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:

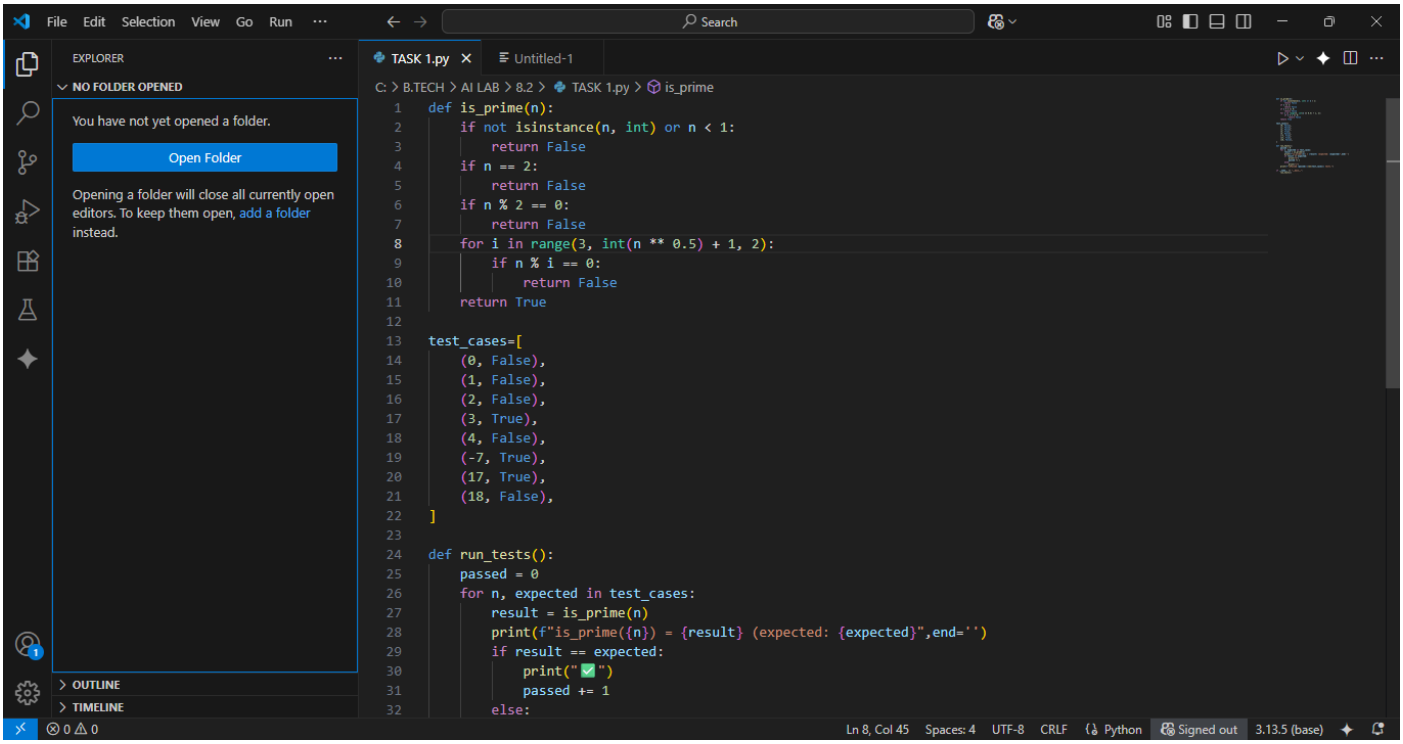
• Only integers > 1 can be prime.

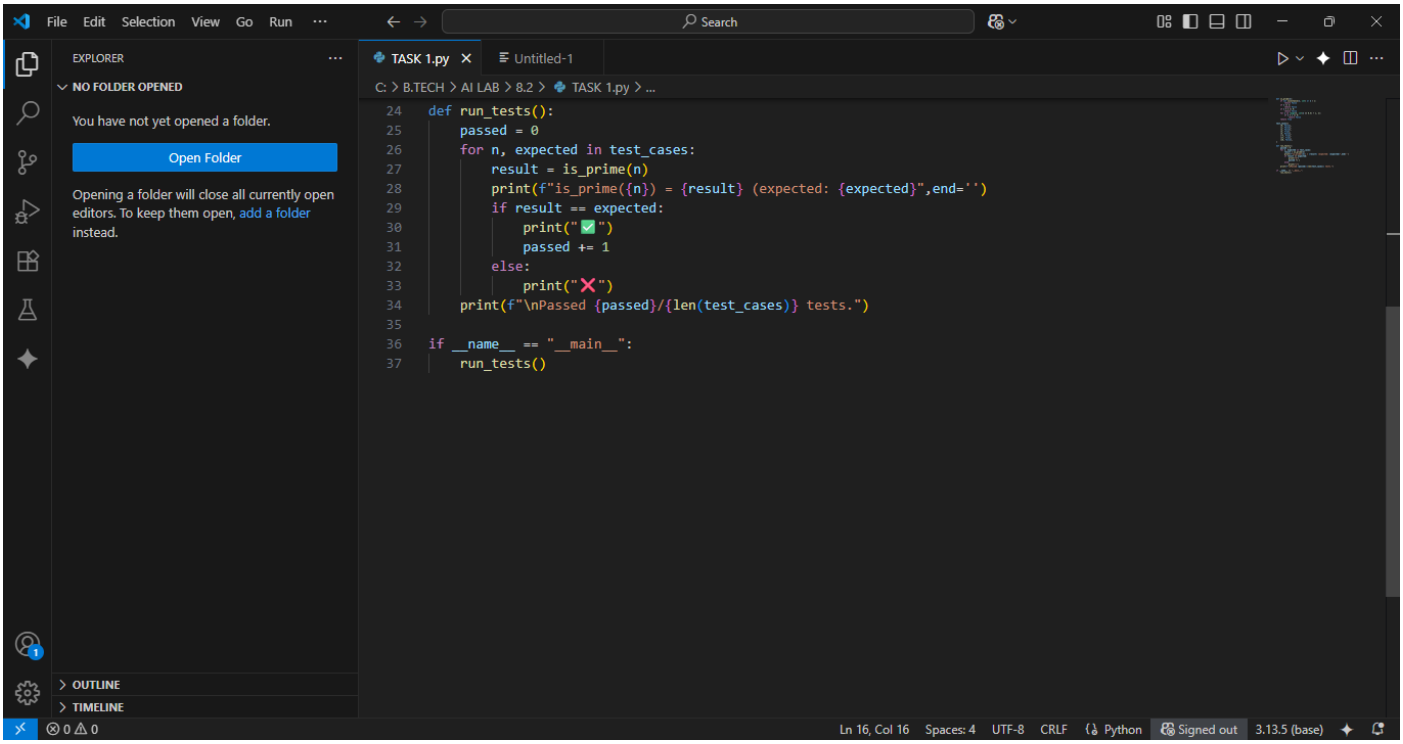
• Check edge cases: 0, 1, 2, negative numbers, and large primes.

PROMPT:

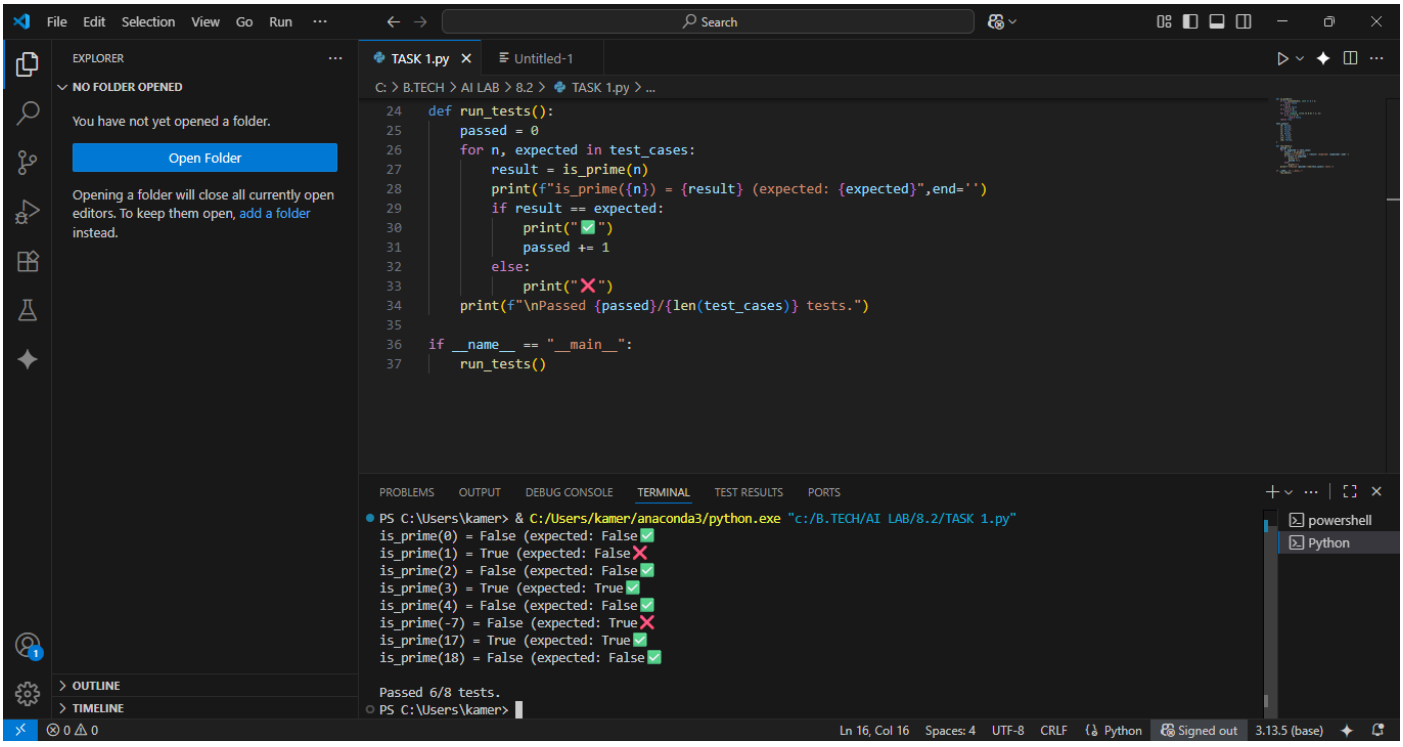
Please write a Python function is_prime(n) and a corresponding suite of test cases to validate it. The function should be implemented first, followed by the tests.

CODE:




File Edit Selection View Go Run
Search
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
TASK 1.py
Untitled-1
C: > B.TECH > AI LAB > 8.2 > TASK 1.py > ...
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:
33 print("❌")
34 print(f"\nPassed {passed}/{len(test_cases)} tests.")
35
36 if __name__ == "__main__":
37 run_tests()
OUTLINE
TIMELINE
Ln 16, Col 16 Spaces: 4 UTF-8 CRLF Python Signed out 3.13.5 (base)

OUTPUT:


File Edit Selection View Go Run
Search
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
TASK 1.py
Untitled-1
C: > B.TECH > AI LAB > 8.2 > TASK 1.py > ...
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:
33 print("❌")
34 print(f"\nPassed {passed}/{len(test_cases)} tests.")
35
36 if __name__ == "__main__":
37 run_tests()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS
PS C:\Users\kamer> & C:/Users/kamer/anaconda3/python.exe "c:/B.TECH/AI LAB/8.2/TASK 1.py"
is_prime(0) = False (expected: False✅
is_prime(1) = True (expected: False❌
is_prime(2) = False (expected: False✅
is_prime(3) = True (expected: True✅
is_prime(4) = False (expected: False✅
is_prime(-7) = False (expected: True❌
is_prime(17) = True (expected: True✅
is_prime(18) = False (expected: False✅
Passed 6/8 tests.
PS C:\Users\kamer>
powershell
Python
OUTLINE
TIMELINE
Ln 16, Col 16 Spaces: 4 UTF-8 CRLF Python Signed out 3.13.5 (base)

TASK 2

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

Requirements:

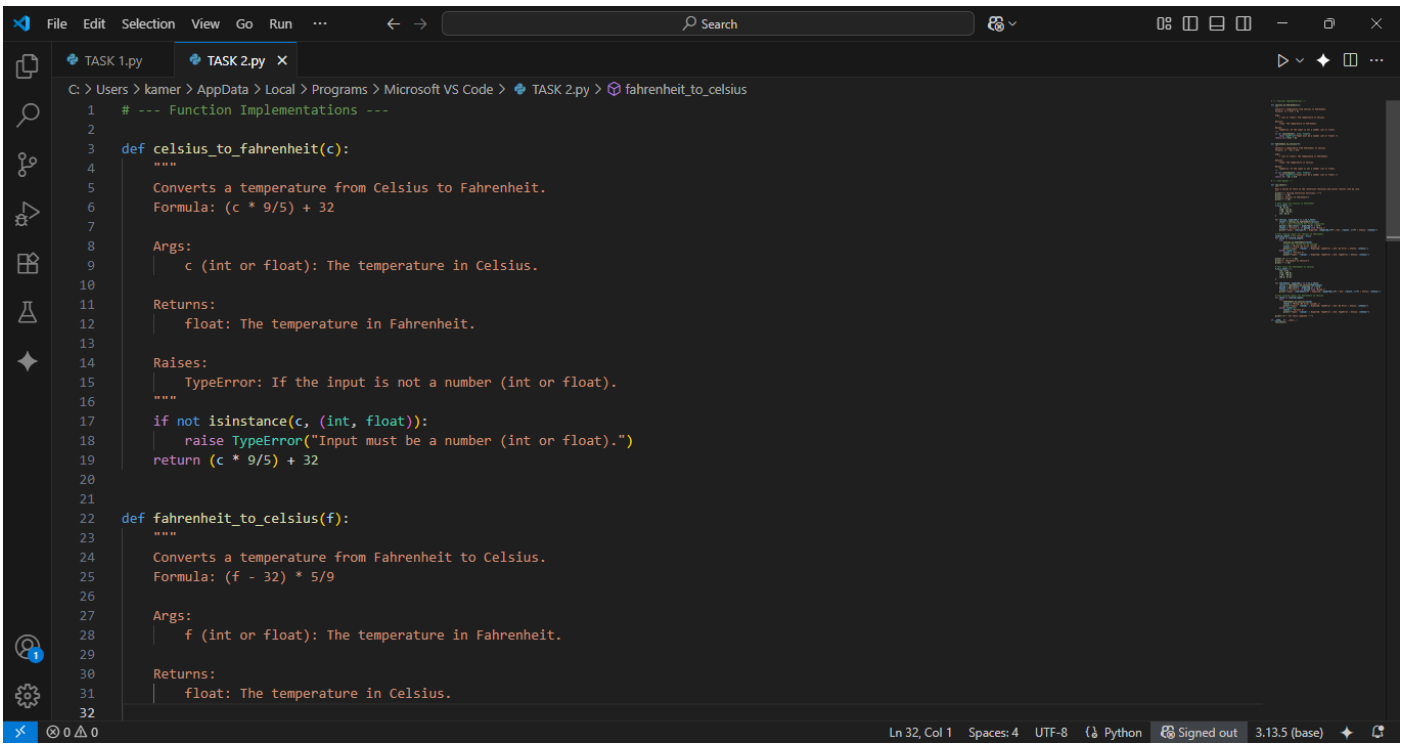
• Validate known pairs: 0°C = 32°F, 100°C = 212°F.

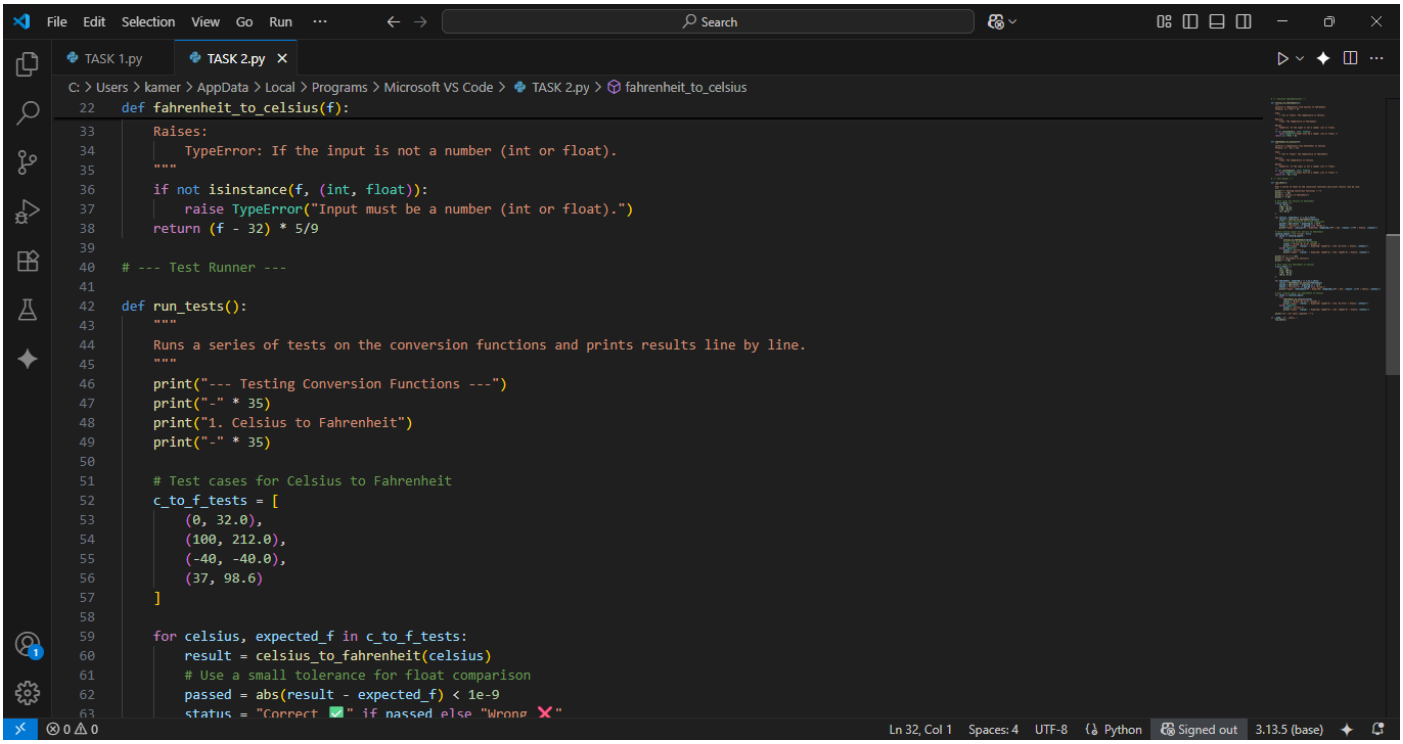
• Include decimals and invalid inputs like strings or None.

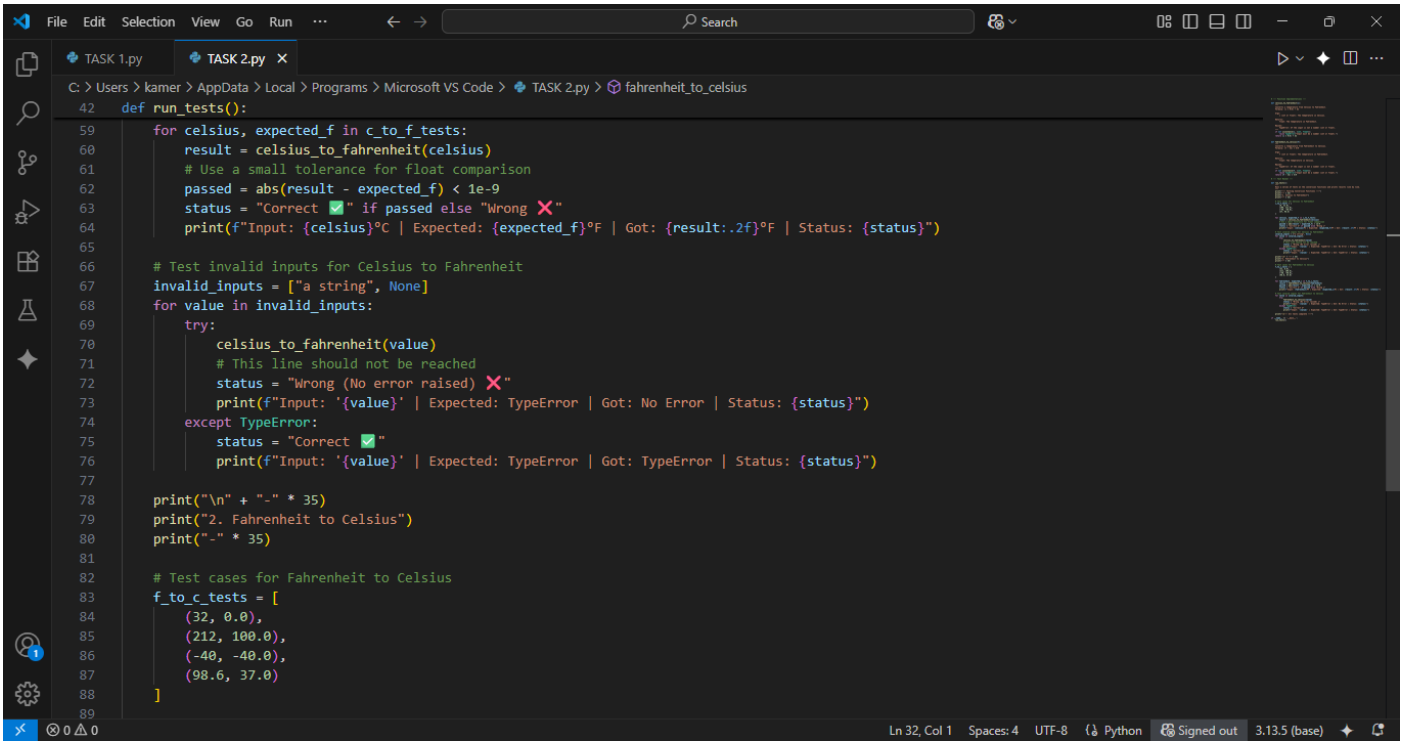
PROMPT:

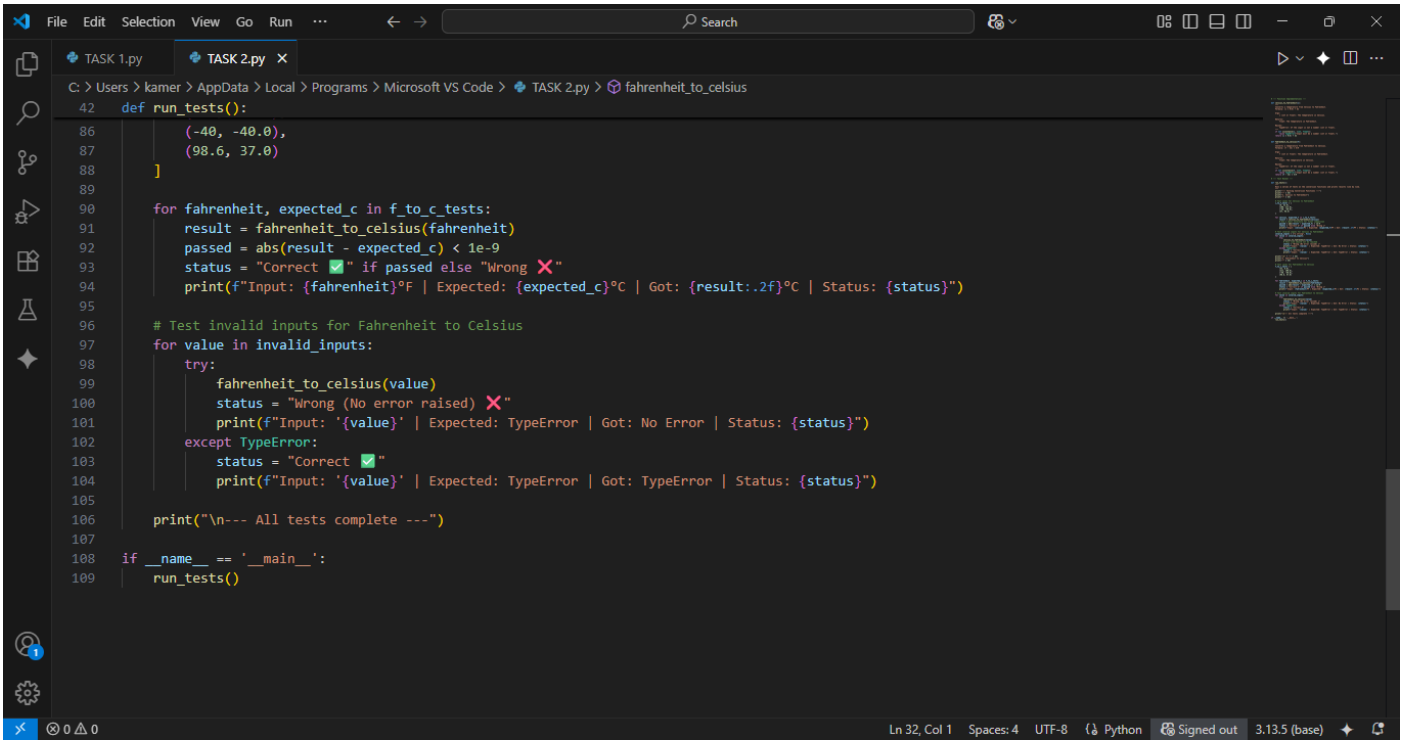
Please write two Python functions for temperature conversion, celsius_to_fahrenheit(c) and fahrenheit_to_celsius(f), and a corresponding suite of test cases to validate them.

CODE:

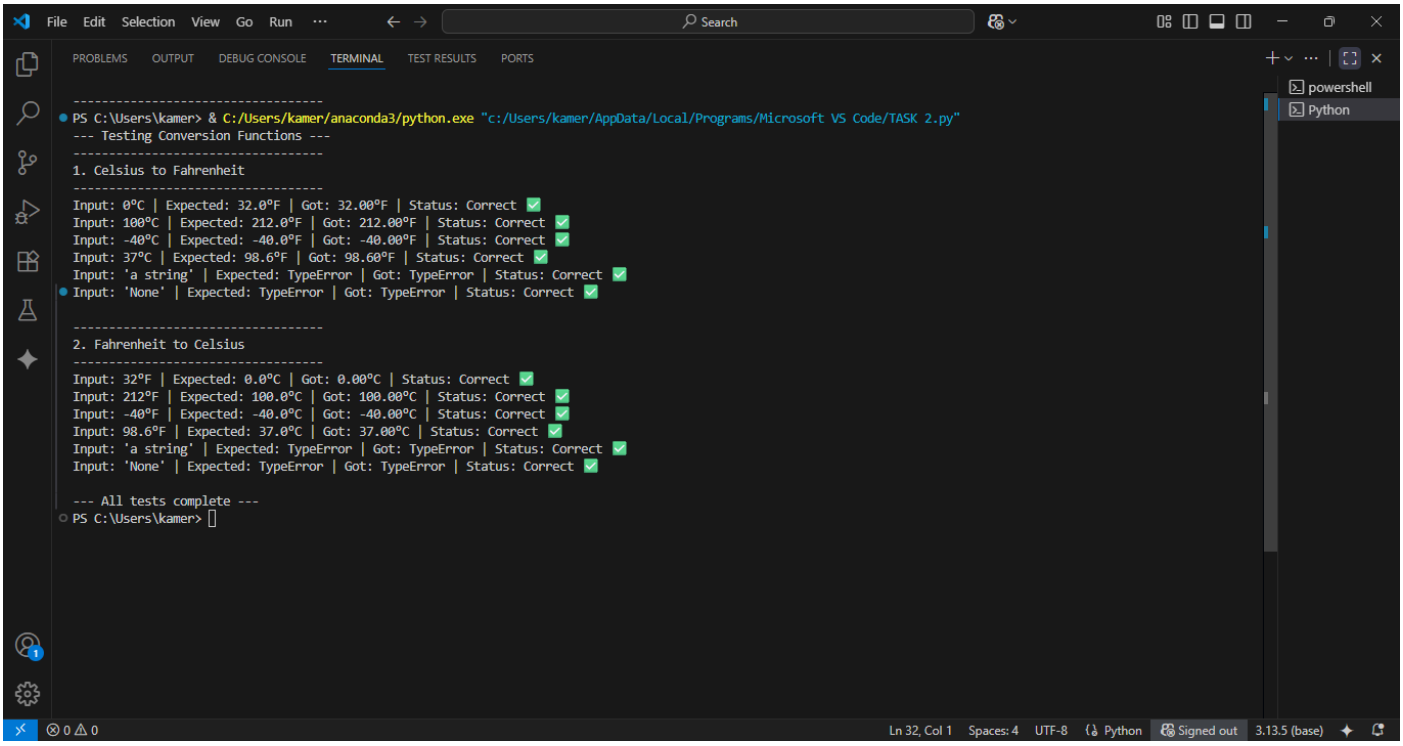



File Edit Selection View Go Run
Search
TASK 1.py
TASK 2.py
C: > Users > kamer > AppData > Local > Programs > Microsoft VS Code > TASK 2.py > fahrenheit_to_celsius
def fahrenheit_to_celsius(f):
Raises:
TypeError: If the input is not a number (int or float).
"""
if not isinstance(f, (int, float)):
raise TypeError("Input must be a number (int or float).")
return (f - 32) * 5/9
# --- Test Runner ---
def run_tests():
"""
Runs a series of tests on the conversion functions and prints results line by line.
"""
print("--- Testing Conversion Functions ---")
print("-" * 35)
print("1. Celsius to Fahrenheit")
print("-" * 35)
# Test cases for Celsius to Fahrenheit
c_to_f_tests = [
(0, 32.0),
(100, 212.0),
(-40, -40.0),
(37, 98.6)
]
for celsius, expected_f in c_to_f_tests:
result = celsius_to_fahrenheit(celsius)
# Use a small tolerance for float comparison
passed = abs(result - expected_f) < 1e-9
status = "Correct ✅" if passed else "Wrong ❌"
Ln 32, 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
C: > Users > kamer > AppData > Local > Programs > Microsoft VS Code > TASK 2.py > fahrenheit_to_celsius
def run_tests():
for celsius, expected_f in c_to_f_tests:
result = celsius_to_fahrenheit(celsius)
# Use a small tolerance for float comparison
passed = abs(result - expected_f) < 1e-9
status = "Correct ✅" if passed else "Wrong ❌"
print(f"Input: {celsius}°C | Expected: {expected_f}°F | Got: {result:.2f}°F | Status: {status}")
# Test invalid inputs for Celsius to Fahrenheit
invalid_inputs = ["a string", None]
for value in invalid_inputs:
try:
celsius_to_fahrenheit(value)
# This line should not be reached
status = "Wrong (No error raised) ❌"
print(f"Input: '{value}' | Expected: TypeError | Got: No Error | Status: {status}")
except TypeError:
status = "Correct ✅"
print(f"Input: '{value}' | Expected: TypeError | Got: TypeError | Status: {status}")
print("\n" + "-" * 35)
print("2. Fahrenheit to Celsius")
print("-" * 35)
# Test cases for Fahrenheit to Celsius
f_to_c_tests = [
(32, 0.0),
(212, 100.0),
(-40, -40.0),
(98.6, 37.0)
]
Ln 32, 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
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
powershell
Python
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.

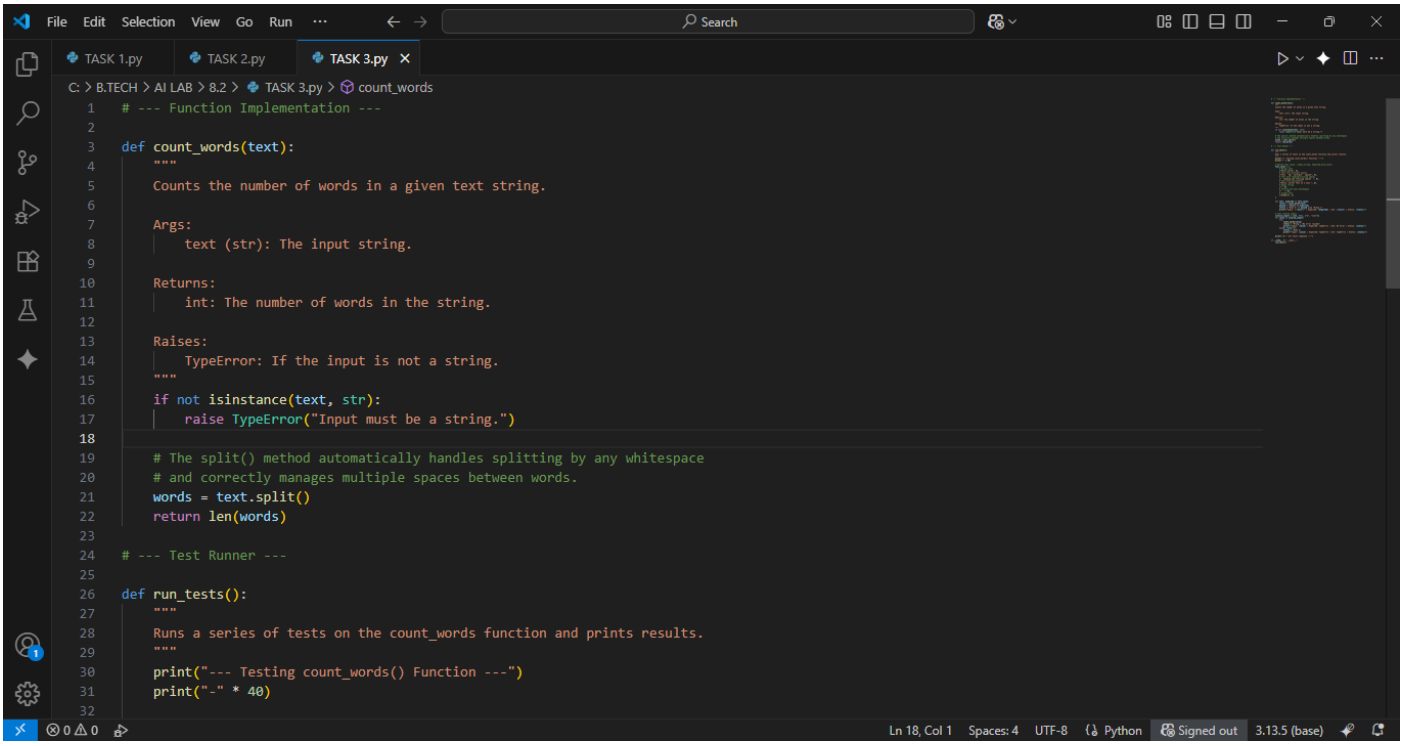
Requirements

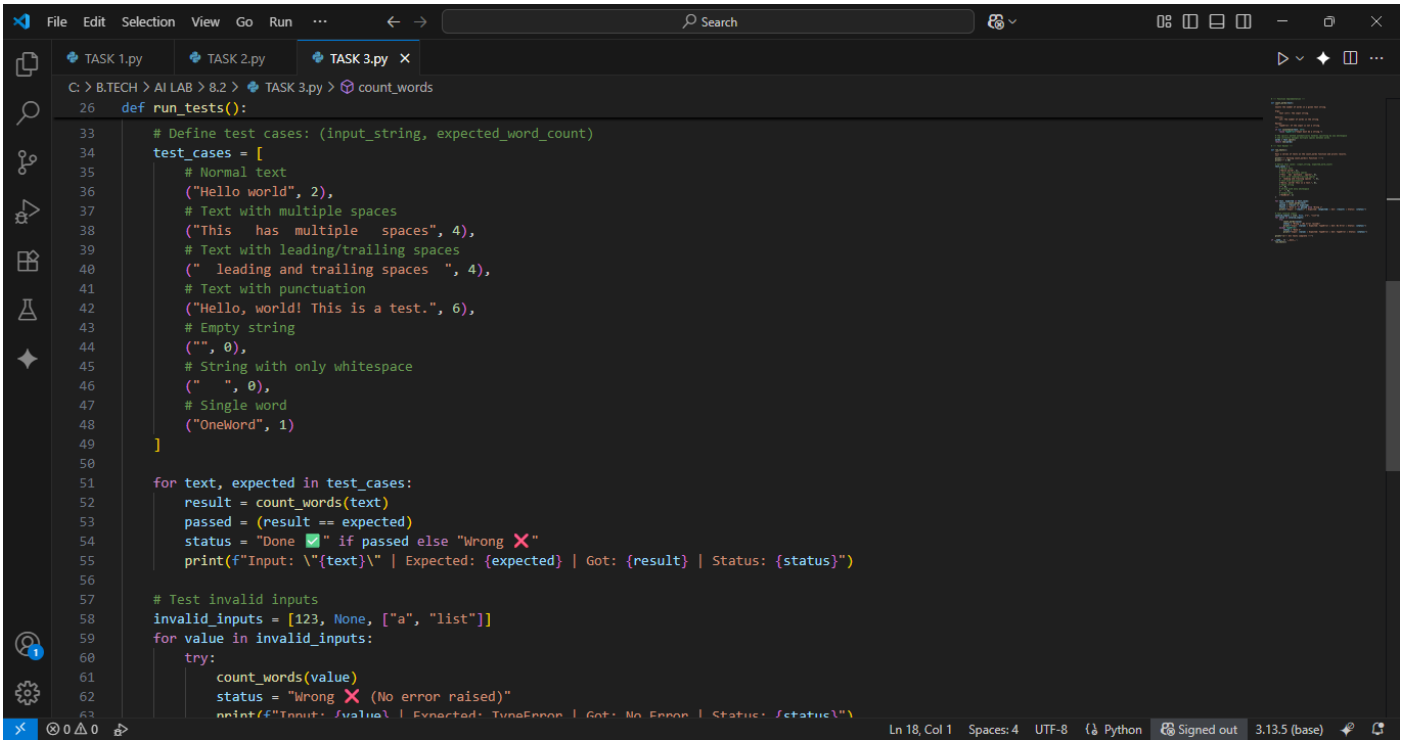
Handle normal text, multiple spaces, punctuation, and empty strings.

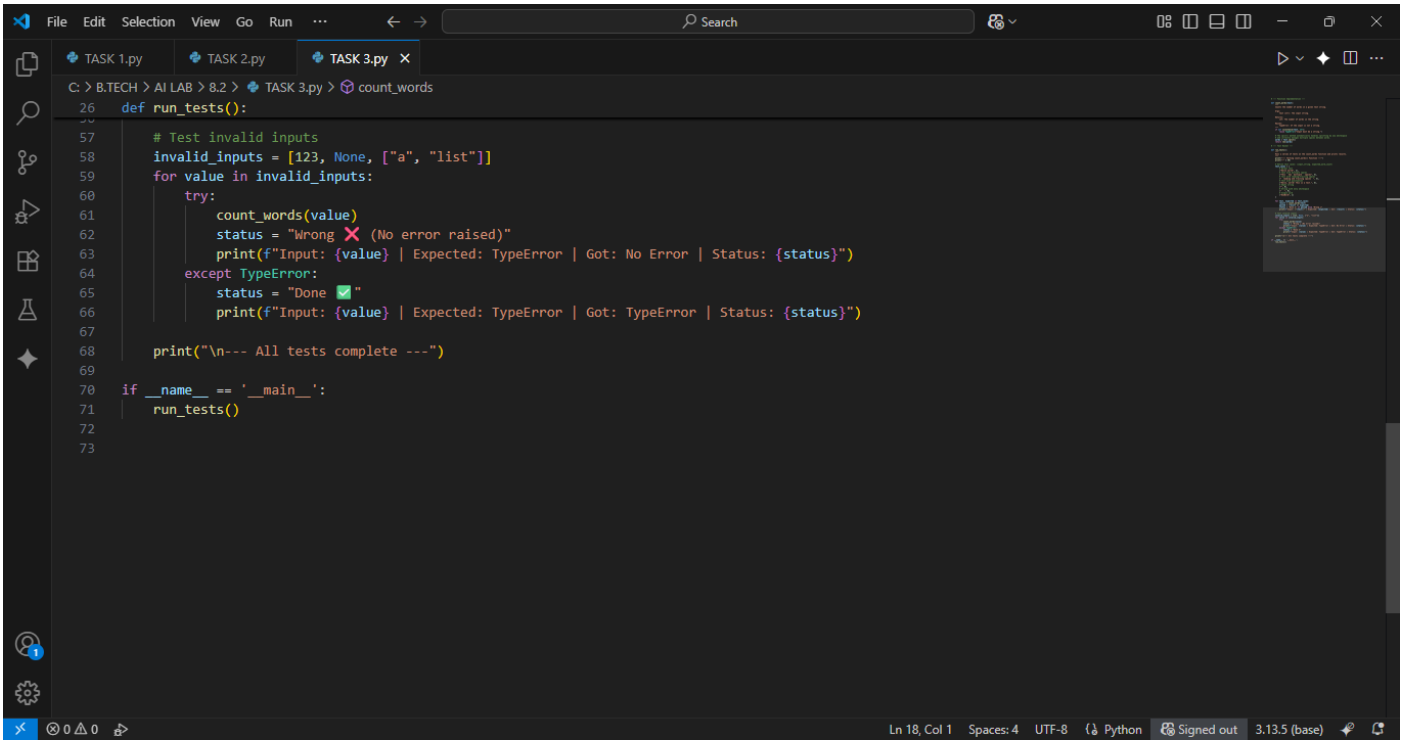
PROMPT:

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

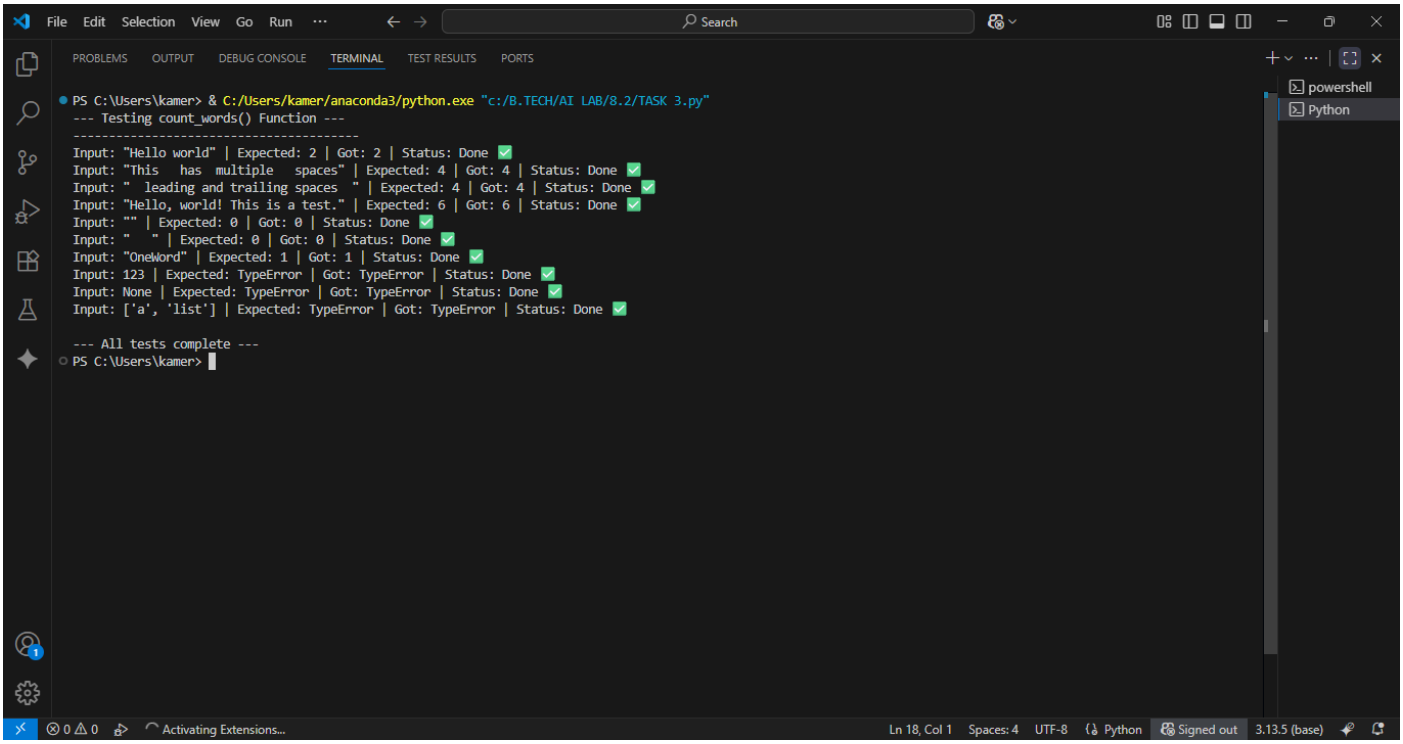
CODE:




File Edit Selection View Go Run
Search
TASK 1.py
TASK 2.py
TASK 3.py
C: > B.TECH > AI LAB > 8.2 > TASK 3.py > count_words
def run_tests():
# Define test cases: (input_string, expected_word_count)
test_cases = [
# Normal text
("Hello world", 2),
Text with multiple spaces
("This has multiple spaces", 4),
Text with leading/trailing spaces
(" leading and trailing spaces ", 4),
Text with punctuation
("Hello, world! This is a test.", 6),
Empty string
("", 0),
String with only whitespace
(" ", 0),
Single word
("OneWord", 1)
]
for text, expected in test_cases:
result = count_words(text)
passed = (result == expected)
status = "Done ✅" if passed else "Wrong ❌"
print(f"Input: \"{text}\" | Expected: {expected} | Got: {result} | Status: {status}")
# Test invalid inputs
invalid_inputs = [123, None, ["a", "list"]]
for value in invalid_inputs:
try:
count_words(value)
status = "Wrong ❌ (No error raised)"
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
C: > B.TECH > AI LAB > 8.2 > TASK 3.py > count_words
def run_tests():
# Test invalid inputs
invalid_inputs = [123, None, ["a", "list"]]
for value in invalid_inputs:
try:
count_words(value)
status = "Wrong ❌ (No error raised)"
print(f"Input: {value} | Expected: TypeError | Got: No Error | Status: {status}")
except TypeError:
status = "Done ✅"
print(f"Input: {value} | Expected: TypeError | Got: TypeError | Status: {status}")
print("\n--- All tests complete ---")
if __name__ == '__main__':
run_tests()
Ln 18, Col 1 Spaces: 4 UTF-8 Python Signed out 3.13.5 (base)

OUTPUT:



TASK 4

Task Description:

• Generate test cases for a BankAccount class with:

Methods:

deposit(amount)

withdraw(amount)

check_balance()

Requirements:

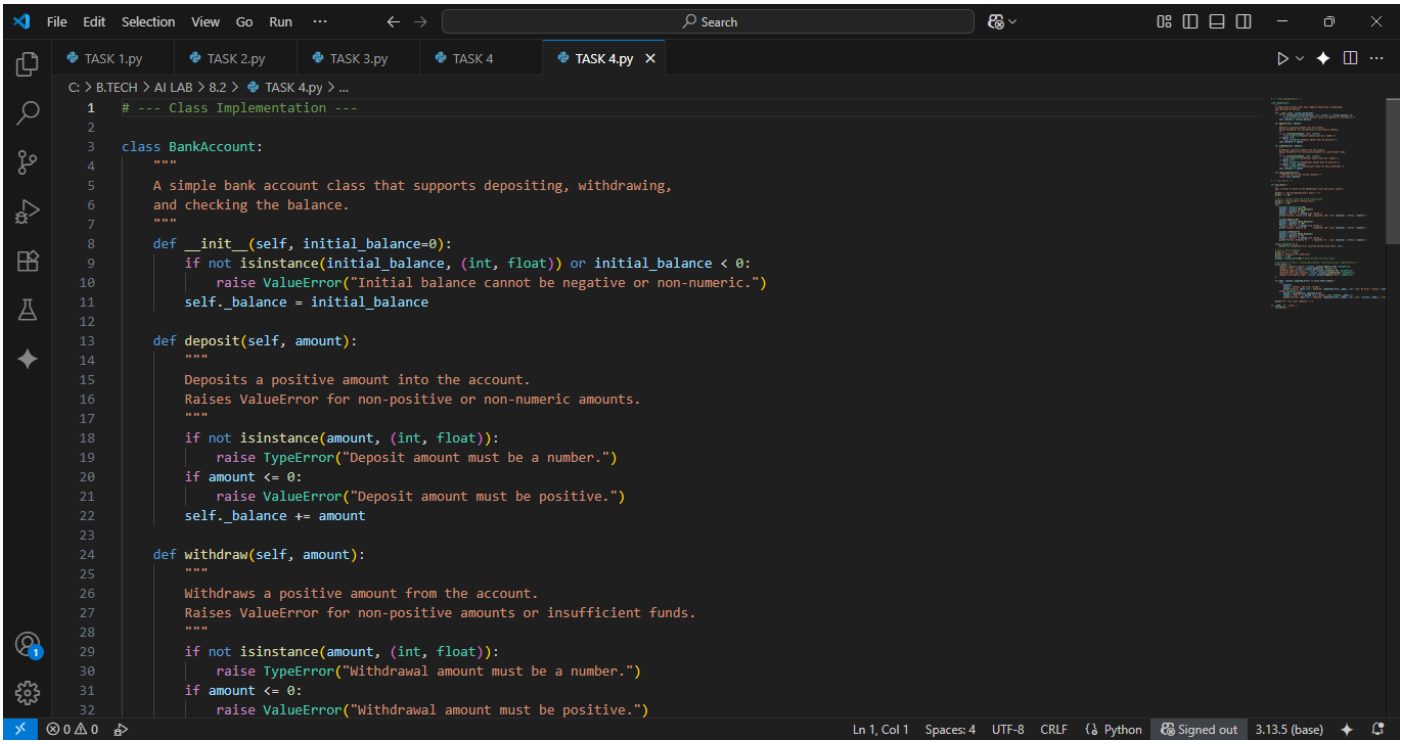
• Negative deposits/withdrawals should raise an error.

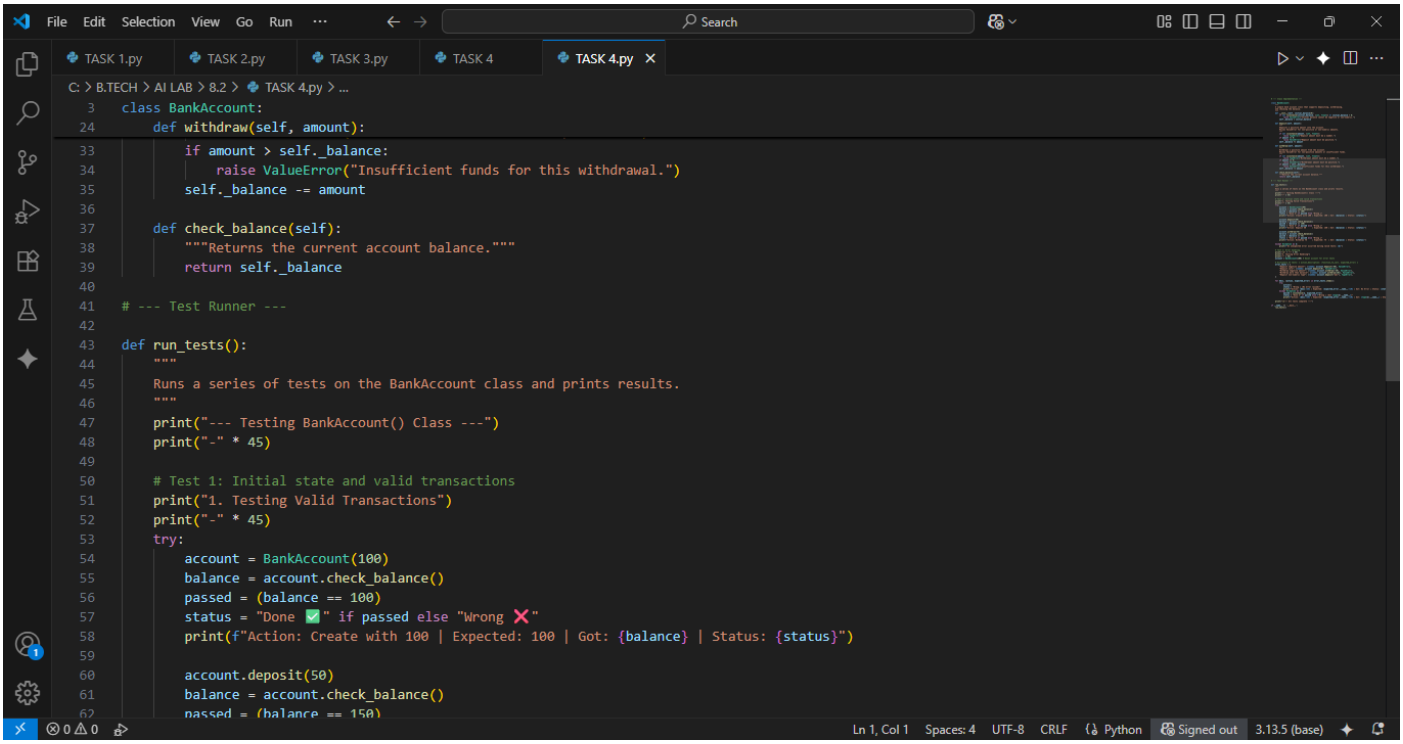
• Cannot withdraw more than balance.

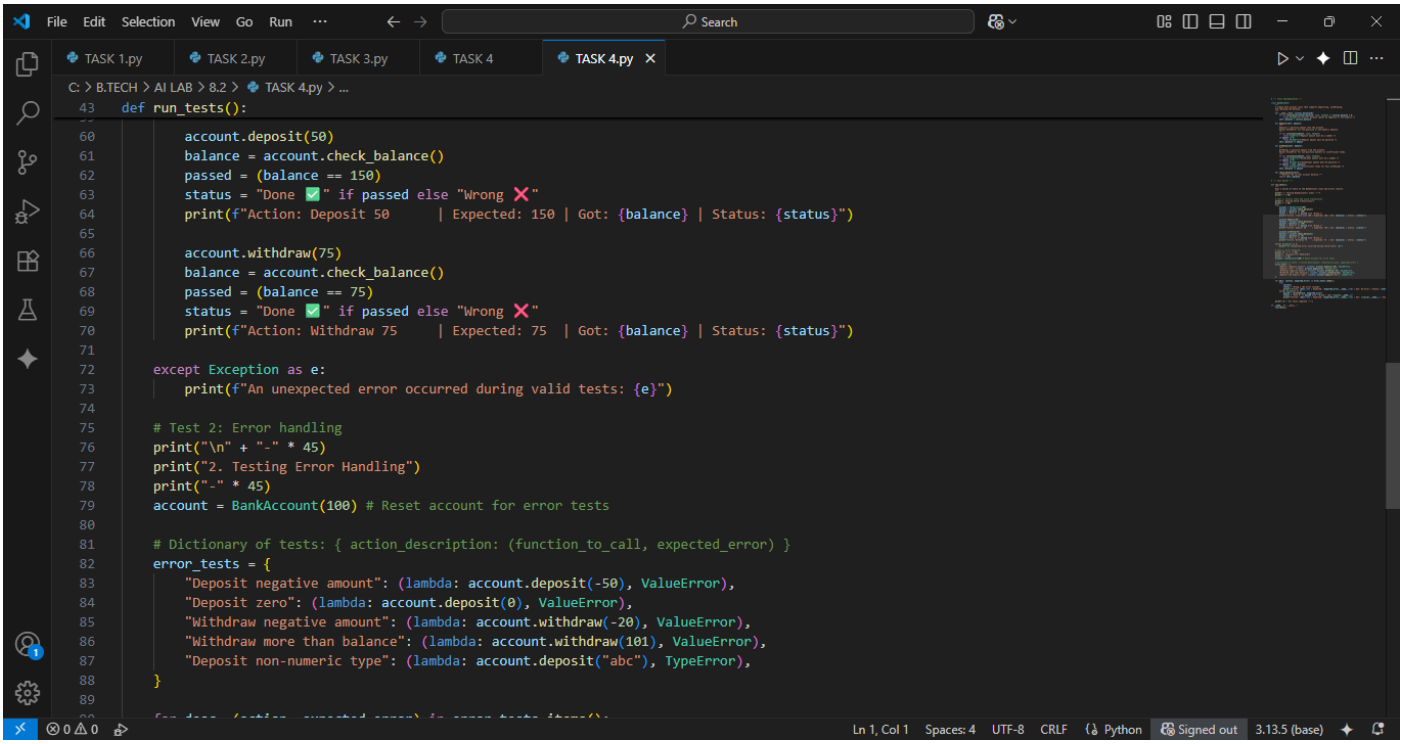
PROMPT:

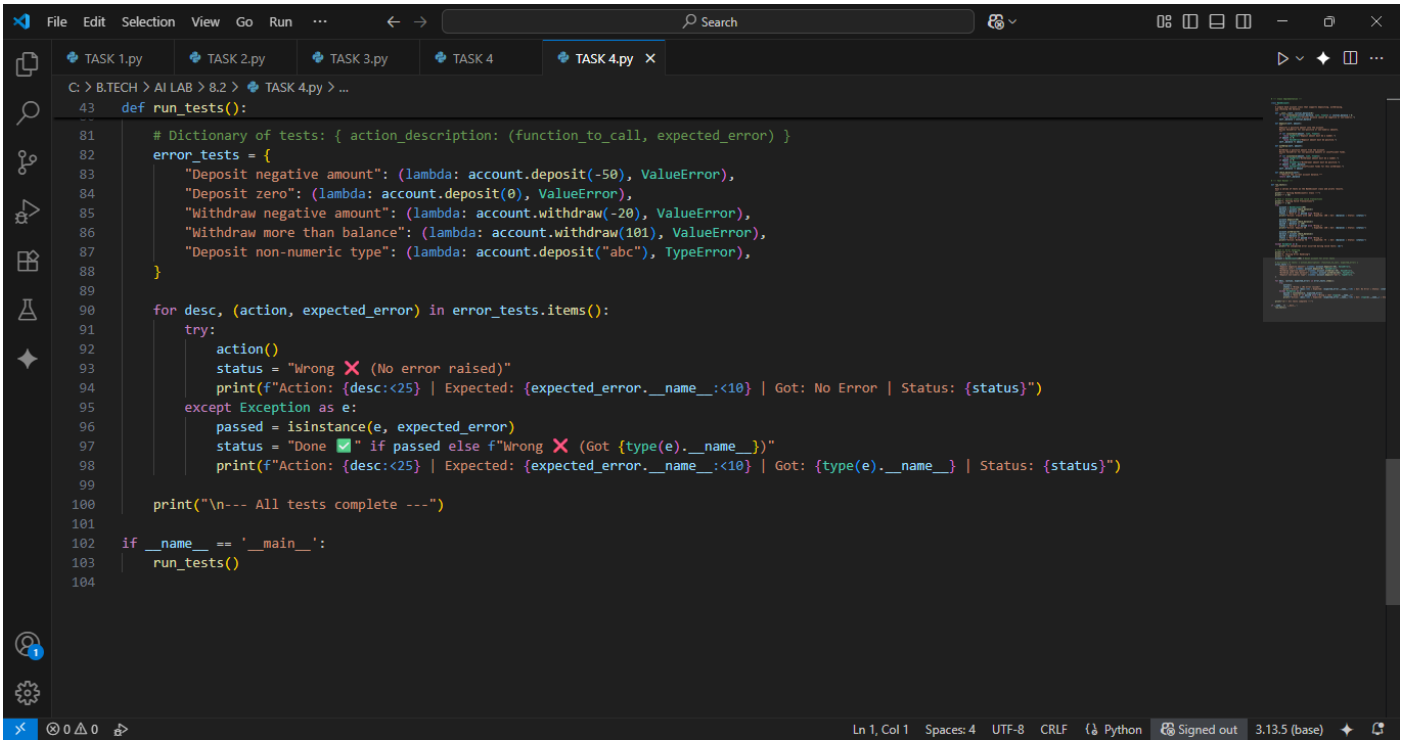
Please write a Python BankAccount class and a corresponding suite of test cases to validate its functionality.

CODE:

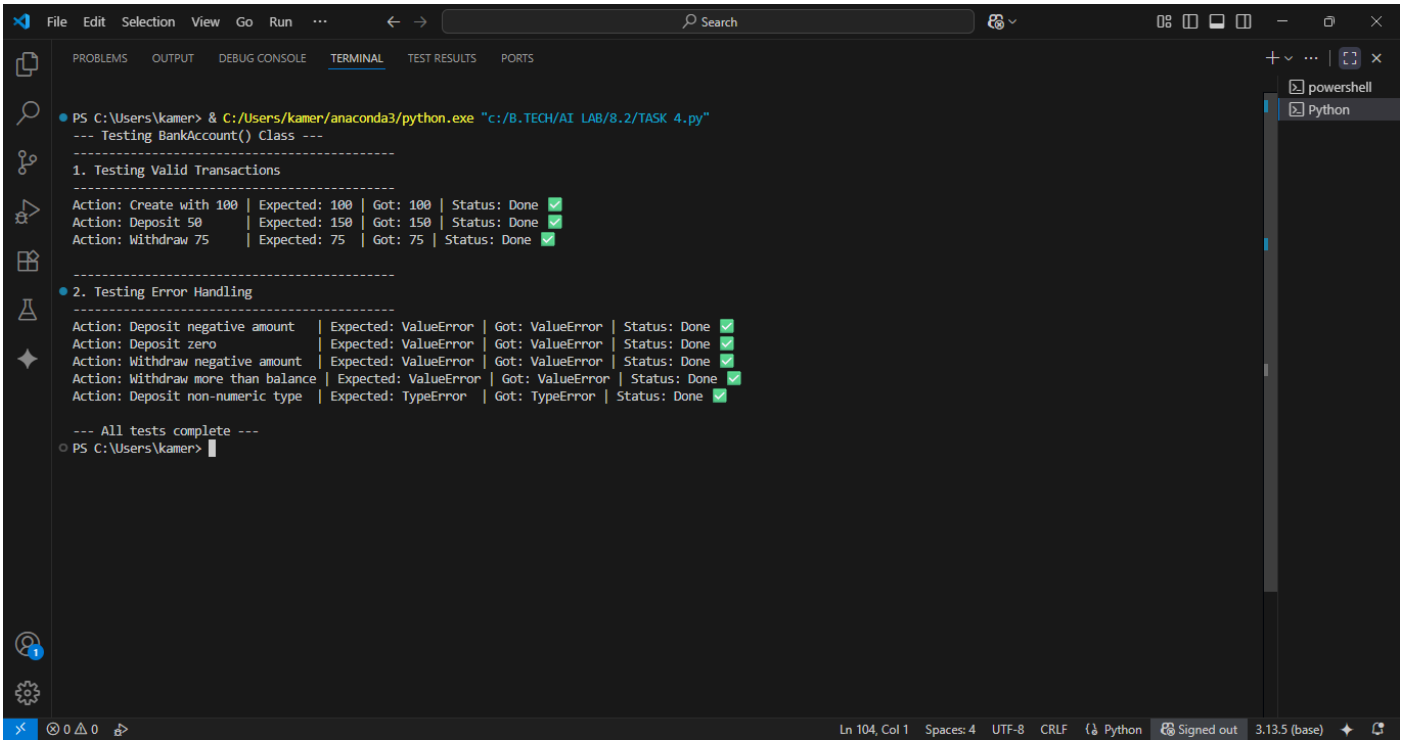

File Edit Selection View Go Run ···
Search
TASK 1.py
TASK 2.py
TASK 3.py
TASK 4
TASK 4.py
C: > B.TECH > AI LAB > 8.2 > TASK 4.py > ...
# --- Class Implementation ---
class BankAccount:
"""
A simple bank account class that supports depositing, withdrawing,
and checking the balance.
"""
def __init__(self, initial_balance=0):
if not isinstance(initial_balance, (int, float)) or initial_balance < 0:
raise ValueError("Initial balance cannot be negative or non-numeric.")
self._balance = initial_balance
def deposit(self, amount):
"""
Deposits a positive amount into the account.
Raises ValueError for non-positive or non-numeric amounts.
"""
if not isinstance(amount, (int, float)):
raise TypeError("Deposit amount must be a number.")
if amount <= 0:
raise ValueError("Deposit amount must be positive.")
self._balance += amount
def withdraw(self, amount):
"""
Withdraws a positive amount from the account.
Raises ValueError for non-positive amounts or insufficient funds.
"""
if not isinstance(amount, (int, float)):
raise TypeError("Withdrawal amount must be a number.")
if amount <= 0:
raise ValueError("Withdrawal amount must be positive.")
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python Signed out 3.13.5 (base)


File Edit Selection View Go Run ···
Search
TASK 1.py
TASK 2.py
TASK 3.py
TASK 4
TASK 4.py
C: > B.TECH > AI LAB > 8.2 > TASK 4.py > ...
class BankAccount:
def withdraw(self, amount):
if amount > self._balance:
raise ValueError("Insufficient funds for this withdrawal.")
self._balance -= amount
def check_balance(self):
"""Returns the current account balance."""
return self._balance
# --- Test Runner ---
def run_tests():
"""
Runs a series of tests on the BankAccount class and prints results.
"""
print("--- Testing BankAccount() Class ---")
print("-" * 45)
# Test 1: Initial state and valid transactions
print("1. Testing Valid Transactions")
print("-" * 45)
try:
account = BankAccount(100)
balance = account.check_balance()
passed = (balance == 100)
status = "Done ✅" if passed else "Wrong ❌"
print(f"Action: Create with 100 | Expected: 100 | Got: {balance} | Status: {status}")
account.deposit(50)
balance = account.check_balance()
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python Signed out 3.13.5 (base)


File Edit Selection View Go Run
Search
TASK 1.py
TASK 2.py
TASK 3.py
TASK 4
TASK 4.py
C: > B.TECH > AI LAB > 8.2 > TASK 4.py > ...
def run_tests():
account.deposit(50)
balance = account.check_balance()
passed = (balance == 150)
status = "Done ✅" if passed else "Wrong ❌"
print(f"Action: Deposit 50 | Expected: 150 | Got: {balance} | Status: {status}")
account.withdraw(75)
balance = account.check_balance()
passed = (balance == 75)
status = "Done ✅" if passed else "Wrong ❌"
print(f"Action: Withdraw 75 | Expected: 75 | Got: {balance} | Status: {status}")
except Exception as e:
print(f"An unexpected error occurred during valid tests: {e}")
# Test 2: Error handling
print("\n" + "-" * 45)
print("2. Testing Error Handling")
print("-" * 45)
account = BankAccount(100) # Reset account for error tests
# Dictionary of tests: { action_description: (function_to_call, expected_error) }
error_tests = {
"Deposit negative amount": (lambda: account.deposit(-50), ValueError),
"Deposit zero": (lambda: account.deposit(0), ValueError),
"Withdraw negative amount": (lambda: account.withdraw(-20), ValueError),
"Withdraw more than balance": (lambda: account.withdraw(101), ValueError),
"Deposit non-numeric type": (lambda: account.deposit("abc"), TypeError),
}
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python Signed out 3.13.5 (base)


File Edit Selection View Go Run
Search
TASK 1.py
TASK 2.py
TASK 3.py
TASK 4
TASK 4.py
C: > B.TECH > AI LAB > 8.2 > TASK 4.py > ...
def run_tests():
# Dictionary of tests: { action_description: (function_to_call, expected_error) }
error_tests = {
"Deposit negative amount": (lambda: account.deposit(-50), ValueError),
"Deposit zero": (lambda: account.deposit(0), ValueError),
"Withdraw negative amount": (lambda: account.withdraw(-20), ValueError),
"Withdraw more than balance": (lambda: account.withdraw(101), ValueError),
"Deposit non-numeric type": (lambda: account.deposit("abc"), TypeError),
}
for desc, (action, expected_error) in error_tests.items():
try:
action()
status = "Wrong ❌ (No error raised)"
print(f"Action: {desc:<25} | Expected: {expected_error.__name__:<10} | Got: No Error | Status: {status}")
except Exception as e:
passed = isinstance(e, expected_error)
status = "Done ✅" if passed else f"Wrong ❌ (Got {type(e).__name__})"
print(f"Action: {desc:<25} | Expected: {expected_error.__name__:<10} | Got: {type(e).__name__} | Status: {status}")
print("\n--- All tests complete ---")
if __name__ == '__main__':
run_tests()
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python Signed out 3.13.5 (base)

OUTPUT:



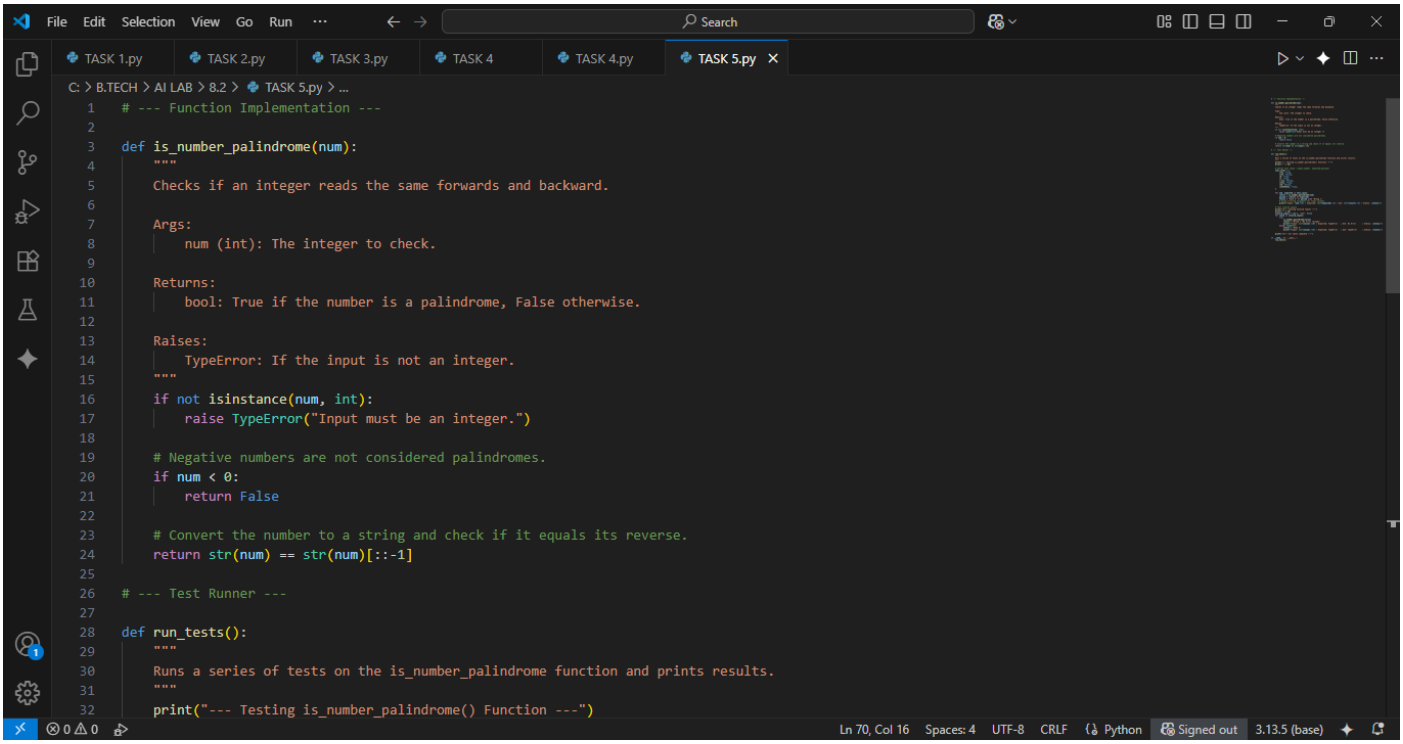
TASK 5

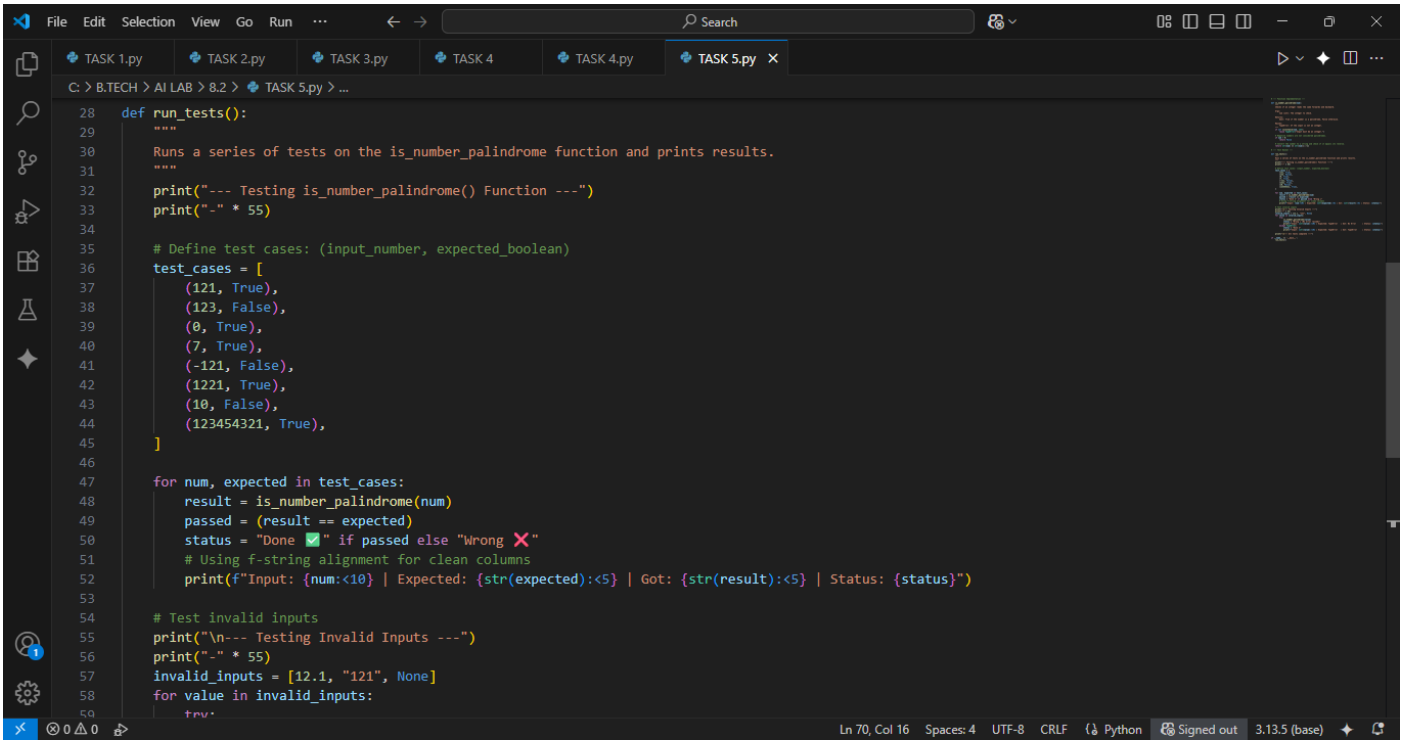
Task Description:
Generate test cases for is_number_palindrome(num), which checks if an integer reads
the same backward.
Examples:
121 → True
123 → False
0, negative numbers → handled gracefully.

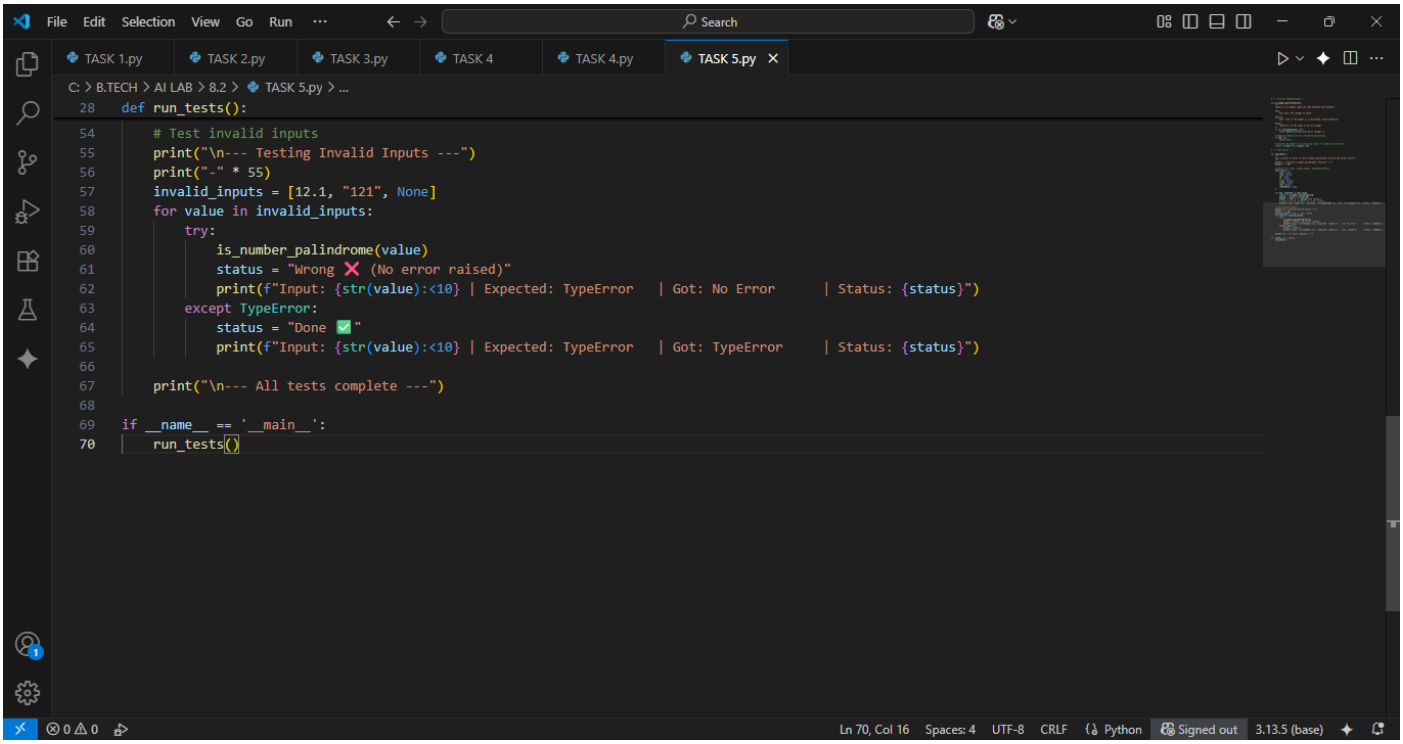
PROMPT:

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

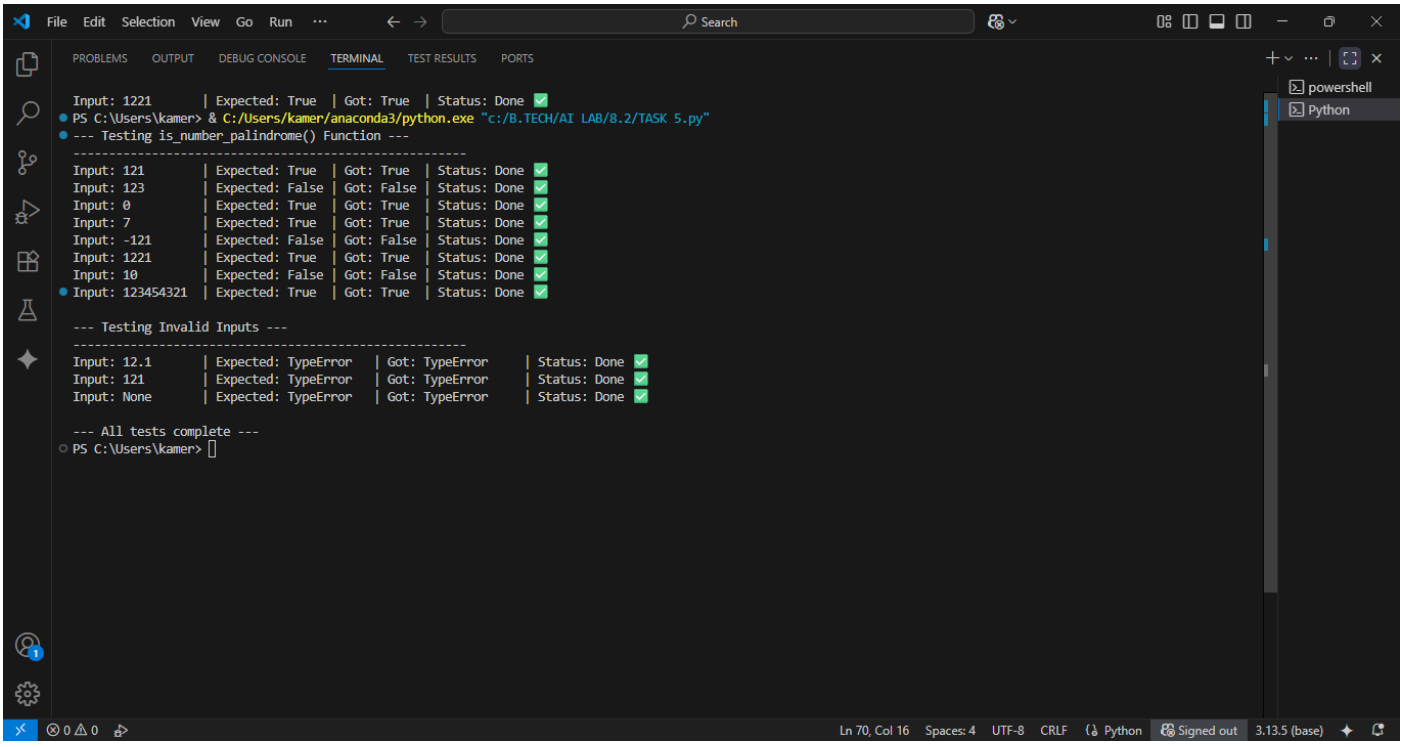
CODE:


File Edit Selection View Go Run
Search
TASK 1.py
TASK 2.py
TASK 3.py
TASK 4
TASK 4.py
TASK 5.py
C: > B.TECH > AI LAB > 8.2 > TASK 5.py > ...
# --- Function Implementation ---
def is_number_palindrome(num):
"""
Checks if an integer reads the same forwards and backward.
Args:
num (int): The integer to check.
Returns:
bool: True if the number is a palindrome, False otherwise.
Raises:
TypeError: If the input is not an integer.
"""
if not isinstance(num, int):
raise TypeError("Input must be an integer.")
# Negative numbers are not considered palindromes.
if num < 0:
return False
Convert the number to a string and check if it equals its reverse.
return str(num) == str(num)[::-1]
# --- Test Runner ---
def run_tests():
"""
Runs a series of tests on the is_number_palindrome function and prints results.
"""
print("--- Testing is_number_palindrome() Function ---")
Ln 70, Col 16
Spaces: 4
UTF-8
CRLF
Python
Signed out
3.13.5 (base)


File Edit Selection View Go Run
Search
TASK 1.py
TASK 2.py
TASK 3.py
TASK 4
TASK 4.py
TASK 5.py
C: > B.TECH > AI LAB > 8.2 > TASK 5.py > ...
def run_tests():
"""
Runs a series of tests on the is_number_palindrome function and prints results.
"""
print("--- Testing is_number_palindrome() Function ---")
print("-" * 55)
# Define test cases: (input_number, expected_boolean)
test_cases = [
(121, True),
(123, False),
(0, True),
(7, True),
(-121, False),
(1221, True),
(10, False),
(123454321, True),
]
for num, expected in test_cases:
result = is_number_palindrome(num)
passed = (result == expected)
status = "Done ✅" if passed else "Wrong ❌"
# Using f-string alignment for clean columns
print(f"Input: {num:<10} | Expected: {str(expected):<5} | Got: {str(result):<5} | Status: {status}")
# Test invalid inputs
print("\n--- Testing Invalid Inputs ---")
print("-" * 55)
invalid_inputs = [12.1, "121", None]
for value in invalid_inputs:
Ln 70, Col 16
Spaces: 4
UTF-8
CRLF
Python
Signed out
3.13.5 (base)


File Edit Selection View Go Run
Search
TASK 1.py
TASK 2.py
TASK 3.py
TASK 4
TASK 4.py
TASK 5.py
C: > B.TECH > AI LAB > 8.2 > TASK 5.py > ...
def run_tests():
# Test invalid inputs
print("\n--- Testing Invalid Inputs ---")
print("-" * 55)
invalid_inputs = [12.1, "121", None]
for value in invalid_inputs:
try:
is_number_palindrome(value)
status = "Wrong ❌ (No error raised)"
print(f"Input: {str(value):<10} | Expected: TypeError | Got: No Error | Status: {status}")
except TypeError:
status = "Done ✅"
print(f"Input: {str(value):<10} | Expected: TypeError | Got: TypeError | Status: {status}")
print("\n--- All tests complete ---")
if __name__ == '__main__':
run_tests()
Ln 70, Col 16 Spaces: 4 UTF-8 CRLF Python Signed out 3.13.5 (base)

OUTPUT:


File Edit Selection View Go Run
Search
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS
powershell
Python
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>
Ln 70, Col 16 Spaces: 4 UTF-8 CRLF Python Signed out 3.13.5 (base)

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.