

# ASSIGNMENT-8

NAME:Nagarjuna Reddy

HT.NO: 2403A52064

BATCH: 03

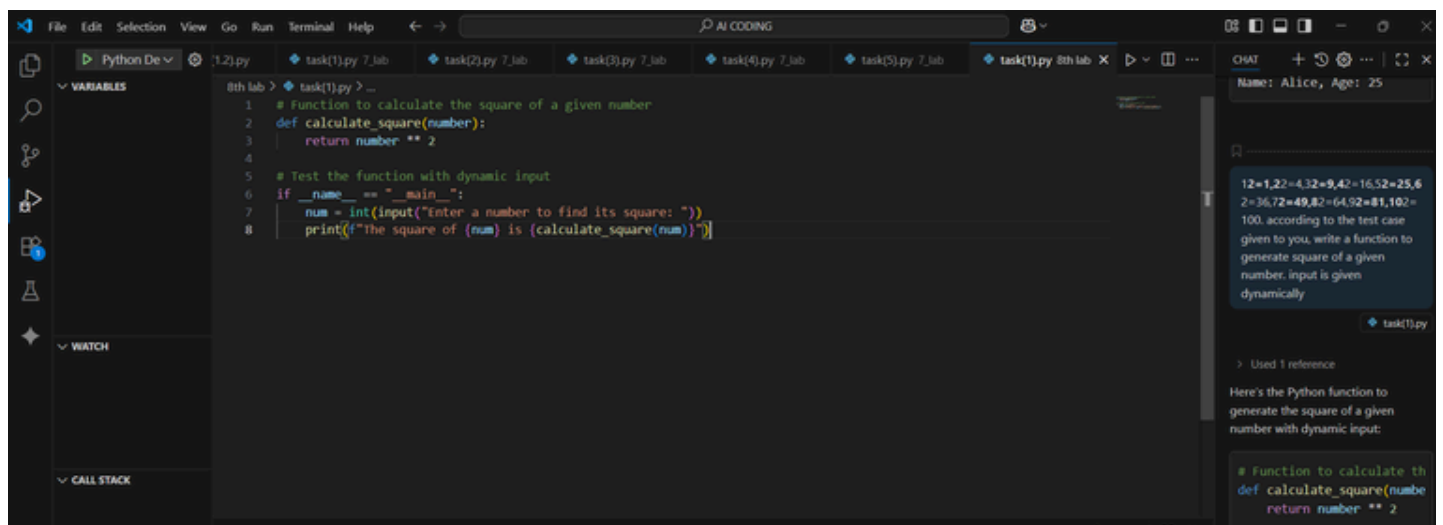
## Task-1

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:



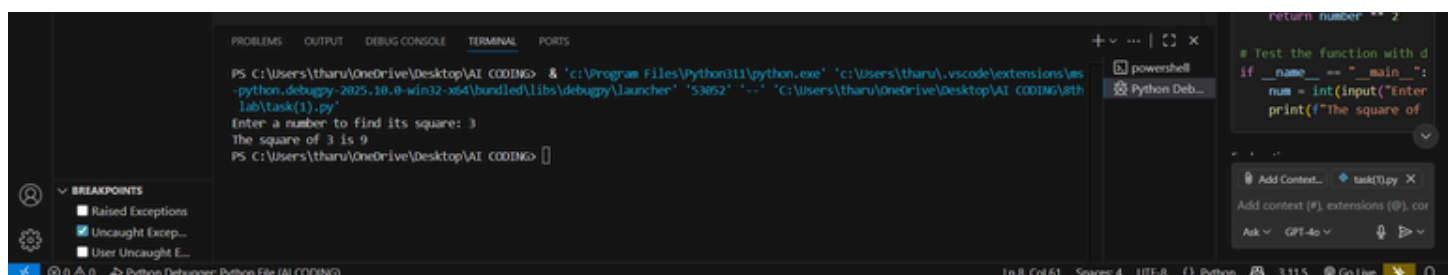
The screenshot shows the VS Code editor with a Python file named 'task(1).py'. The code defines a function 'calculate\_square' and tests it with dynamic input. The AI Copilot sidebar on the right provides suggestions and test cases.

```
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)}")
```

AI Copilot suggestions:

- Test cases: 12=1,22=4,32=9,42=16,52=25,62=36,72=49,82=64,92=81,102=100.
- Python function to generate the square of a given number with dynamic input:

OP:



The screenshot shows the VS Code terminal with the output of the Python script. The user entered '3' and the output is 'The square of 3 is 9'.

```
PS 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' '53852' '-' 'c:\Users\tharu\OneDrive\Desktop\AI CODING\8th lab\task(1).py'
Enter a number to find its square: 3
The square of 3 is 9
PS C:\Users\tharu\OneDrive\Desktop\AI CODING>
```

## 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.

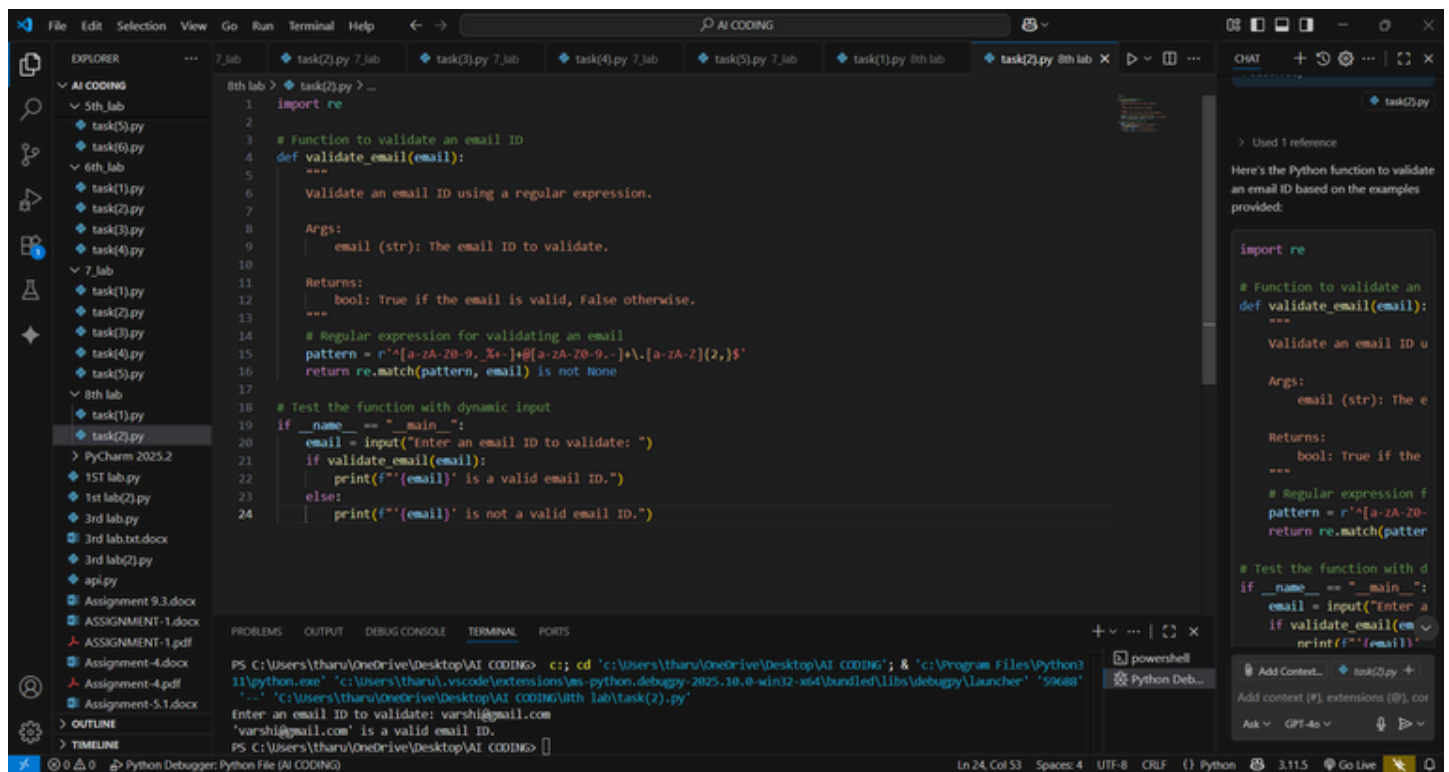
## Task-2:

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

## Prompt:

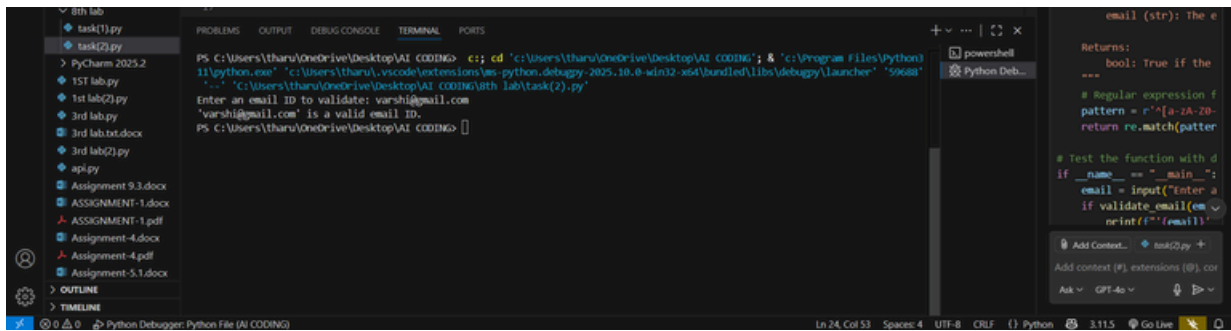
write a function which validates a email id. examples of a mail id is containt@gmail.com, madam@gmail.com, anyone@gmail.com. according to the test case/examples given generate code.

## Code:



```
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.')
25
26 # Run the script
27 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\vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher' '59688'
28 '...' 'C:\Users\tharu\OneDrive\Desktop\AI CODING\8th lab\task(2).py'
29 Enter an email ID to validate: varshi@gmail.com
30 'varshi@gmail.com' is a valid email ID.
31 PS C:\Users\tharu\OneDrive\Desktop\AI CODING>
```

## OP:



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:

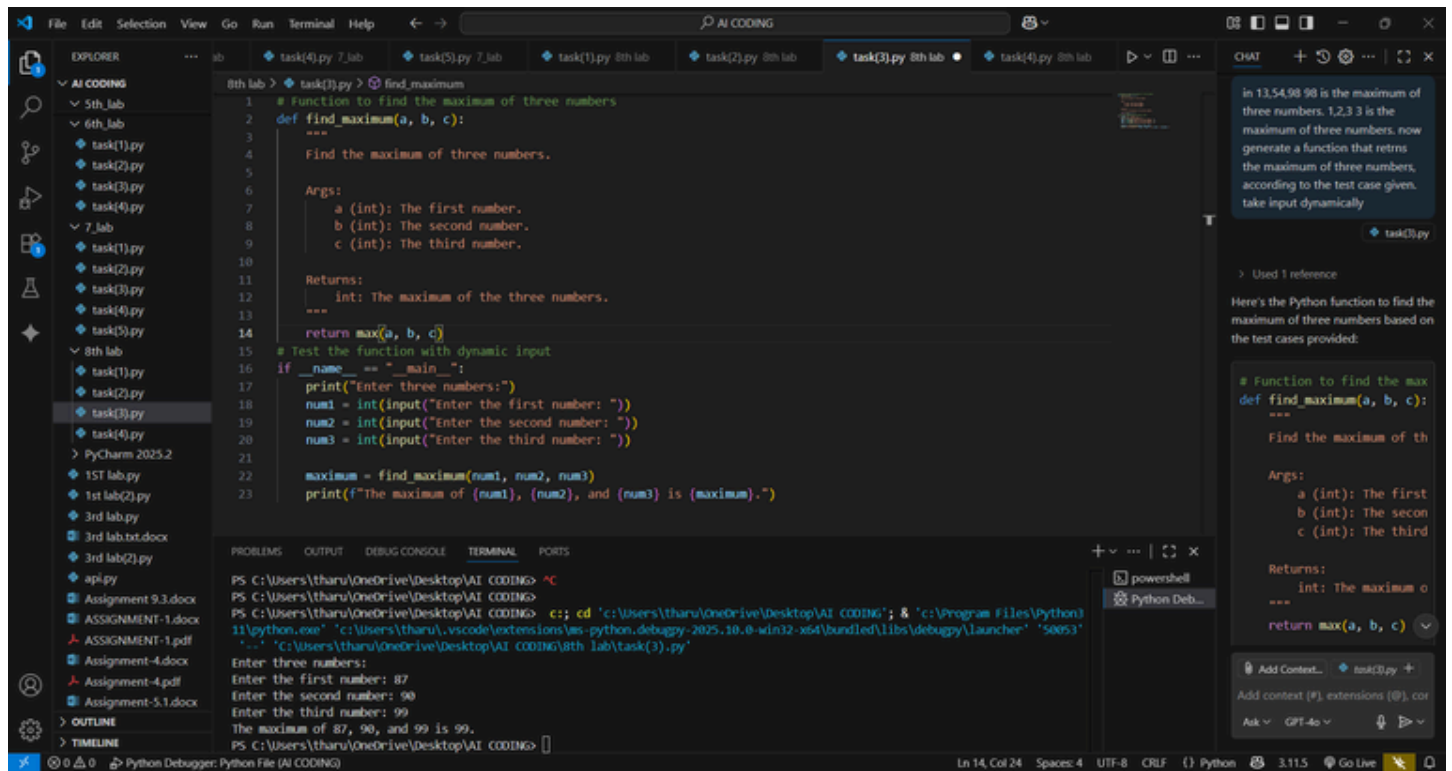
Write test cases for a function that returns the maximum of three numbers.

Prompt Copilot/Cursor to write the logic based on tests

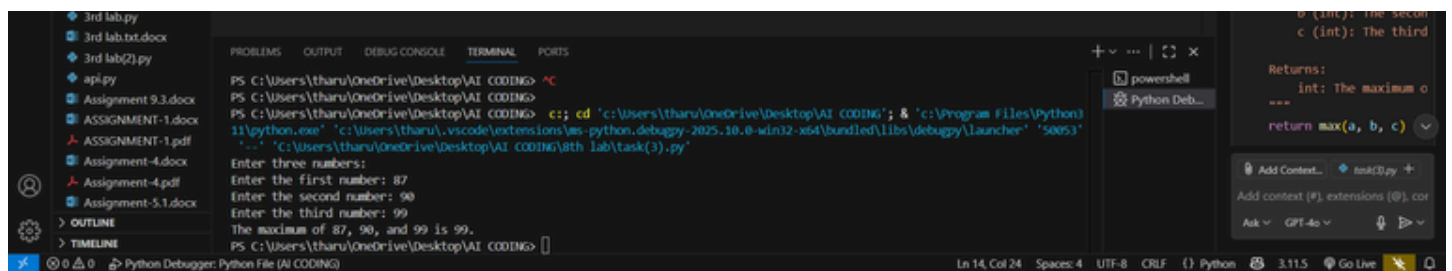
Prompt:

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:



OP:



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.

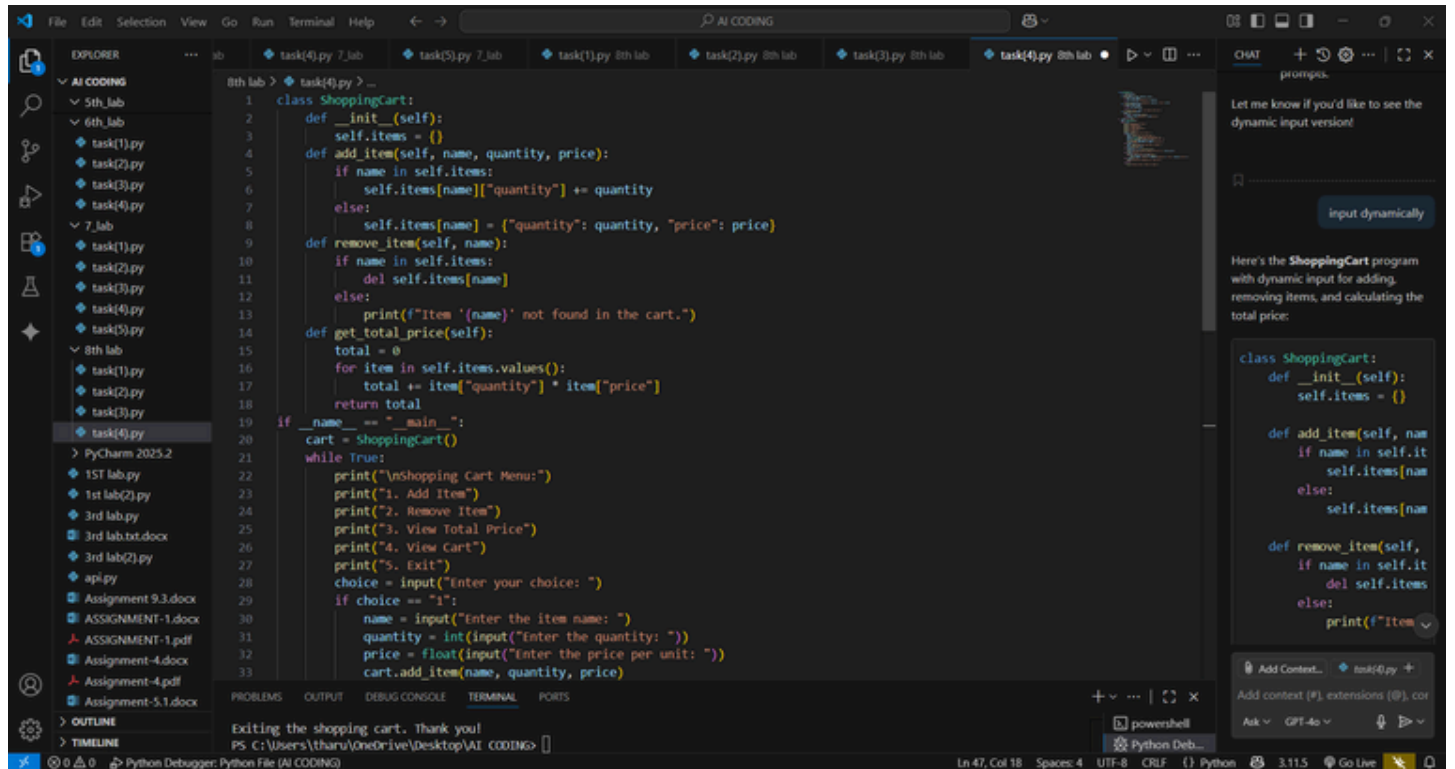
Task-4:

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.

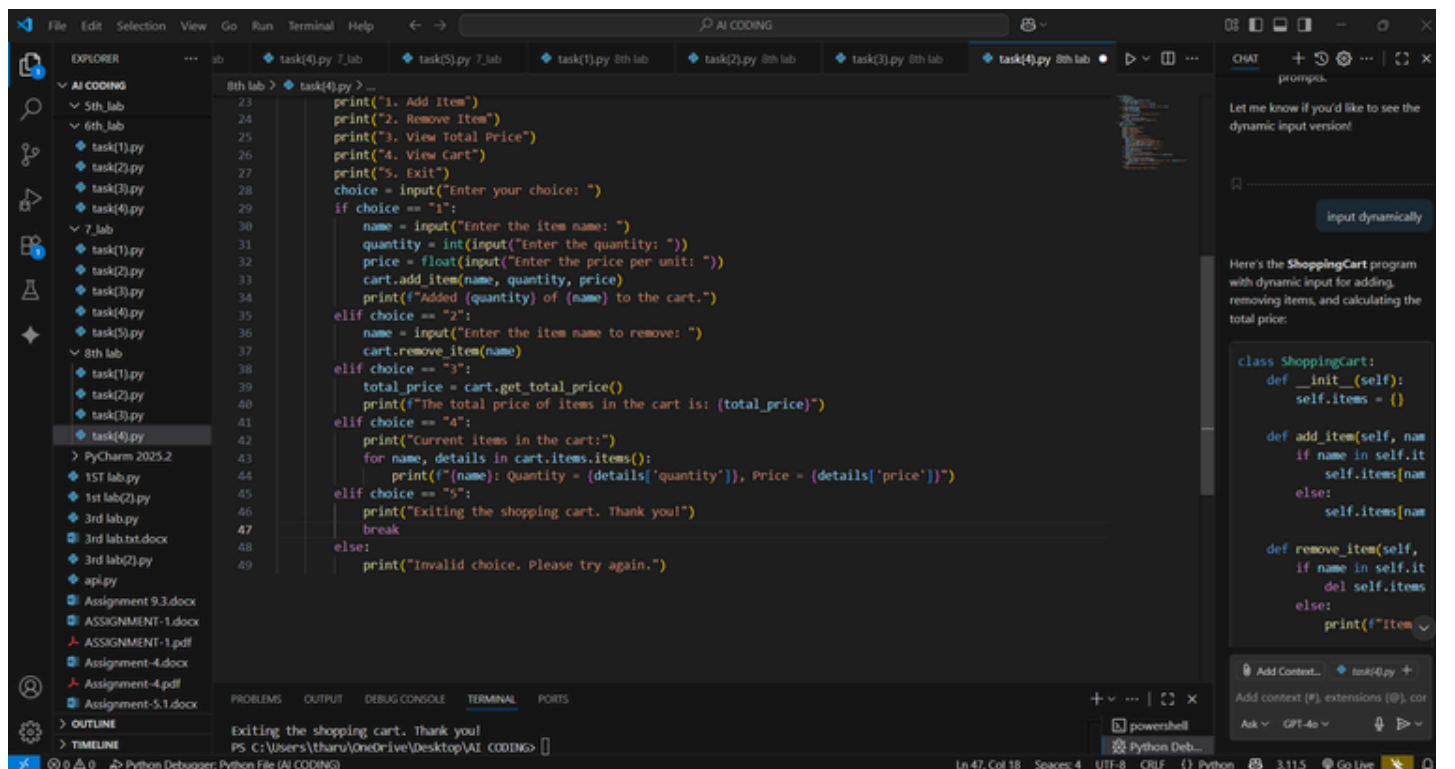
Prompt: Now generate a Python class `ShoppingCart` that can add items, remove items, and calculate the total price according to the given test cases. Take input dynamically from the user. Input: Add Apple with quantity 2 and price 3.0 →

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:



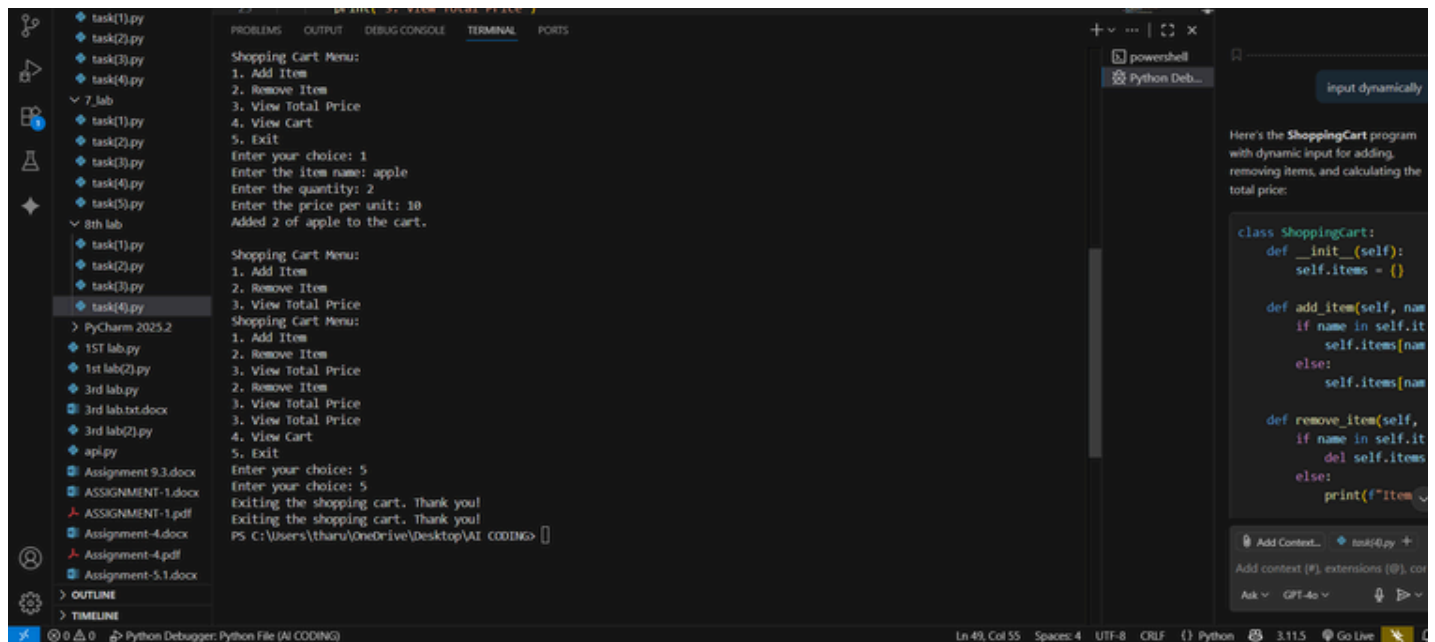
```
8th lab > task(4).py > ...
1 class ShoppingCart:
2     def __init__(self):
3         self.items = {}
4     def add_item(self, name, quantity, price):
5         if name in self.items:
6             self.items[name]["quantity"] += quantity
7         else:
8             self.items[name] = {"quantity": quantity, "price": price}
9     def remove_item(self, name):
10        if name in self.items:
11            del self.items[name]
12        else:
13            print(f"Item '{name}' not found in the cart.")
14    def get_total_price(self):
15        total = 0
16        for item in self.items.values():
17            total += item["quantity"] * item["price"]
18        return total
19    if __name__ == "__main__":
20        cart = ShoppingCart()
21        while True:
22            print("\nShopping Cart Menu:")
23            print("1. Add Item")
24            print("2. Remove Item")
25            print("3. View Total Price")
26            print("4. View Cart")
27            print("5. Exit")
28            choice = input("Enter your choice: ")
29            if choice == "1":
30                name = input("Enter the item name: ")
31                quantity = int(input("Enter the quantity: "))
32                price = float(input("Enter the price per unit: "))
33                cart.add_item(name, quantity, price)
34            elif choice == "2":
35                name = input("Enter the item name to remove: ")
36                cart.remove_item(name)
37            elif choice == "3":
38                total_price = cart.get_total_price()
39                print(f"The total price of items in the cart is: {total_price}")
40            elif choice == "4":
41                print("Current items in the cart:")
42                for name, details in cart.items.items():
43                    print(f"({name}): Quantity = {details['quantity']}, Price = {details['price']}")
44            elif choice == "5":
45                print("Exiting the shopping cart. Thank you!")
46                break
47            else:
48                print("Invalid choice. Please try again.")
```



```
8th lab > task(4).py > ...
23 print("1. Add Item")
24 print("2. Remove Item")
25 print("3. View Total Price")
26 print("4. View Cart")
27 print("5. Exit")
28 choice = input("Enter your choice: ")
29 if choice == "1":
30     name = input("Enter the item name: ")
31     quantity = int(input("Enter the quantity: "))
32     price = float(input("Enter the price per unit: "))
33     cart.add_item(name, quantity, price)
34     print(f"Added {quantity} of {name} to the cart.")
35 elif choice == "2":
36     name = input("Enter the item name to remove: ")
37     cart.remove_item(name)
38 elif choice == "3":
39     total_price = cart.get_total_price()
40     print(f"The total price of items in the cart is: {total_price}")
41 elif choice == "4":
42     print("Current items in the cart:")
43     for name, details in cart.items.items():
44         print(f"({name}): Quantity = {details['quantity']}, Price = {details['price']}")
45 elif choice == "5":
46     print("Exiting the shopping cart. Thank you!")
47     break
48 else:
49     print("Invalid choice. Please try again.")
```



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:

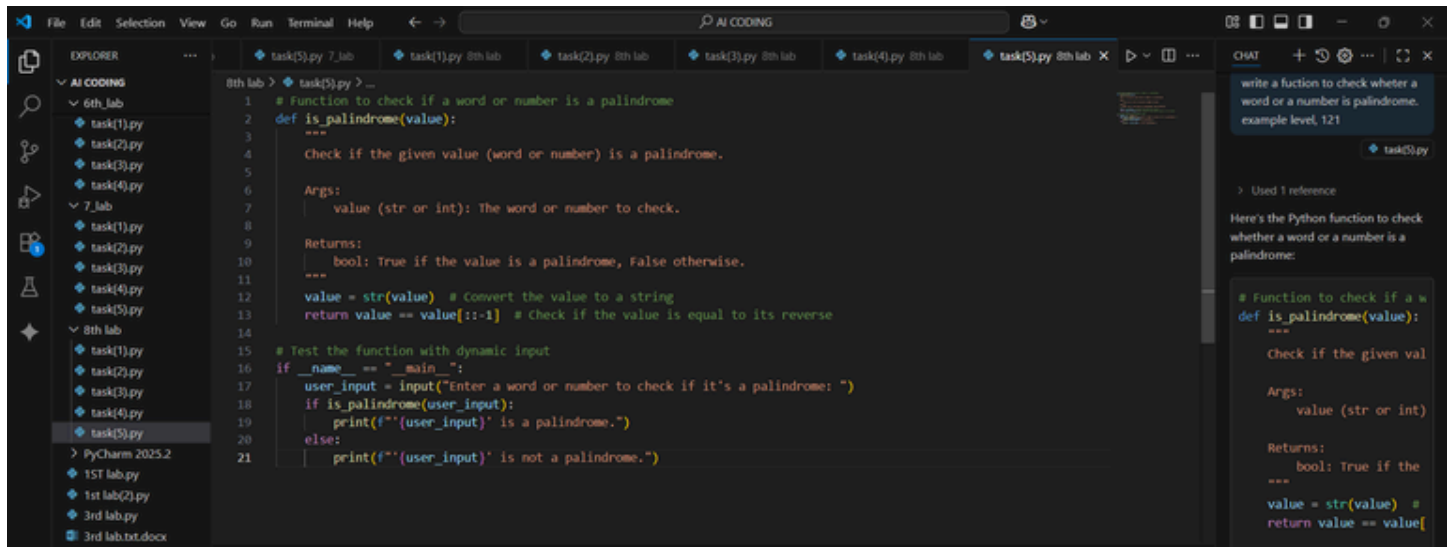
Write tests for a palindrome checker (e.g., `is_palindrome("level")` → `True`). Let Copilot suggest the function based on test case expectations.

Prompt:

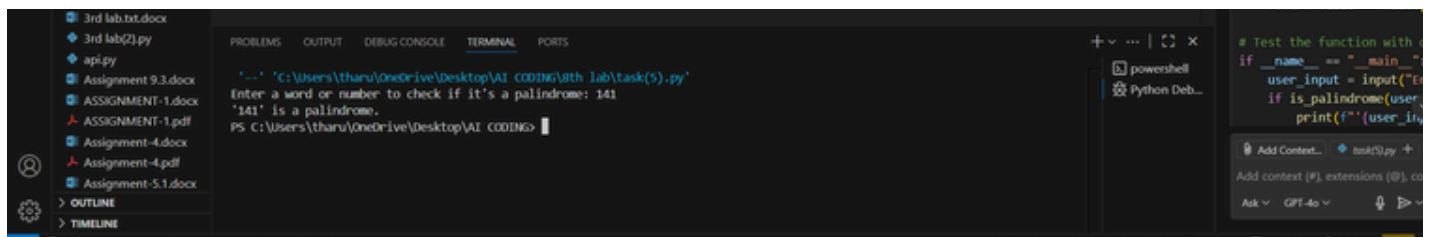
write a function to check whether a word or a number is palindrome or not.

Example: level, 121

Code:



OP:



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