

## ASSIGNMENT-8

NAME: G.Rishikesh

HT.NO: 2403A52046

BATCH: AIB03

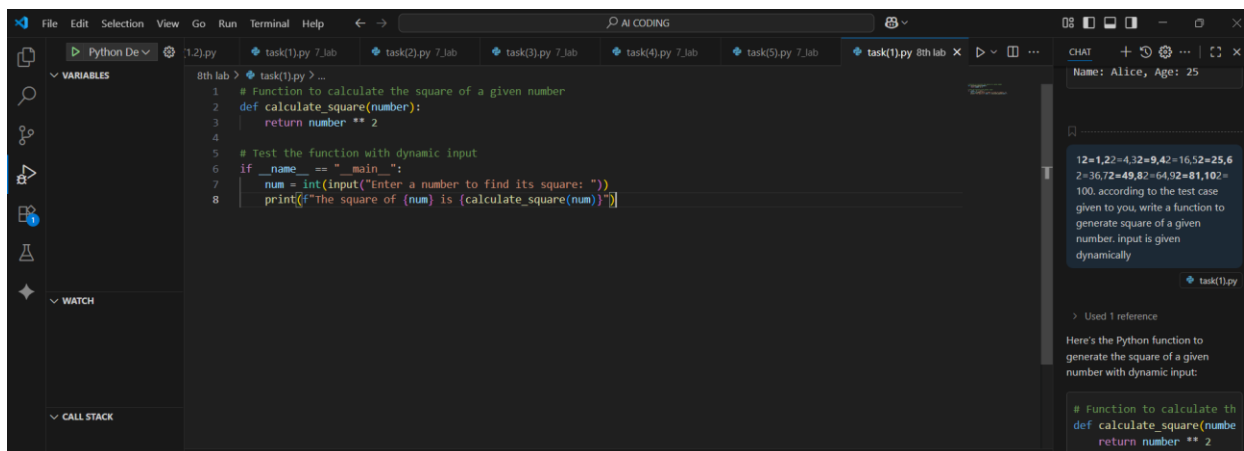
### Task-1

Write a test case to check if a function returns the square of a number. Then write the function with help from GitHub Copilot or Cursor AI.

Prompt:

1\*\*2=1,2\*\*2=4,3\*\*2=9,4\*\*2=16,5\*\*2=25,6\*\*2=36,7\*\*2=49,8\*\*2=64,9\*\*2=81,10\*\*2=100. according to the test case given to you, write a function to generate square of a given number. input is given dynamically

Code:



```
8th lab > task(1).py > ...
1 # Function to calculate the square of a given number
2 def calculate_square(number):
3     return number ** 2
4
5 # Test the function with dynamic input
6 if __name__ == "__main__":
7     num = int(input("Enter a number to find its square: "))
8     print(f"The square of {num} is {calculate_square(num)}")
```

CHAT

Name: Alice, Age: 25

12=1,22=4,32=9,42=16,52=25,62=36,72=49,82=64,92=81,102=100. according to the test case given to you, write a function to generate square of a given number. input is given dynamically

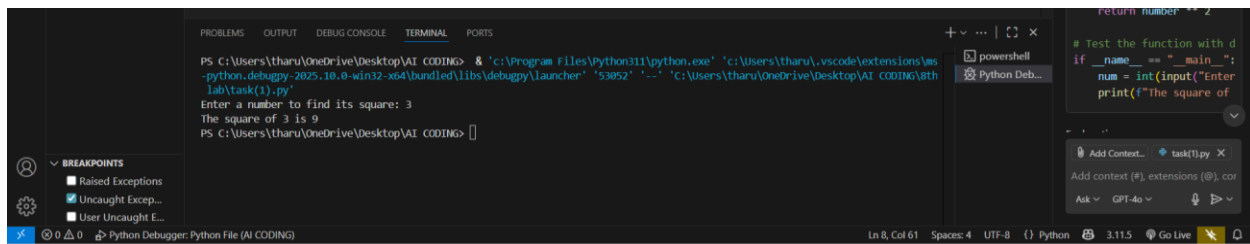
task(1).py

Used 1 reference

Here's the Python function to generate the square of a given number with dynamic input:

```
# Function to calculate the square of a given number
def calculate_square(number):
    return number ** 2
```

OP:



## Observation:

The `calculate_square` function takes a number as input and returns its square using the `**` operator. I have prompted to input a number dynamically. The program calculates the square of the input number and prints the result.

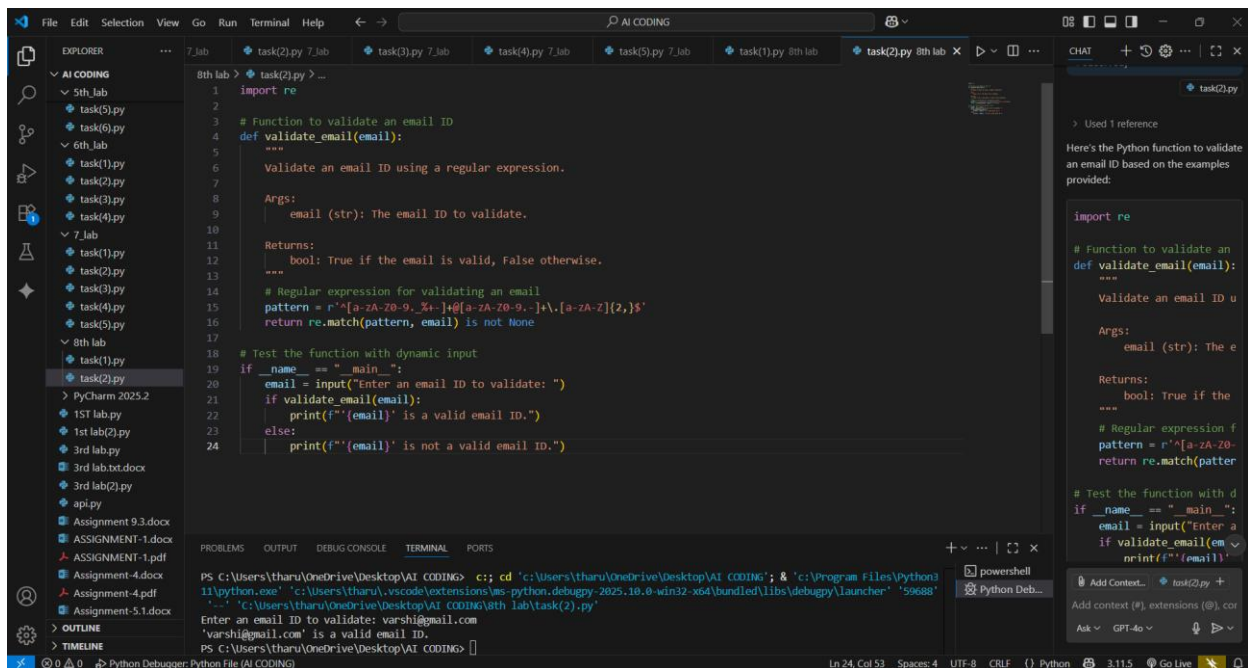
## Task-2:

Create test cases to validate an email address (e.g., contains `@` and `.com`). Use AI assistance to implement the `validate_email()` function

## Prompt:

write a function which validates a email id. examples of a mail id is `containt@gmail.com`, `madam@gmail.com`, `anyone@gmail.com`. according to the test case/examples given generate code.

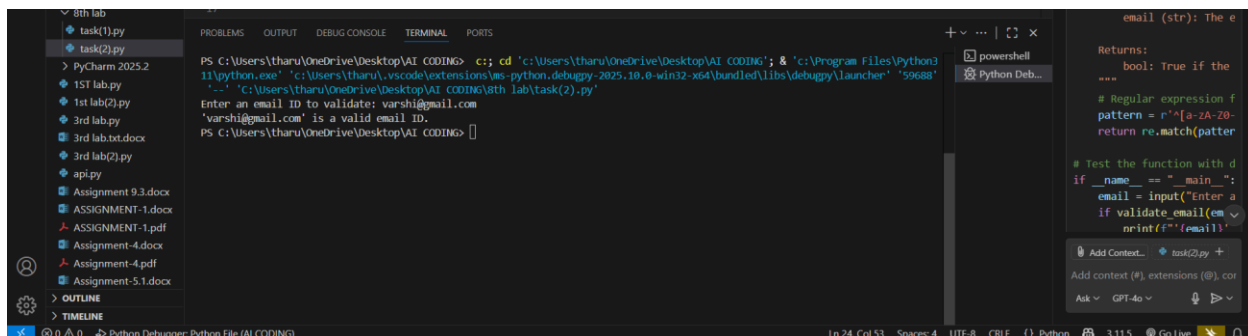
## Code:



```
8th lab > task(2).py > ...
1 import re
2
3 # Function to validate an email ID
4 def validate_email(email):
5     """
6     Validate an email ID using a regular expression.
7
8     Args:
9         email (str): The email ID to validate.
10
11     Returns:
12         bool: True if the email is valid, False otherwise.
13     """
14     # Regular expression for validating an email
15     pattern = r'^[a-zA-Z0-9_%.]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
16     return re.match(pattern, email) is not None
17
18 # Test the function with dynamic input
19 if __name__ == "__main__":
20     email = input("Enter an email ID to validate: ")
21     if validate_email(email):
22         print(f'{email} is a valid email ID.')
23     else:
24         print(f'{email} is not a valid email ID.')

PS C:\Users\tharu\OneDrive\Desktop\AI CODING> c:\cd 'c:\Users\tharu\OneDrive\Desktop\AI CODING'; & 'c:\Program Files\Python311\python.exe' 'c:\Users\tharu\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher' '59688'
Enter an email ID to validate: varshig@gmail.com
'varshig@gmail.com' is a valid email ID.
PS C:\Users\tharu\OneDrive\Desktop\AI CODING>
```

OP:



```
8th lab > task(2).py > ...
1 import re
2
3 # Function to validate an email ID
4 def validate_email(email):
5     """
6     Validate an email ID using a regular expression.
7
8     Args:
9         email (str): The email ID to validate.
10
11     Returns:
12         bool: True if the email is valid, False otherwise.
13     """
14     # Regular expression for validating an email
15     pattern = r'^[a-zA-Z0-9_%.]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
16     return re.match(pattern, email) is not None
17
18 # Test the function with dynamic input
19 if __name__ == "__main__":
20     email = input("Enter an email ID to validate: ")
21     if validate_email(email):
22         print(f'{email} is a valid email ID.')
23     else:
24         print(f'{email} is not a valid email ID.')

PS C:\Users\tharu\OneDrive\Desktop\AI CODING> c:\cd 'c:\Users\tharu\OneDrive\Desktop\AI CODING'; & 'c:\Program Files\Python311\python.exe' 'c:\Users\tharu\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher' '59688'
Enter an email ID to validate: varshig@gmail.com
'varshig@gmail.com' is a valid email ID.
PS C:\Users\tharu\OneDrive\Desktop\AI CODING>
```

Observation:

Examples like containt@gmail.com, madam@gmail.com, and anyone@gmail.com are valid because, They contain alphanumeric characters before the @. They have a domain name after the @ (e.g., gmail.com). The domain name ends with a valid top-level domain (e.g., .com). Emails without an @ symbol or domain name are invalid. Emails with special characters in invalid positions (e.g., @example.com or name@.com) are invalid.

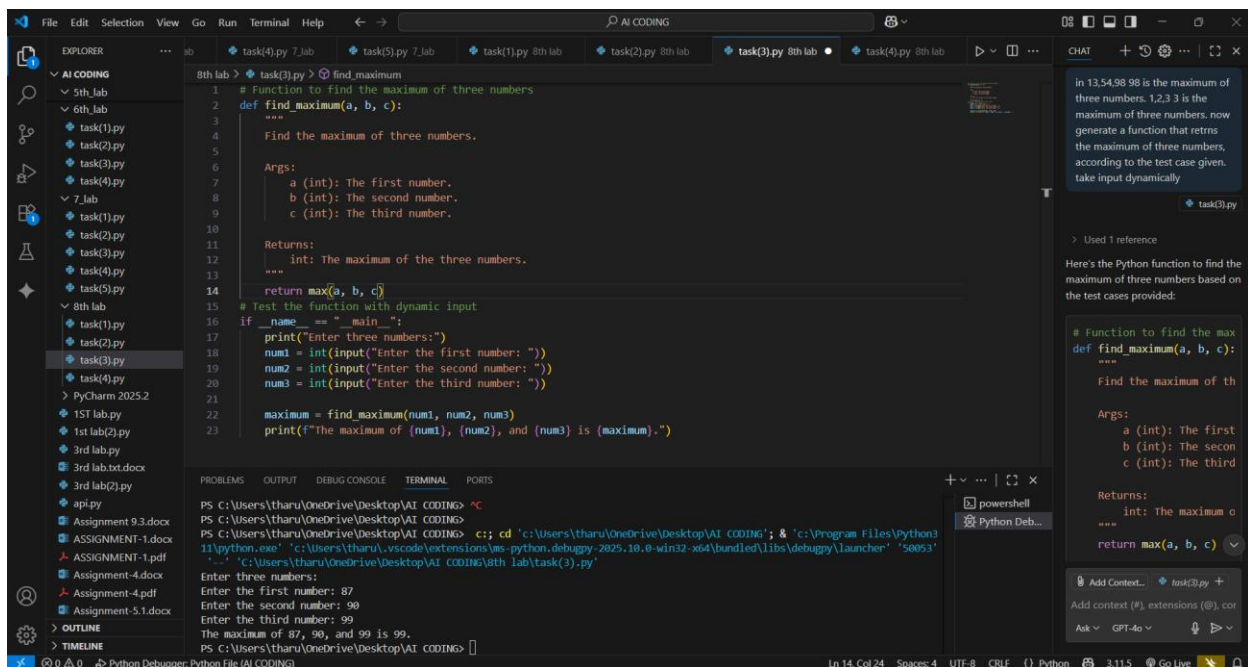
Task-3:

Write test cases for a function that returns the maximum of three numbers. Prompt Copilot/Cursor to write the logic based on tests

Prompt:

in 13,54,98 98 is the maximum of three numbers. 1,2,3 3 is the maximum of three numbers. now generate a function that returns the maximum of three numbers, according to the test case given. take input dynamically

Code:



The screenshot shows a code editor with a file explorer on the left, a main editor window, a terminal at the bottom, and a chat panel on the right. The file explorer shows a project named 'AI CODING' with several files. The main editor window displays a Python script named 'task(3).py' with the following code:

```
1 # Function to find the maximum of three numbers
2 def find_maximum(a, b, c):
3     """
4     Find the maximum of three numbers.
5
6     Args:
7         a (int): The first number.
8         b (int): The second number.
9         c (int): The third number.
10
11     Returns:
12         int: The maximum of the three numbers.
13     """
14     return max(a, b, c)
15
16 # Test the function with dynamic input
17 if __name__ == "__main__":
18     print("Enter three numbers:")
19     num1 = int(input("Enter the first number: "))
20     num2 = int(input("Enter the second number: "))
21     num3 = int(input("Enter the third number: "))
22
23     maximum = find_maximum(num1, num2, num3)
24     print(f"The maximum of {num1}, {num2}, and {num3} is {maximum}.")
```

The terminal at the bottom shows the execution of the script:

```
PS C:\Users\tharu\OneDrive\Desktop\AI CODING> ^C
PS C:\Users\tharu\OneDrive\Desktop\AI CODING> cd 'c:\Users\tharu\OneDrive\Desktop\AI CODING'; & 'c:\Program Files\Python311\python.exe' 'c:\Users\tharu\OneDrive\Desktop\AI CODING\ath lab\task(3).py'
Enter three numbers:
Enter the first number: 87
Enter the second number: 90
Enter the third number: 99
The maximum of 87, 90, and 99 is 99.
PS C:\Users\tharu\OneDrive\Desktop\AI CODING>
```

The chat panel on the right contains a prompt and a response from the AI:

in 13,54,98 98 is the maximum of three numbers. 1,2,3 3 is the maximum of three numbers. now generate a function that returns the maximum of three numbers, according to the test case given. take input dynamically

Used 1 reference

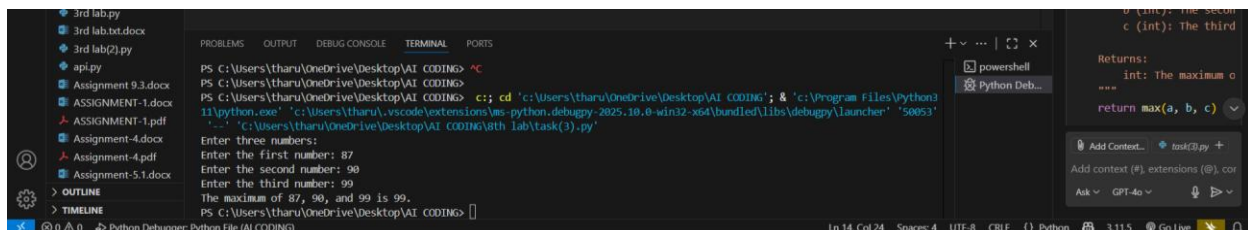
Here's the Python function to find the maximum of three numbers based on the test cases provided:

```
# Function to find the max
def find_maximum(a, b, c):
    """
    Find the maximum of three numbers.

    Args:
        a (int): The first number.
        b (int): The second number.
        c (int): The third number.

    Returns:
        int: The maximum of the three numbers.
    """
    return max(a, b, c)
```

OP:



The screenshot shows a code editor with a file explorer on the left, a main editor window, a terminal at the bottom, and a chat panel on the right. The file explorer shows a project named 'AI CODING' with several files. The main editor window displays a Python script named 'task(3).py' with the following code:

```
1 # Function to find the maximum of three numbers
2 def find_maximum(a, b, c):
3     """
4     Find the maximum of three numbers.
5
6     Args:
7         a (int): The first number.
8         b (int): The second number.
9         c (int): The third number.
10
11     Returns:
12         int: The maximum of the three numbers.
13     """
14     return max(a, b, c)
15
16 # Test the function with dynamic input
17 if __name__ == "__main__":
18     print("Enter three numbers:")
19     num1 = int(input("Enter the first number: "))
20     num2 = int(input("Enter the second number: "))
21     num3 = int(input("Enter the third number: "))
22
23     maximum = find_maximum(num1, num2, num3)
24     print(f"The maximum of {num1}, {num2}, and {num3} is {maximum}.")
```

The terminal at the bottom shows the execution of the script:

```
PS C:\Users\tharu\OneDrive\Desktop\AI CODING> ^C
PS C:\Users\tharu\OneDrive\Desktop\AI CODING> cd 'c:\Users\tharu\OneDrive\Desktop\AI CODING'; & 'c:\Program Files\Python311\python.exe' 'c:\Users\tharu\OneDrive\Desktop\AI CODING\ath lab\task(3).py'
Enter three numbers:
Enter the first number: 87
Enter the second number: 90
Enter the third number: 99
The maximum of 87, 90, and 99 is 99.
PS C:\Users\tharu\OneDrive\Desktop\AI CODING>
```

The chat panel on the right contains a prompt and a response from the AI:

in 13,54,98 98 is the maximum of three numbers. 1,2,3 3 is the maximum of three numbers. now generate a function that returns the maximum of three numbers, according to the test case given. take input dynamically

Used 1 reference

Here's the Python function to find the maximum of three numbers based on the test cases provided:

```
# Function to find the max
def find_maximum(a, b, c):
    """
    Find the maximum of three numbers.

    Args:
        a (int): The first number.
        b (int): The second number.
        c (int): The third number.

    Returns:
        int: The maximum of the three numbers.
    """
    return max(a, b, c)
```

Observation:

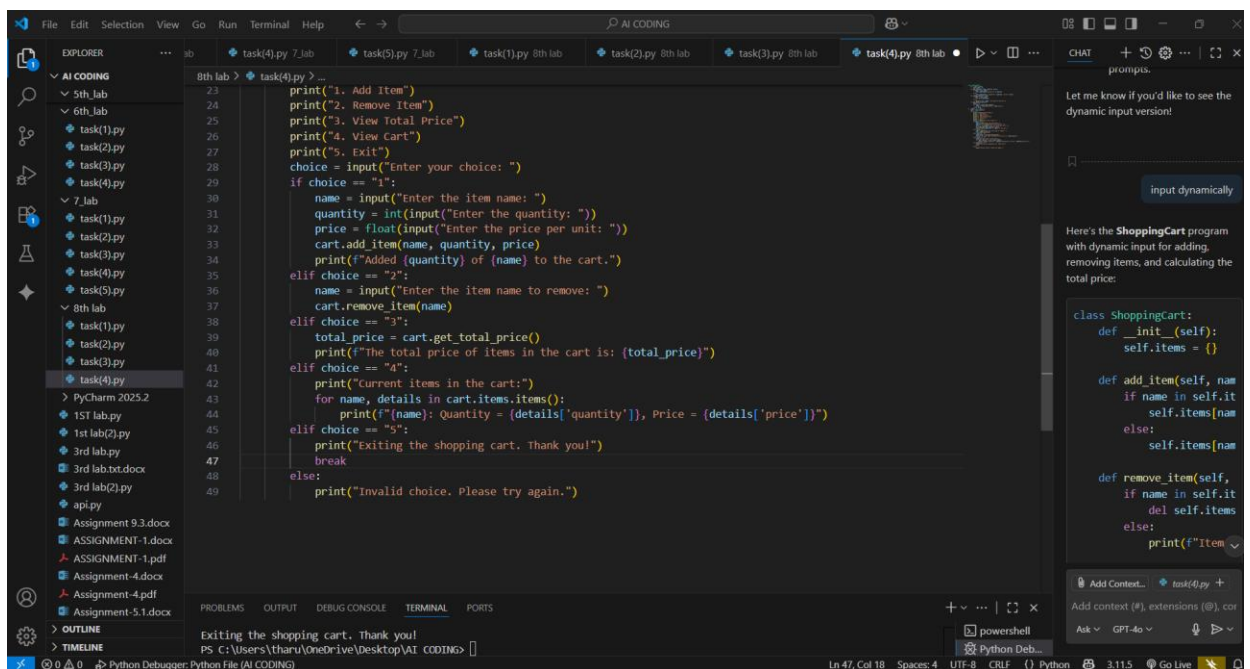
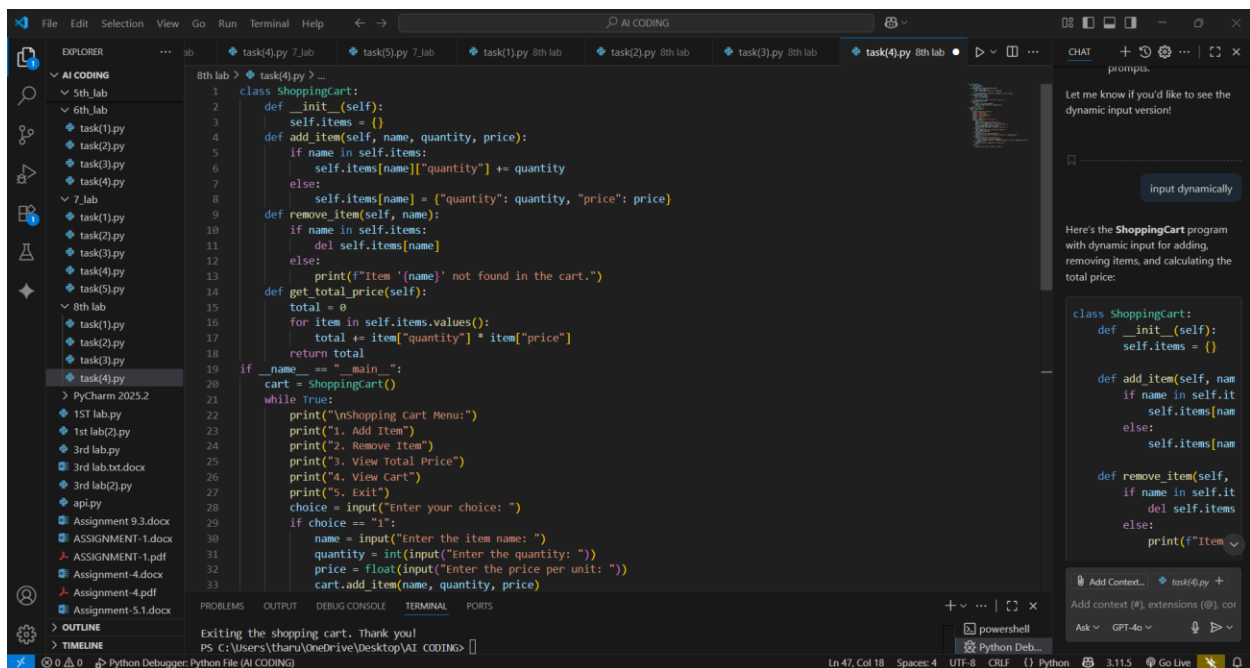
The `find_maximum` function takes three numbers as arguments and returns the maximum using Python's built-in `max()` function. I have prompted to input three numbers dynamically. The program calculates the maximum of the three numbers and prints the result.

Task-4:

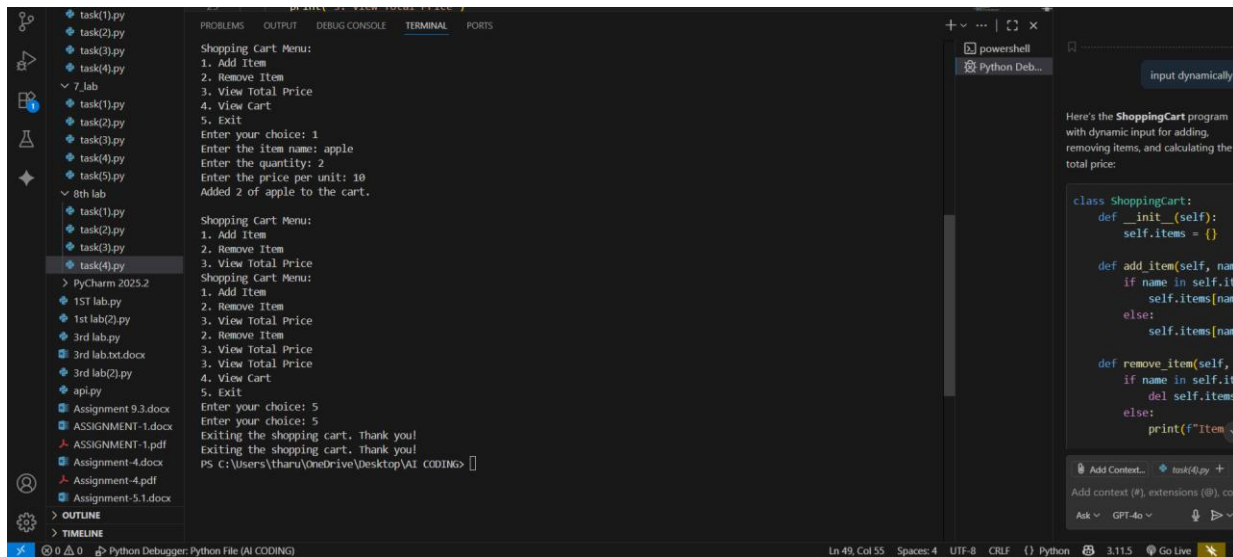
Use TDD to write a shopping cart class with methods to add, remove, and get total price. First write tests for each method, then generate code using AI.

Prompt: Now generate a Python class `ShoppingCart` that can add items, remove items, and calculate the total price according to the given test cases. Take input dynamically from the user. Input: Add Apple with quantity 2 and price 3.0 → Output: `{"Apple": {"quantity": 2, "price": 3.0}}`. Input: Remove Apple from the cart → Output: `{}`. Input: Add Apple (quantity 2, price 3.0) and Banana (quantity 1, price 1.5) → Output: Total price is 7.5.

Code:



OP:



Observation: The program uses input() to allow the user to interact with the shopping cart dynamically. Users can add items, remove items, view the total price, and see the cart's contents.

- **Option 1:** Add an item to the cart.
- **Option 2:** Remove an item from the cart.
- **Option 3:** Calculate and display the total price of items in the cart.
- **Option 4:** Display all items in the cart.
- **Option 5:** Exit the program.

If the user tries to remove an item that doesn't exist, the program displays an appropriate message

Task-5:

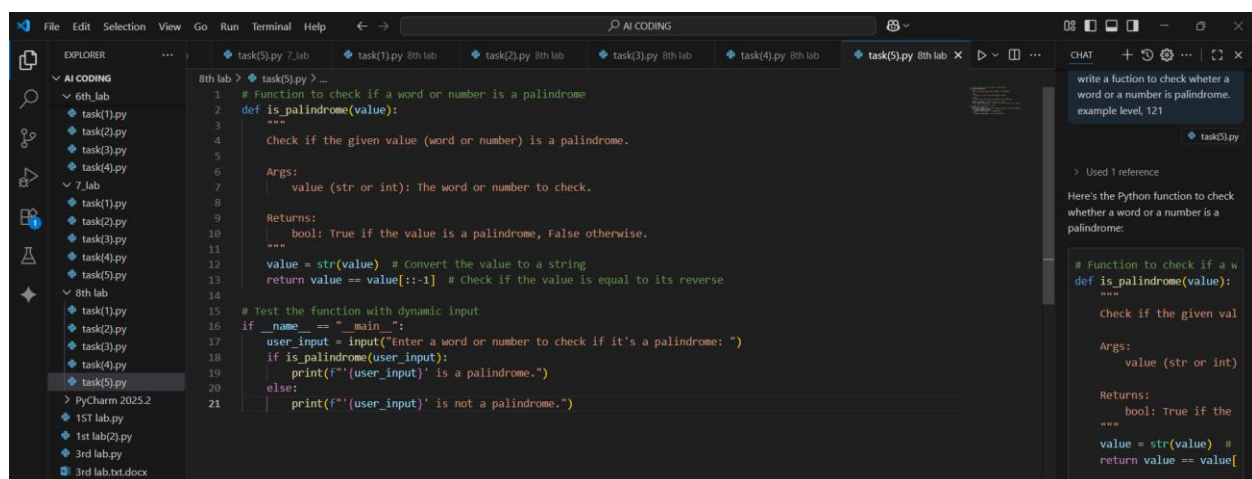


Write tests for a palindrome checker (e.g., `is_palindrome("level")` → `True`). Let Copilot suggest the function based on test case expectations.

Prompt:

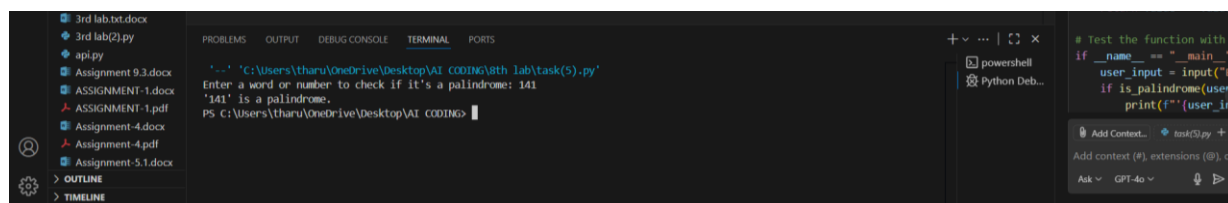
write a function to check whether a word or a number is palindrome or not. Example: level, 121

Code:



```
1 # Function to check if a word or number is a palindrome
2 def is_palindrome(value):
3     """
4     Check if the given value (word or number) is a palindrome.
5
6     Args:
7         value (str or int): The word or number to check.
8
9     Returns:
10         bool: True if the value is a palindrome, False otherwise.
11     """
12     value = str(value) # Convert the value to a string
13     return value == value[::-1] # Check if the value is equal to its reverse
14
15 # Test the function with dynamic input
16 if __name__ == "__main__":
17     user_input = input("Enter a word or number to check if it's a palindrome: ")
18     if is_palindrome(user_input):
19         print(f"{user_input} is a palindrome.")
20     else:
21         print(f"{user_input} is not a palindrome.")
```

OP:



```
PS C:\Users\tharu\OneDrive\Desktop\VAI CODING\8th lab> python task(5).py
Enter a word or number to check if it's a palindrome: 141
'141' is a palindrome.
```

Observation:

The input value is converted to a string using `str(value)` to handle both words and numbers. The function checks if the string is equal to its reverse using slicing (`value[::-1]`). I have prompted to enter a word or number dynamically. The program prints whether the input is a palindrome



