Lab Assignment-8.3

2303A51546

Batch-10

TASK-1:

Prompt:

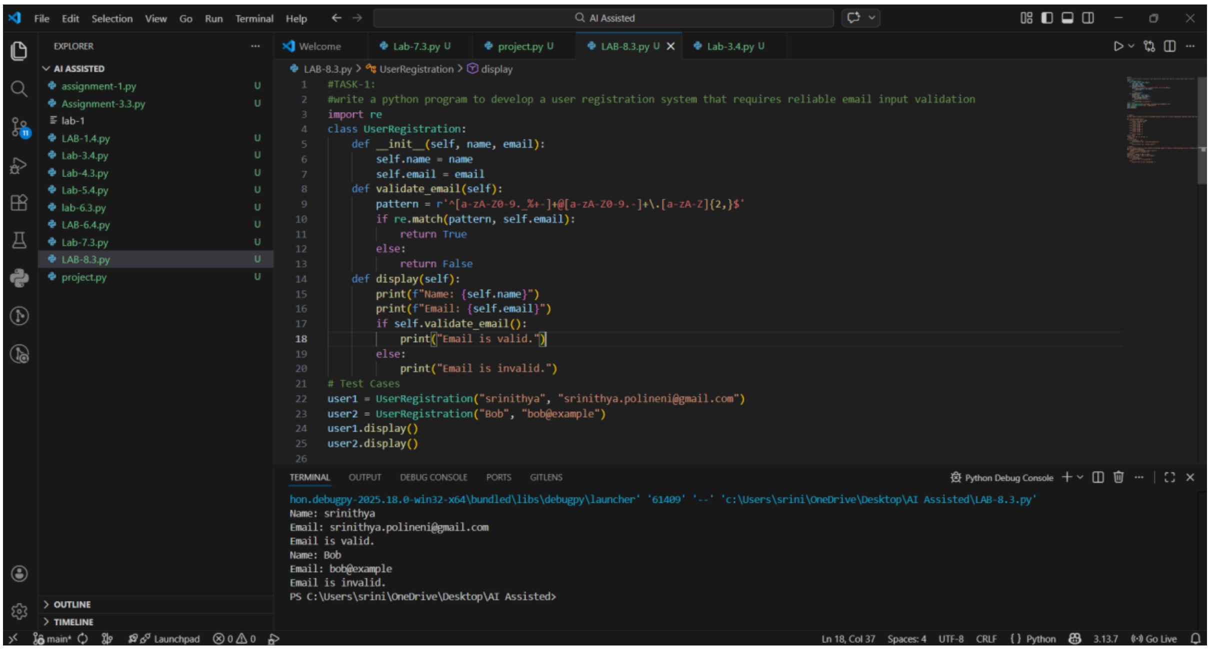
write a python program to develop a user registration system that requires reliable email input validation

CODE:

```
import re

class UserRegistration:

  def __init__(self, name, email):

    self.name = name

    self.email = email

  def validate_email(self):

    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'

    if re.match(pattern, self.email):

      return True

    else:

      return False

  def display(self):

    print(f"Name: {self.name}")

    print(f"Email: {self.email}")

    if self.validate_email():

      print("Email is valid.")

    else:

      print("Email is invalid.")

# Test Cases

user1 = UserRegistration("srinithya", "srinithya.polineni@gmail.com")
```

```
user2 = UserRegistration("Bob", "bob@example")

user1.display()

user2.display()
```

Output:



```
#TASK-1:
#write a python program to develop a user registration system that requires reliable email input validation
import re
class UserRegistration:
    def __init__(self, name, email):
        self.name = name
        self.email = email
    def validate_email(self):
        pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
        if re.match(pattern, self.email):
            return True
        else:
            return False
    def display(self):
        print(f"Name: {self.name}")
        print(f"Email: {self.email}")
        if self.validate_email():
            print("Email is valid.")
        else:
            print("Email is invalid.")
# Test Cases
user1 = UserRegistration("srinithya", "srinithya.polineni@gmail.com")
user2 = UserRegistration("Bob", "bob@example")
user1.display()
user2.display()
```

```
hon.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61409' '--' 'c:\Users\srini\OneDrive\Desktop\AI Assisted\LAB-8.3.py'
Name: srinithya
Email: srinithya.polineni@gmail.com
Email is valid.
Name: Bob
Email: bob@example
Email is invalid.
PS C:\Users\srini\OneDrive\Desktop\AI Assisted>
```

Analysis:

In this task, a user registration system is implemented to validate email addresses. A class is used to store user details such as name and email. The email validation is done using a regular expression, which checks whether the email follows a proper format containing @ and .. The program correctly identifies valid and invalid email addresses and displays the result. This approach ensures reliable email input validation.

TASK-2:

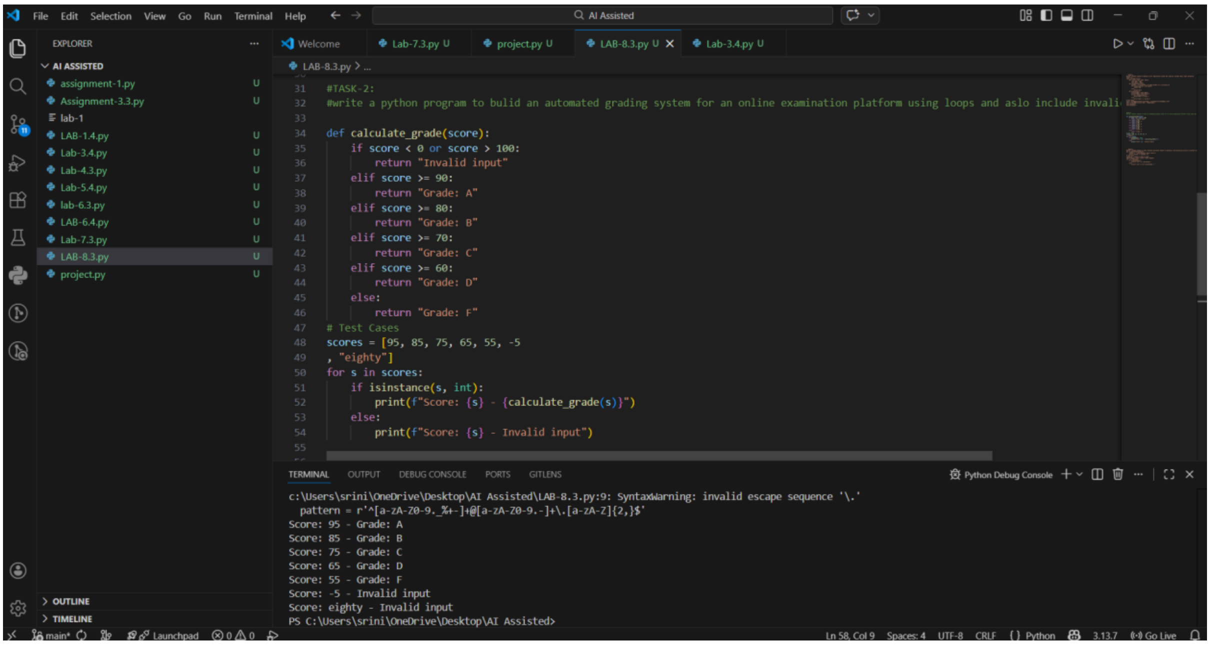
Prompt:

write a python program to bulid an automated grading system for an online examination platform using loops and aslo include invalid input like -5,eighty

CODE:

```
def calculate_grade(score):

  if score < 0 or score > 100:

    return "Invalid input"

  elif score >= 90:
        return "Grade: A"
    elif score >= 80:
        return "Grade: B"
    elif score >= 70:
        return "Grade: C"
    elif score >= 60:
        return "Grade: D"
    else:
        return "Grade: F"
# Test Cases
scores = [95, 85, 75, 65, 55, -5
, "eighty"]
for s in scores:
    if isinstance(s, int):
        print(f"Score: {s} - {calculate_grade(s)}")
    else:
        print(f"Score: {s} - Invalid input")
```

Output:



```
#TASK-2:
#write a python program to bulid an automated grading system for an online examination platform using loops and aslo include invali

def calculate_grade(score):
    if score < 0 or score > 100:
        return "Invalid input"
    elif score >= 90:
        return "Grade: A"
    elif score >= 80:
        return "Grade: B"
    elif score >= 70:
        return "Grade: C"
    elif score >= 60:
        return "Grade: D"
    else:
        return "Grade: F"
# Test Cases
scores = [95, 85, 75, 65, 55, -5
, "eighty"]
for s in scores:
    if isinstance(s, int):
        print(f"Score: {s} - {calculate_grade(s)}")
    else:
        print(f"Score: {s} - Invalid input")
```

```
c:\Users\srini\OneDrive\Desktop\AI Assisted\LAB-8.3.py:9: SyntaxWarning: invalid escape sequence '\.'
  pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
Score: 95 - Grade: A
Score: 85 - Grade: B
Score: 75 - Grade: C
Score: 65 - Grade: D
Score: 55 - Grade: F
Score: -5 - Invalid input
Score: eighty - Invalid input
PS C:\Users\srini\OneDrive\Desktop\AI Assisted>
```

Analysis:

This task focuses on building an automated grading system using conditional statements and loops. The program assigns grades based on the score ranges provided. Boundary values are handled correctly, and invalid inputs such as negative numbers and non-numeric values are detected and handled safely. This prevents incorrect grading and improves the robustness of the program.

TASK-3:

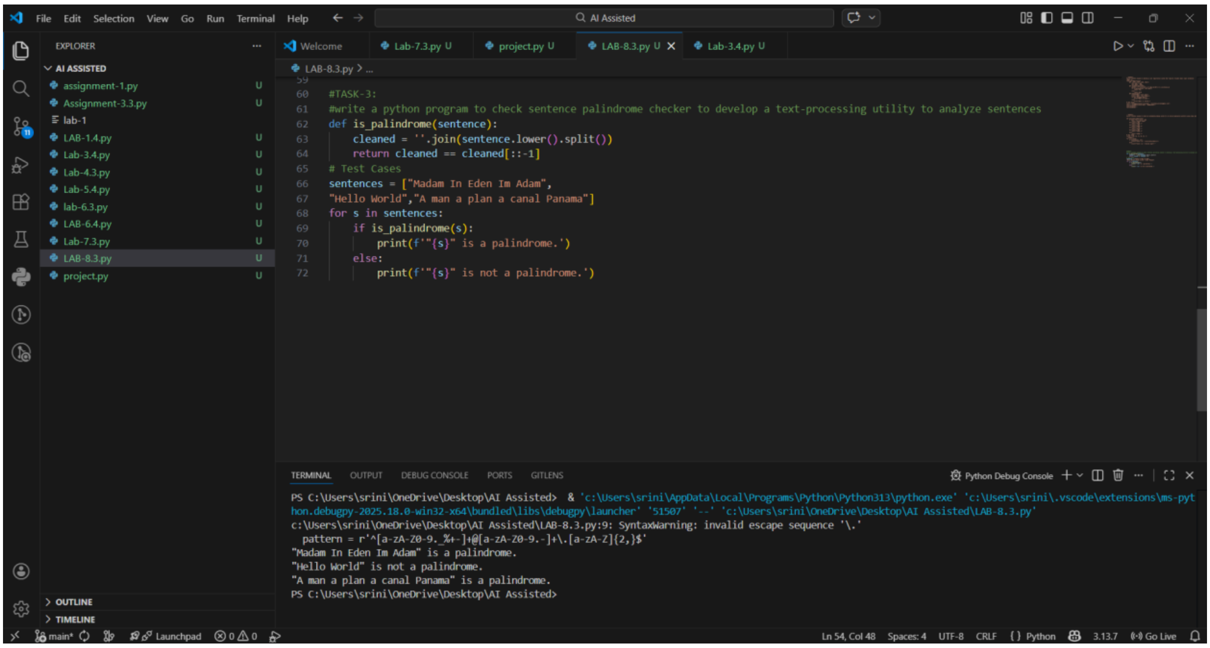
Prompt:

write a python program to check sentence palindrome checker to develop a text-processing utility to analyze sentences

CODE:

```
def is_palindrome(sentence):
    cleaned = ''.join(sentence.lower().split())
    return cleaned == cleaned[::-1]
# Test Cases
sentences = ["Madam In Eden Im Adam",
"Hello World","A man a plan a canal Panama"]
for s in sentences:
    if is_palindrome(s):
        print(f'"{s}" is a palindrome.')
    else:
        print(f'"{s}" is not a palindrome.')
```

Output:



```python
#TASK-3:
#write a python program to check sentence palindrome checker to develop a text-processing utility to analyze sentences
def is_palindrome(sentence):
    cleaned = ''.join(sentence.lower().split())
    return cleaned == cleaned[::-1]
# Test Cases
sentences = ["Madam In Eden Im Adam",
"Hello World","A man a plan a canal Panama"]
for s in sentences:
    if is_palindrome(s):
        print(f'"{s}" is a palindrome.')
    else:
        print(f'"{s}" is not a palindrome.')
```

```
PS C:\Users\srini\OneDrive\Desktop\AI Assisted>  & 'c:\Users\srini\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51507' '--' 'c:\Users\srini\OneDrive\Desktop\AI Assisted\LAB-8.3.py'
c:\Users\srini\OneDrive\Desktop\AI Assisted\LAB-8.3.py:9: SyntaxWarning: invalid escape sequence '\.'
  pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
"Madam In Eden Im Adam" is a palindrome.
"Hello World" is not a palindrome.
"A man a plan a canal Panama" is a palindrome.
PS C:\Users\srini\OneDrive\Desktop\AI Assisted>
```

Analysis:

In this task, a function is used to check whether a given sentence is a palindrome. The sentence is converted to lowercase and spaces are removed before comparison. The cleaned string is then compared with its reverse to determine whether it is a palindrome. The program successfully identifies both palindromic and non-palindromic sentences.

TASK-4:

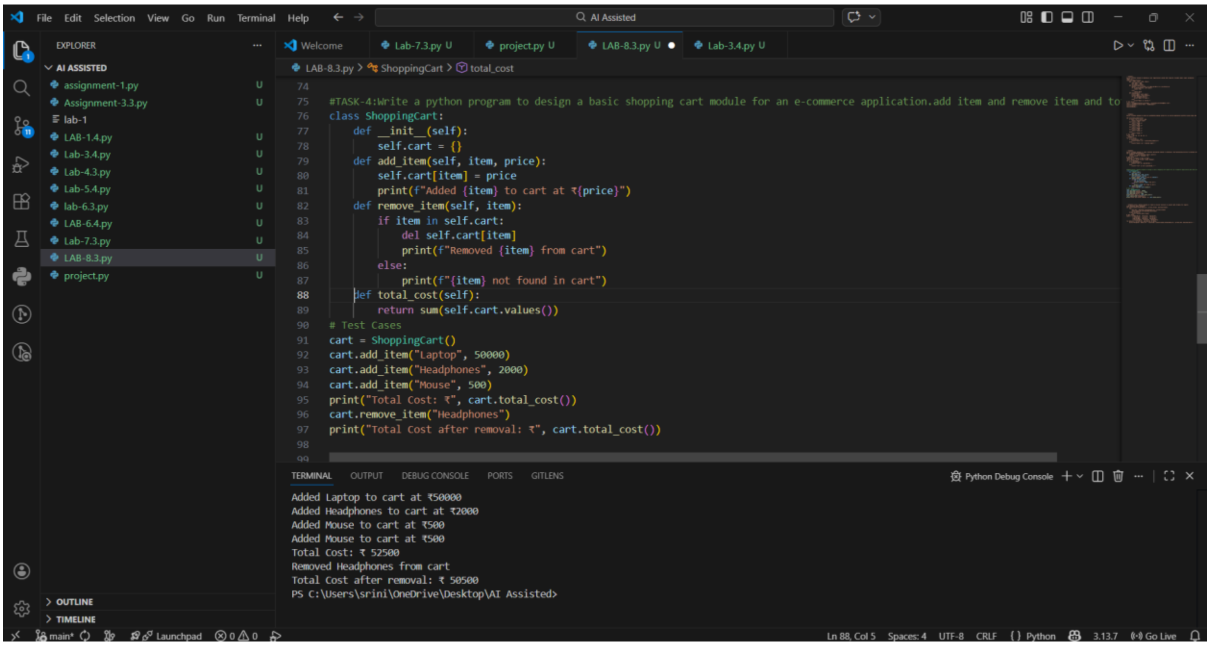
Prompt:

Write a python program to design a basic shopping cart module for an e-commerce application.add item and remove item and total cost

CODE:

```
class ShoppingCart:

  def __init__(self):

    self.cart = {}

  def add_item(self, item, price):

    self.cart[item] = price

    print(f"Added {item} to cart at ₹{price}")

  def remove_item(self, item):

    if item in self.cart:
            del self.cart[item]

            print(f"Removed {item} from cart")

        else:

            print(f"{item} not found in cart")

    def total_cost(self):

        return sum(self.cart.values())

# Test Cases

cart = ShoppingCart()

cart.add_item("Laptop", 50000)

cart.add_item("Headphones", 2000)

cart.add_item("Mouse", 500)

print("Total Cost: ₹", cart.total_cost())

cart.remove_item("Headphones")

print("Total Cost after removal: ₹", cart.total_cost())
```

Output:



```
#TASK-4:Write a python program to design a basic shopping cart module for an e-commerce application.add item and remove item and to
class ShoppingCart:
    def __init__(self):
        self.cart = {}
    def add_item(self, item, price):
        self.cart[item] = price
        print(f"Added {item} to cart at ₹{price}")
    def remove_item(self, item):
        if item in self.cart:
            del self.cart[item]
            print(f"Removed {item} from cart")
        else:
            print(f"{item} not found in cart")
    def total_cost(self):
        return sum(self.cart.values())
# Test Cases
cart = ShoppingCart()
cart.add_item("Laptop", 50000)
cart.add_item("Headphones", 2000)
cart.add_item("Mouse", 500)
print("Total Cost: ₹", cart.total_cost())
cart.remove_item("Headphones")
print("Total Cost after removal: ₹", cart.total_cost())
```

```
Added Laptop to cart at ₹50000
Added Headphones to cart at ₹2000
Added Mouse to cart at ₹500
Added Mouse to cart at ₹500
Total Cost: ₹ 52500
Removed Headphones from cart
Total Cost after removal: ₹ 50500
PS C:\Users\srini\OneDrive\Desktop\AI Assisted>
```

Analysis:

This task implements a basic shopping cart system using a class. A dictionary is used to store items and their prices. The program allows adding items, removing items, and calculating the total cost. It correctly updates the cart after each operation and handles cases where an item does not exist in the cart. This demonstrates the use of object-oriented programming concepts.

TASK-5:

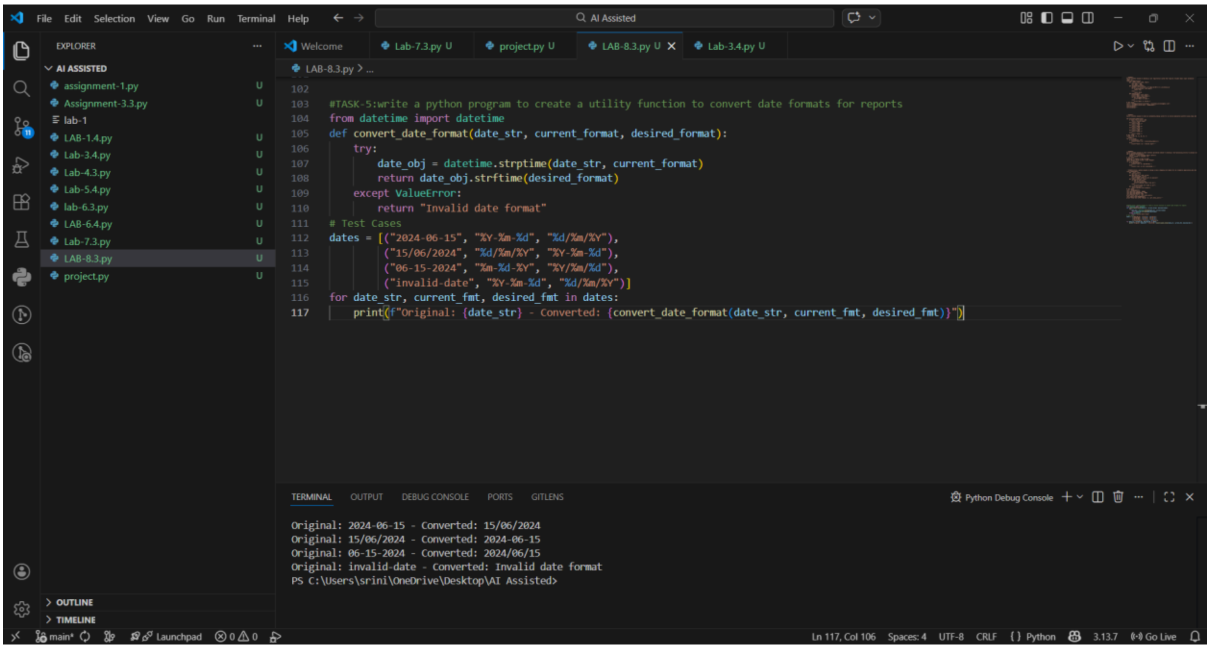
Prompt:

write a python program to create a utility function to convert date formats for reports

CODE:

```
from datetime import datetime

def convert_date_format(date_str, current_format, desired_format):

    try:

        date_obj = datetime.strptime(date_str, current_format)

        return date_obj.strftime(desired_format)

    except ValueError:

        return "Invalid date format"

# Test Cases

dates = [("2024-06-15", "%Y-%m-%d", "%d/%m/%Y"),

         ("15/06/2024", "%d/%m/%Y", "%Y-%m-%d"),

         ("06-15-2024", "%m-%d-%Y", "%Y/%m/%d"),

         ("invalid-date", "%Y-%m-%d", "%d/%m/%Y")]

for date_str, current_fmt, desired_fmt in dates:

    print(f"Original: {date_str} - Converted: {convert_date_format(date_str, current_fmt, desired_fmt)}")
```

Output:



```python

#TASK-5:write a python program to create a utility function to convert date formats for reports
from datetime import datetime
def convert_date_format(date_str, current_format, desired_format):
    try:
        date_obj = datetime.strptime(date_str, current_format)
        return date_obj.strftime(desired_format)
    except ValueError:
        return "Invalid date format"
# Test Cases
dates = [("2024-06-15", "%Y-%m-%d", "%d/%m/%Y"),
         ("15/06/2024", "%d/%m/%Y", "%Y-%m-%d"),
         ("06-15-2024", "%m-%d-%Y", "%Y/%m/%d"),
         ("invalid-date", "%Y-%m-%d", "%d/%m/%Y")]
for date_str, current_fmt, desired_fmt in dates:
    print(f"Original: {date_str} - Converted: {convert_date_format(date_str, current_fmt, desired_fmt)}")
```

```
Original: 2024-06-15 - Converted: 15/06/2024
Original: 15/06/2024 - Converted: 2024-06-15
Original: 06-15-2024 - Converted: 2024/06/15
Original: invalid-date - Converted: Invalid date format
PS C:\Users\srini\OneDrive\Desktop\AI Assisted>
```

Analysis:

In this task, a utility function is created to convert date formats using the datetime module. The program converts a date from one format to another and uses exception handling to manage invalid date inputs. This ensures that the program does not crash and provides meaningful output for incorrect formats.