

# AI AISSITED CODING

## ASSIGNMENT-8

NAME: N.PRANAY KUMAR

HT.NO: 2403A52087

BATCH: 04

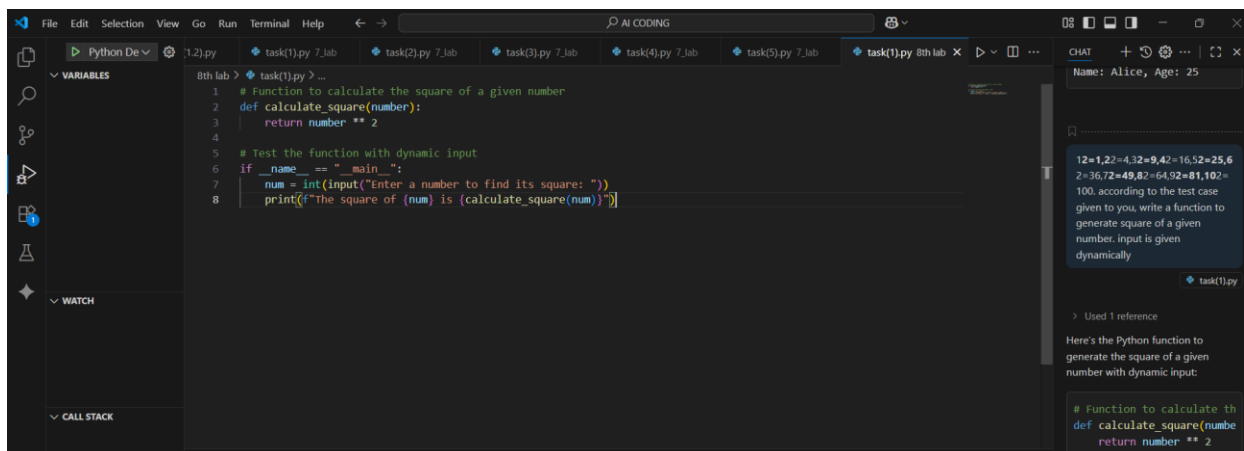
### Task-1

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

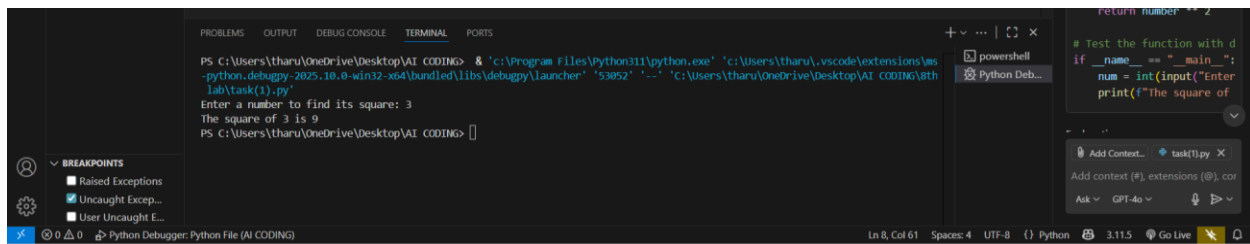
Code:



The screenshot shows a code editor with a Python file named 'task(1).py'. The code defines a function 'calculate\_square' that takes a number and returns its square. It also includes a test case that prompts the user to enter a number and prints the result. On the right side, there is a chat window with a prompt: '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'. The response from the chat is: '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'.

```
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)}")
```

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.

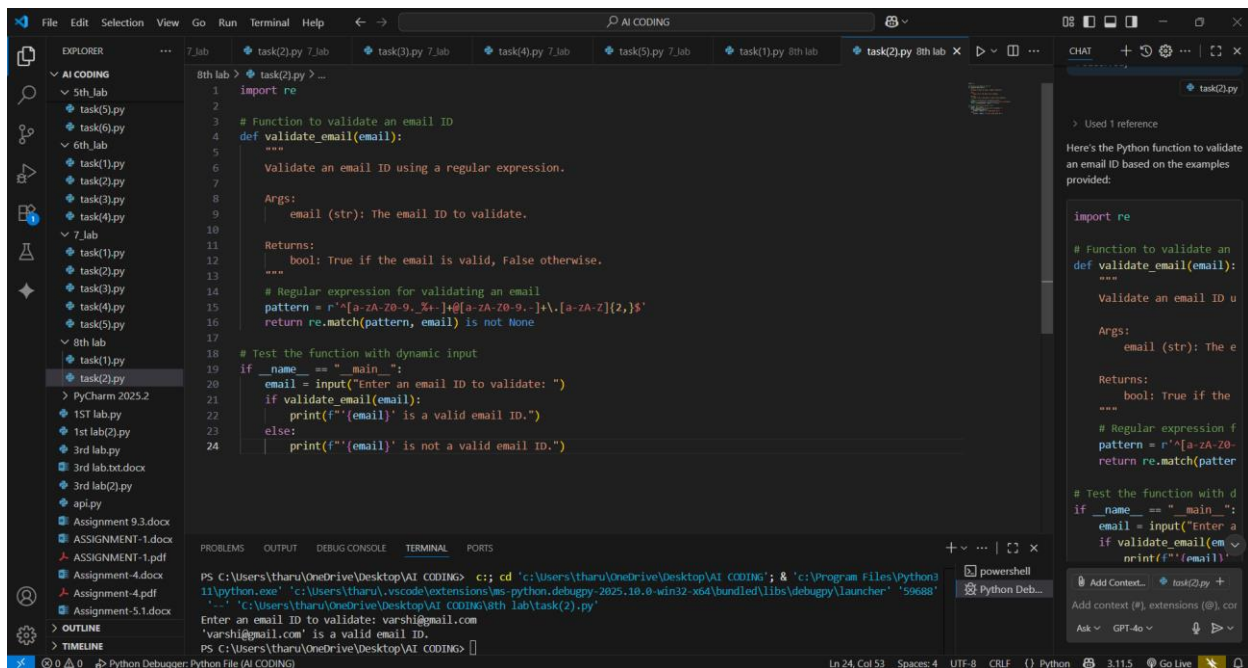
## Task-2:

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

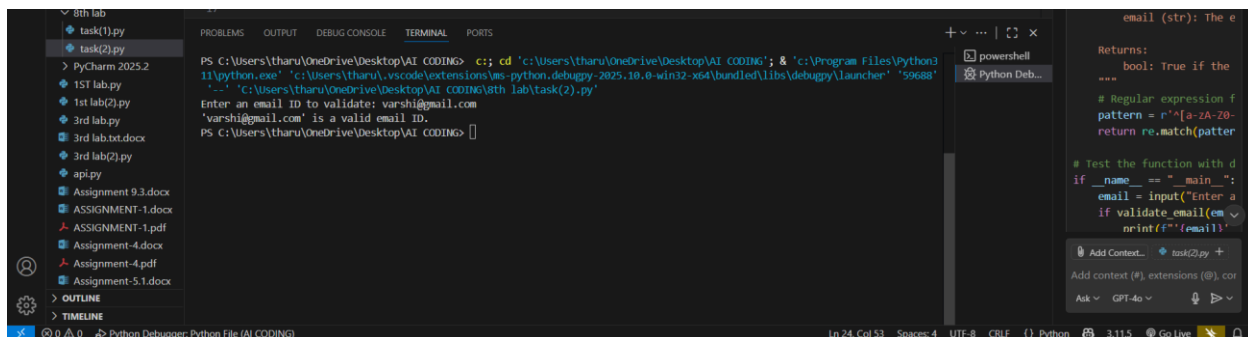
## Prompt:

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:



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:

The screenshot shows a code editor with a file explorer on the left, a main editor window, and a terminal at the bottom. 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> c:; 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>
```

OP:

This screenshot is identical to the one above, showing the same code editor environment with the 'task(3).py' file and its execution in the terminal.

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.

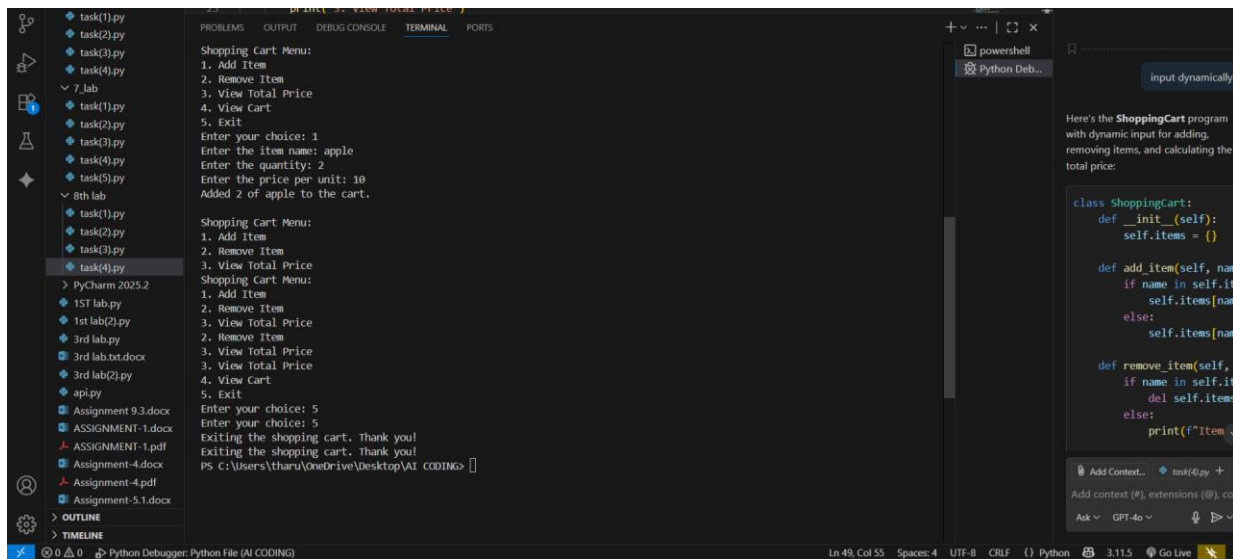
Prompt: Now generate a Python class `ShoppingCart` that can add items, remove items, and calculate the total price according to the given test cases. Take input dynamically from the user. Input: Add Apple with quantity 2 and price 3.0 → Output: `{"Apple": {"quantity": 2, "price": 3.0}}`. Input: Remove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

```
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
20 if __name__ == "__main__":
21     cart = ShoppingCart()
22     while True:
23         print("\nShopping Cart Menu:")
24         print("1. Add Item")
25         print("2. Remove Item")
26         print("3. View Total Price")
27         print("4. View Cart")
28         print("5. Exit")
29         choice = input("Enter your choice: ")
30         if choice == "1":
31             name = input("Enter the item name: ")
32             quantity = int(input("Enter the quantity: "))
33             price = float(input("Enter the price per unit: "))
34             cart.add_item(name, quantity, price)
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: {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.")
```

```
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: {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:

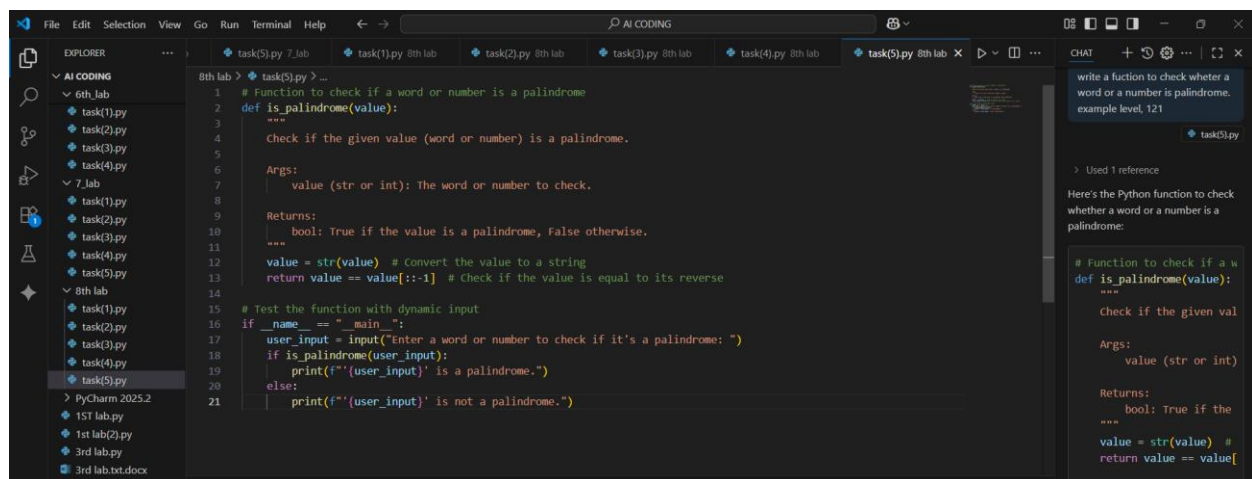


Write tests for a palindrome checker (e.g., `is_palindrome("level")` → `True`). Let Copilot suggest the function based on test case expectations.

Prompt:

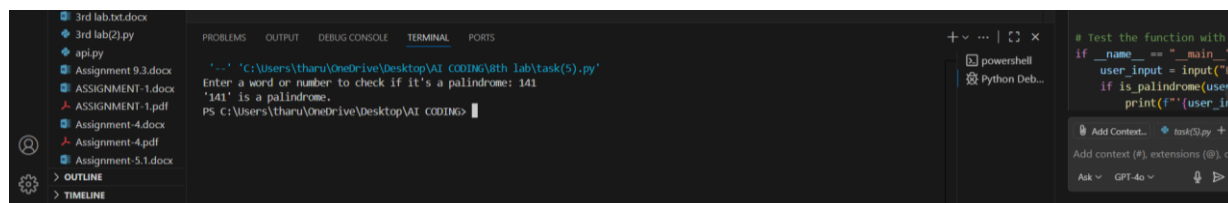
write a function to check whether a word or a number is palindrome or not. Example: level, 121

Code:



```
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>
C:\Users\tharu\OneDrive\Desktop\VAI CODING\ath lab\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



