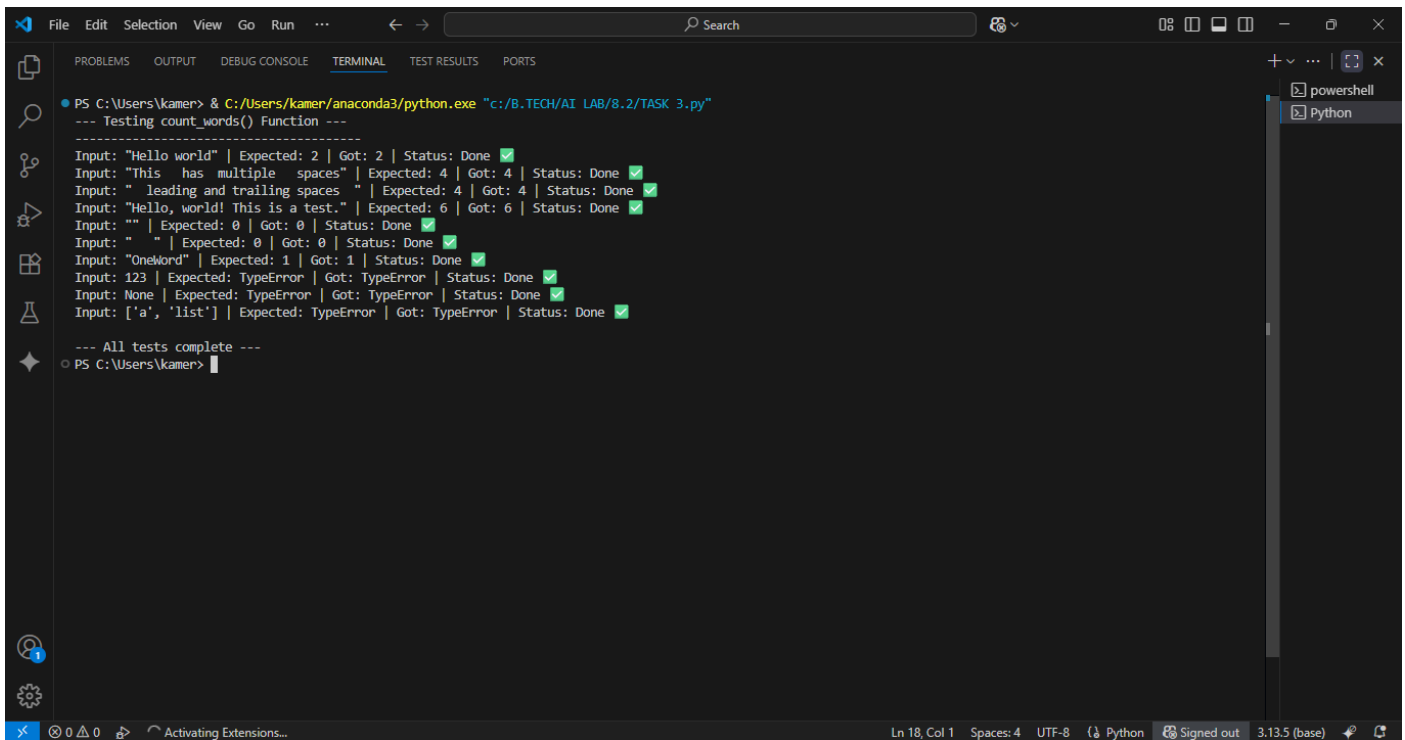


```
File Edit Selection View Go Run ... Search
TASK 1.py TASK 2.py TASK 3.py x
C: > B.TECH > AI LAB > 8.2 > TASK 3.py > count_words
1 # --- Function Implementation ---
2
3 def count_words(text):
4     """
5     Counts the number of words in a given text string.
6
7     Args:
8         text (str): The input string.
9
10    Returns:
11        int: The number of words in the string.
12
13    Raises:
14        TypeError: If the input is not a string.
15    """
16    if not isinstance(text, str):
17        raise TypeError("Input must be a string.")
18
19    # The split() method automatically handles splitting by any whitespace
20    # and correctly manages multiple spaces between words.
21    words = text.split()
22    return len(words)
23
24 # --- Test Runner ---
25
26 def run_tests():
27     """
28     Runs a series of tests on the count_words function and prints results.
29
30     print("--- Testing count_words() Function ---")
31     print("-" * 40)
32
33 Ln 18, Col 1 Spaces: 4 UTF-8 Python Signed out 3.13.5 (base)
```

```
File Edit Selection View Go Run ... Search
TASK 1.py TASK 2.py TASK 3.py x
C: > B.TECH > AI LAB > 8.2 > TASK 3.py > count_words
26 def run_tests():
33     # Define test cases: (input_string, expected_word_count)
34     test_cases = [
35         # Normal text
36         ("Hello world", 2),
37         # Text with multiple spaces
38         ("This has multiple spaces", 4),
39         # Text with leading/trailing spaces
40         (" leading and trailing spaces ", 4),
41         # Text with punctuation
42         ("Hello, world! This is a test.", 6),
43         # Empty string
44         ("", 0),
45         # String with only whitespace
46         (" ", 0),
47         # Single word
48         ("OneWord", 1)
49     ]
51     for text, expected in test_cases:
52         result = count_words(text)
53         passed = (result == expected)
54         status = "Done ✓" if passed else "Wrong ✗"
55         print(f"Input: \"{text}\" | Expected: {expected} | Got: {result} | Status: {status}")
56
57     # Test invalid inputs
58     invalid_inputs = [123, None, ["a", "list"]]
59     for value in invalid_inputs:
60         try:
61             count_words(value)
62             status = "Wrong ✗ (No error raised)"
63             print(f"Input: {value} | Expected: TypeError | Got: No Error | Status: {status}")
64
65     print("\n--- All tests complete ---")
66
67 if __name__ == '__main__':
68     run_tests()
69
70
71
72
73
Ln 18, Col 1 Spaces: 4 UTF-8 Python Signed out 3.13.5 (base)
```

```
File Edit Selection View Go Run ... Search
TASK 1.py TASK 2.py TASK 3.py x
C: > B.TECH > AI LAB > 8.2 > TASK 3.py > count_words
26 def run_tests():
27
28     # Test invalid inputs
29     invalid_inputs = [123, None, ["a", "list"]]
30     for value in invalid_inputs:
31         try:
32             count_words(value)
33             status = "Wrong ✗ (No error raised)"
34             print(f"Input: {value} | Expected: TypeError | Got: No Error | Status: {status}")
35         except TypeError:
36             status = "Done ✓"
37             print(f"Input: {value} | Expected: TypeError | Got: TypeError | Status: {status}")
38
39     print("\n--- All tests complete ---")
40
41 if __name__ == '__main__':
42     run_tests()
43
44
45
46
47
Ln 18, Col 1 Spaces: 4 UTF-8 Python Signed out 3.13.5 (base)
```

OUTPUT:



The screenshot shows a VS Code terminal window with the following content:

```
PS C:\Users\kamer> & C:/Users/kamer/anaconda3/python.exe "c:/B.TECH/AI LAB/8.2/TASK 3.py"
--- Testing count_words() Function ---
-----
Input: "Hello world" | Expected: 2 | Got: 2 | Status: Done ✓
Input: "This has multiple spaces" | Expected: 4 | Got: 4 | Status: Done ✓
Input: " leading and trailing spaces " | Expected: 4 | Got: 4 | Status: Done ✓
Input: "Hello, world! This is a test." | Expected: 6 | Got: 6 | Status: Done ✓
Input: "" | Expected: 0 | Got: 0 | Status: Done ✓
Input: " " | Expected: 0 | Got: 0 | Status: Done ✓
Input: "OneWord" | Expected: 1 | Got: 1 | Status: Done ✓
Input: 123 | Expected: TypeError | Got: TypeError | Status: Done ✓
Input: None | Expected: TypeError | Got: TypeError | Status: Done ✓
Input: ['a', 'list'] | Expected: TypeError | Got: TypeError | Status: Done ✓

--- All tests complete ---
PS C:\Users\kamer>
```

TASK 4

Task Description:

- 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:

Please write a Python BankAccount class and a corresponding suite of test cases to validate its functionality.

CODE:


```
File Edit Selection View Go Run ... Search
TASK 1.py TASK 2.py TASK 3.py TASK 4 TASK 4.py X
C:\B.TECH > AI LAB > 8.2 > TASK 4.py > ...
1 # --- Class Implementation ---
2
3 class BankAccount:
4     """
5     A simple bank account class that supports depositing, withdrawing,
6     and checking the balance.
7     """
8     def __init__(self, initial_balance=0):
9         if not isinstance(initial_balance, (int, float)) or initial_balance < 0:
10             raise ValueError("Initial balance cannot be negative or non-numeric.")
11         self._balance = initial_balance
12
13     def deposit(self, amount):
14         """
15         Deposits a positive amount into the account.
16         Raises ValueError for non-positive or non-numeric amounts.
17         """
18         if not isinstance(amount, (int, float)):
19             raise TypeError("Deposit amount must be a number.")
20         if amount <= 0:
21             raise ValueError("Deposit amount must be positive.")
22         self._balance += amount
23
24     def withdraw(self, amount):
25         """
26         Withdraws a positive amount from the account.
27         Raises ValueError for non-positive amounts or insufficient funds.
28         """
29         if not isinstance(amount, (int, float)):
30             raise TypeError("Withdrawal amount must be a number.")
31         if amount <= 0:
32             raise ValueError("Withdrawal amount must be positive.")
```

```
File Edit Selection View Go Run ... Search
TASK 1.py TASK 2.py TASK 3.py TASK 4 TASK 4.py X
C:\B.TECH > AI LAB > 8.2 > TASK 4.py > ...
3 class BankAccount:
24     def withdraw(self, amount):
33         if amount > self._balance:
34             raise ValueError("Insufficient funds for this withdrawal.")
35         self._balance -= amount
36
37     def check_balance(self):
38         """Returns the current account balance."""
39         return self._balance
40
41 # --- Test Runner ---
42
43 def run_tests():
44     """
45     Runs a series of tests on the BankAccount class and prints results.
46     """
47     print("--- Testing BankAccount() Class ---")
48     print("-" * 45)
49
50     # Test 1: Initial state and valid transactions
51     print("1. Testing Valid Transactions")
52     print("-" * 45)
53     try:
54         account = BankAccount(100)
55         balance = account.check_balance()
56         passed = (balance == 100)
57         status = "Done ✅" if passed else "Wrong ❌"
58         print(f"Action: Create with 100 | Expected: 100 | Got: {balance} | Status: {status}")
59
60         account.deposit(50)
61         balance = account.check_balance()
62         passed = (balance == 150)
```

```
File Edit Selection View Go Run ... Search
TASK 1.py TASK 2.py TASK 3.py TASK 4 TASK 4.py X
C:\B.TECH > AI LAB > 8.2 > TASK 4.py > ...
43 def run_tests():
44     ...
60     account.deposit(50)
61     balance = account.check_balance()
62     passed = (balance == 150)
63     status = "Done ✓" if passed else "Wrong ✗"
64     print(f"Action: Deposit 50 | Expected: 150 | Got: {balance} | Status: {status}")
65
66     account.withdraw(75)
67     balance = account.check_balance()
68     passed = (balance == 75)
69     status = "Done ✓" if passed else "Wrong ✗"
70     print(f"Action: Withdraw 75 | Expected: 75 | Got: {balance} | Status: {status}")
71
72 except Exception as e:
73     print(f"An unexpected error occurred during valid tests: {e}")
74
75 # Test 2: Error handling
76 print("\n" + "-" * 45)
77 print("2. Testing Error Handling")
78 print("-" * 45)
79 account = BankAccount(100) # Reset account for error tests
80
81 # Dictionary of tests: { action_description: (function_to_call, expected_error) }
82 error_tests = {
83     "Deposit negative amount": (lambda: account.deposit(-50), ValueError),
84     "Deposit zero": (lambda: account.deposit(0), ValueError),
85     "Withdraw negative amount": (lambda: account.withdraw(-20), ValueError),
86     "Withdraw more than balance": (lambda: account.withdraw(101), ValueError),
87     "Deposit non-numeric type": (lambda: account.deposit("abc"), TypeError),
88 }
89
90 ...
```

```
File Edit Selection View Go Run ... Search
TASK 1.py TASK 2.py TASK 3.py TASK 4 TASK 4.py X
C:\B.TECH > AI LAB > 8.2 > TASK 4.py > ...
43 def run_tests():
44     ...
81 # Dictionary of tests: { action_description: (function_to_call, expected_error) }
82 error_tests = {
83     "Deposit negative amount": (lambda: account.deposit(-50), ValueError),
84     "Deposit zero": (lambda: account.deposit(0), ValueError),
85     "Withdraw negative amount": (lambda: account.withdraw(-20), ValueError),
86     "Withdraw more than balance": (lambda: account.withdraw(101), ValueError),
87     "Deposit non-numeric type": (lambda: account.deposit("abc"), TypeError),
88 }
89
90 for desc, (action, expected_error) in error_tests.items():
91     try:
92         action()
93         status = "Wrong ✗ (No error raised)"
94         print(f"Action: {desc:<25} | Expected: {expected_error.__name__:<10} | Got: No Error | Status: {status}")
95     except Exception as e:
96         passed = isinstance(e, expected_error)
97         status = "Done ✓" if passed else f"Wrong ✗ (Got {type(e).__name__})"
98         print(f"Action: {desc:<25} | Expected: {expected_error.__name__:<10} | Got: {type(e).__name__} | Status: {status}")
99
100 print("\n--- All tests complete ---")
101
102 if __name__ == '__main__':
103     run_tests()
104
```

OUTPUT:

The screenshot shows a VS Code terminal window with the following content:

```
PS C:\Users\kamer> & C:/Users/kamer/anaconda3/python.exe "c:/B.TECH/AI LAB/8.2/TASK 4.py"
--- Testing BankAccount() Class ---
-----
1. Testing Valid Transactions
-----
Action: Create with 100 | Expected: 100 | Got: 100 | Status: Done ✓
Action: Deposit 50 | Expected: 150 | Got: 150 | Status: Done ✓
Action: Withdraw 75 | Expected: 75 | Got: 75 | Status: Done ✓
-----
2. Testing Error Handling
-----
Action: Deposit negative amount | Expected: ValueError | Got: ValueError | Status: Done ✓
Action: Deposit zero | Expected: ValueError | Got: ValueError | Status: Done ✓
Action: Withdraw negative amount | Expected: ValueError | Got: ValueError | Status: Done ✓
Action: Withdraw more than balance | Expected: ValueError | Got: ValueError | Status: Done ✓
Action: Deposit non-numeric type | Expected: TypeError | Got: TypeError | Status: Done ✓
-----
--- All tests complete ---
PS C:\Users\kamer>
```

TASK 5

Task Description:

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:

Please write a Python function `is_number_palindrome(num)` and a corresponding suite of test cases to validate it.

CODE:

```
File Edit Selection View Go Run ... Search
TASK 1.py TASK 2.py TASK 3.py TASK 4 TASK 4.py TASK 5.py x
C:\B.TECH > AI LAB > 8.2 > TASK 5.py > ...
1 # --- Function Implementation ---
2
3 def is_number_palindrome(num):
4     """
5     Checks if an integer reads the same forwards and backward.
6
7     Args:
8         num (int): The integer to check.
9
10    Returns:
11        bool: True if the number is a palindrome, False otherwise.
12
13    Raises:
14        TypeError: If the input is not an integer.
15    """
16    if not isinstance(num, int):
17        raise TypeError("Input must be an integer.")
18
19    # Negative numbers are not considered palindromes.
20    if num < 0:
21        return False
22
23    # Convert the number to a string and check if it equals its reverse.
24    return str(num) == str(num)[::-1]
25
26 # --- Test Runner ---
27
28 def run_tests():
29     """
30     Runs a series of tests on the is_number_palindrome function and prints results.
31     """
32     print("--- Testing is_number_palindrome() Function ---")
```

```
File Edit Selection View Go Run ... Search
TASK 1.py TASK 2.py TASK 3.py TASK 4 TASK 4.py TASK 5.py x
C:\B.TECH > AI LAB > 8.2 > TASK 5.py > ...
28 def run_tests():
29     """
30     Runs a series of tests on the is_number_palindrome function and prints results.
31     """
32     print("--- Testing is_number_palindrome() Function ---")
33     print("-" * 55)
34
35     # Define test cases: (input_number, expected_boolean)
36     test_cases = [
37         (121, True),
38         (123, False),
39         (0, True),
40         (7, True),
41         (-121, False),
42         (1221, True),
43         (10, False),
44         (123454321, True),
45     ]
46
47     for num, expected in test_cases:
48         result = is_number_palindrome(num)
49         passed = (result == expected)
50         status = "Done ✓" if passed else "Wrong ✗"
51         # Using f-string alignment for clean columns
52         print(f"Input: {num:<10} | Expected: {str(expected):<5} | Got: {str(result):<5} | Status: {status}")
53
54     # Test invalid inputs
55     print("\n--- Testing Invalid Inputs ---")
56     print("-" * 55)
57     invalid_inputs = [12.1, "121", None]
58     for value in invalid_inputs:
59         try:
```

```
File Edit Selection View Go Run ... Search
TASK 1.py TASK 2.py TASK 3.py TASK 4 TASK 4.py TASK 5.py x
C: > B.TECH > AI LAB > 8.2 > TASK 5.py > ...
28 def run_tests():
54     # Test invalid inputs
55     print("\n--- Testing Invalid Inputs ---")
56     print("-" * 55)
57     invalid_inputs = [12.1, "121", None]
58     for value in invalid_inputs:
59         try:
60             is_number_palindrome(value)
61             status = "Wrong ❌ (No error raised)"
62             print(f"Input: {str(value):<10} | Expected: TypeError | Got: No Error | Status: {status}")
63         except TypeError:
64             status = "Done ✅"
65             print(f"Input: {str(value):<10} | Expected: TypeError | Got: TypeError | Status: {status}")
66
67     print("\n--- All tests complete ---")
68
69 if __name__ == '__main__':
70     run_tests()
```

OUTPUT:

```
File Edit Selection View Go Run ... Search
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS
Input: 1221 | Expected: True | Got: True | Status: Done ✅
PS C:\Users\kamer> & C:\Users\kamer\anaconda3\python.exe "c:/B.TECH/AI LAB/8.2/TASK 5.py"
--- Testing is_number_palindrome() Function ---
-----
Input: 121 | Expected: True | Got: True | Status: Done ✅
Input: 123 | Expected: False | Got: False | Status: Done ✅
Input: 0 | Expected: True | Got: True | Status: Done ✅
Input: 7 | Expected: True | Got: True | Status: Done ✅
Input: -121 | Expected: False | Got: False | Status: Done ✅
Input: 1221 | Expected: True | Got: True | Status: Done ✅
Input: 10 | Expected: False | Got: False | Status: Done ✅
Input: 123454321 | Expected: True | Got: True | Status: Done ✅
--- Testing Invalid Inputs ---
-----
Input: 12.1 | Expected: TypeError | Got: TypeError | Status: Done ✅
Input: 121 | Expected: TypeError | Got: TypeError | Status: Done ✅
Input: None | Expected: TypeError | Got: TypeError | Status: Done ✅
--- All tests complete ---
PS C:\Users\kamer>
```

OBSERVATION:

I Observed that Gemini AI can easily generate the programs correctly from the given prompt and Gemini AI Provides a Detailed Explanation and Debugging. Gemini AI is a fascinating tool to observe—especially in how it transforms the developer experience.