Assignment -8.3

2303a51343

Batch-10

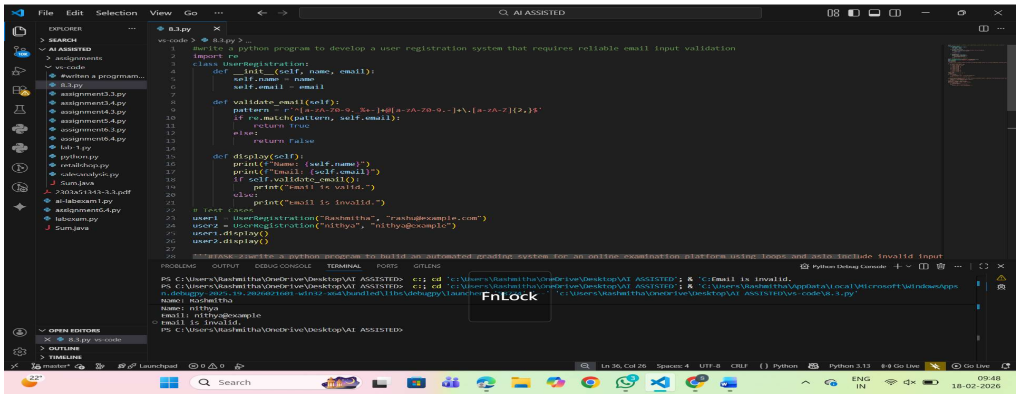
TASK-1

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("Rashmitha", "rashu@example.com")
user2 = UserRegistration("nithya", "nithya@example")
user1.display()
user2.display()
```

Output:



```
#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("Rashmitha", "rashu@example.com")
user2 = UserRegistration("nithya", "nithya@example")
user1.display()
user2.display()

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

```
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED>  c:; cd 'c:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED'; & 'C:Email is invalid.
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED>  c:; cd 'c:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED'; & 'C:\Users\Rashmitha\AppData\Local\Microsoft\WindowsApps
n.debugpy-2025.19.2026021601-win32-x64\bundled\libs\debugpy\launcher [illegible] '--' 'c:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED\vs-code\8.3.py'
Name: Rashmitha
Name: nithya
Email: nithya@example
Email is invalid.
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED>
```

Analysis: This program creates a User Registration class to store a user's name and email address. It checks whether the email is written in a correct format using a pattern. The program uses the re module to compare the email with this pattern. If the email follows the correct format, it prints that the email is valid; otherwise, it prints that the email is invalid.

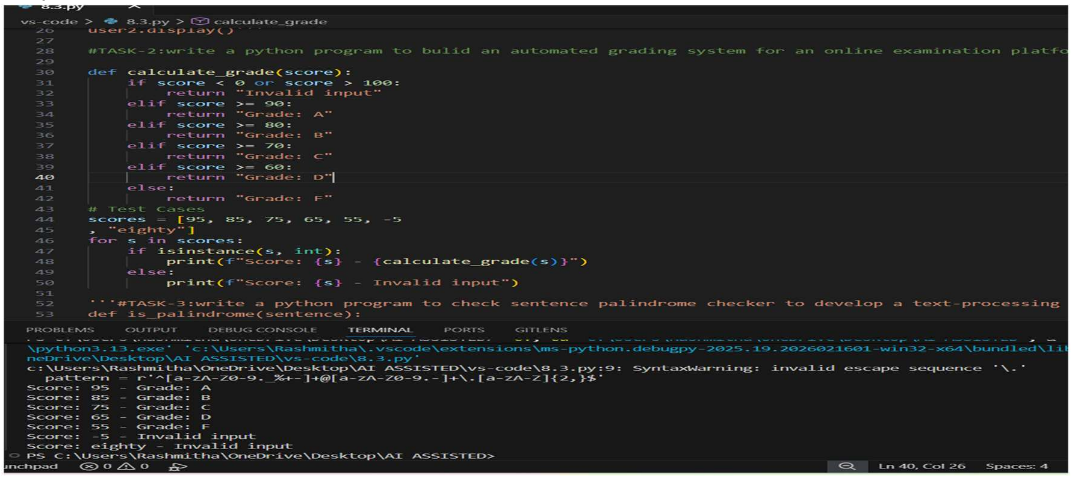
TASK-2

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"
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:



```
vs-code > 8.3.py > calculate_grade
user2.display()'''

#TASK-2:write a python program to bulid an automated grading system for an online examination platfo

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")

'''#TASK-3:write a python program to check sentence palindrome checker to develop a text-processing
def is_palindrome(sentence):
```

```
\python3.13.exe' 'c:\Users\Rashmitha\.vscode\extensions\ms-python.debugpy-2025.19.2026021601-win32-x64\bundled\li
neDrive\Desktop\AI ASSISTED\vs-code\8.3.py'
c:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED\vs-code\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\Rashmitha\OneDrive\Desktop\AI ASSISTED>
```

Analysis: This program creates an automated grading system using a function and a loop. It assigns grades based on score ranges and checks for invalid values like negative numbers or non-numeric inputs. The loop processes each score and prints the result. Invalid inputs such as -5 and "eighty" are handled safely without errors.

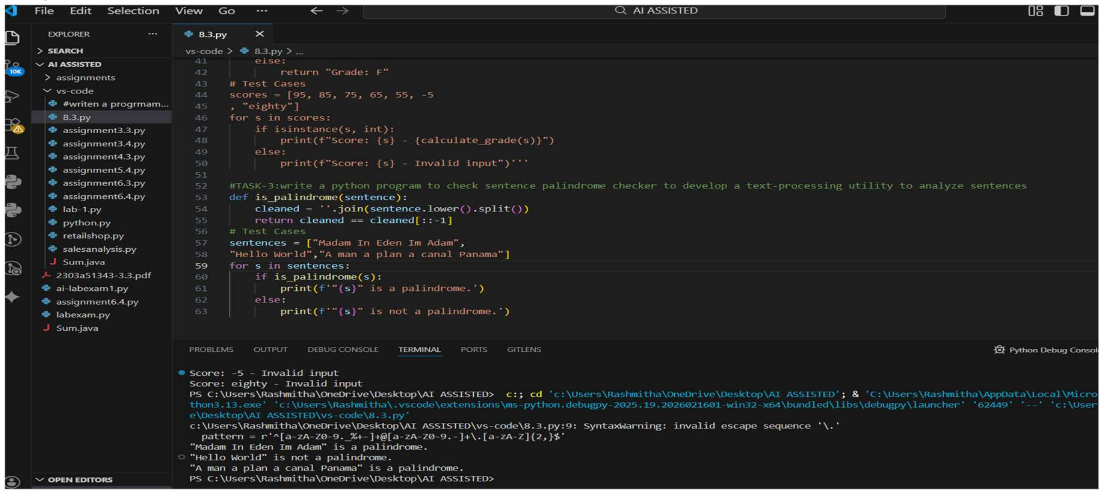
TASK-3

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:



```
    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")'''

#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.')
```

```
Score: -5 - Invalid input
Score: eighty - Invalid input
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED>  c:; cd 'c:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED'; & 'C:\Users\Rashmitha\AppData\Local\Micro
thon3.13.exe' 'c:\Users\Rashmitha\.vscode\extensions\ms-python.debugpy-2025.19.2026021601-win32-x64\bundled\libs\debugpy\launcher' '62449' '--' 'c:\User
e\Desktop\AI ASSISTED\vs-code\8.3.py'
c:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED\vs-code\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\Rashmitha\OneDrive\Desktop\AI ASSISTED>
```

Analysis: This program checks if a sentence is a palindrome. It changes the sentence to lowercase and removes spaces, then compares it with its reverse. If both are the same, it prints that the sentence is a palindrome; otherwise, it prints that it is not.

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)

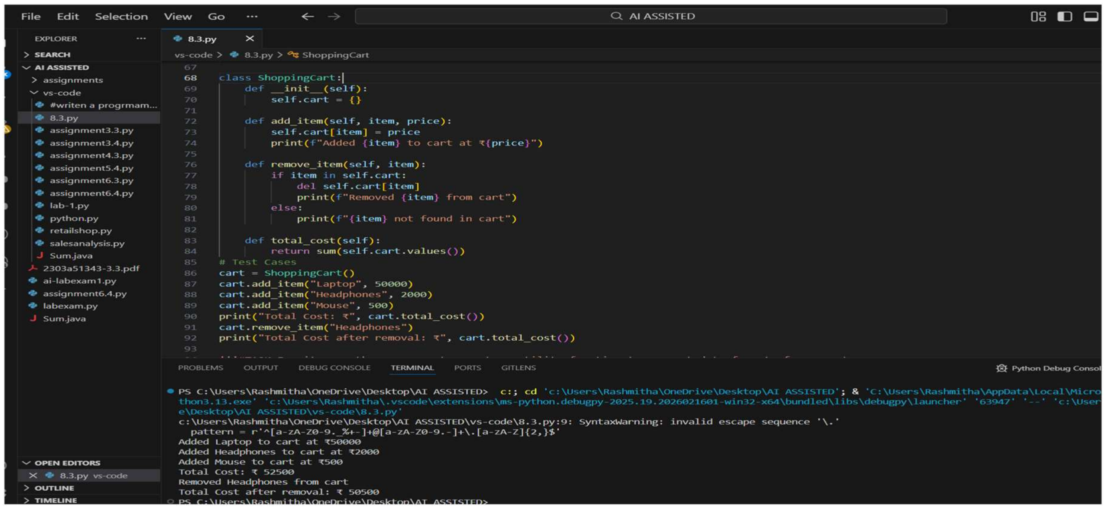
cart.add_item("Mouse", 500)

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

cart.remove_item("Headphones")

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

Output:



```
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())
```

```
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED>  c:; cd 'c:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED'; & 'C:\Users\Rashmitha\AppData\Local\Micro
thon3.13.exe' 'c:\Users\Rashmitha\.vscode\extensions\ms-python.debugpy-2025.19.2026021601-win32-x64\bundled\libs\debugpy\launcher' '63947' '--' 'c:\User
e\Desktop\AI ASSISTED\vs-code\8.3.py'
c:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED\vs-code\8.3.py:9: SyntaxWarning: invalid escape sequence '\.'
  pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
Added Laptop to cart at ₹50000
Added Headphones to cart at ₹2000
Added Mouse to cart at ₹500
Total Cost: ₹ 52500
Removed Headphones from cart
Total Cost after removal: ₹ 50500
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED>
```

Analysis: This program creates a ShoppingCart class to add and remove items with their prices. It keeps track of items in a dictionary and calculates the total cost. The test shows adding items, removing one, and updating the total cost correctly.

TASK-5

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

from datetime import datetime

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"

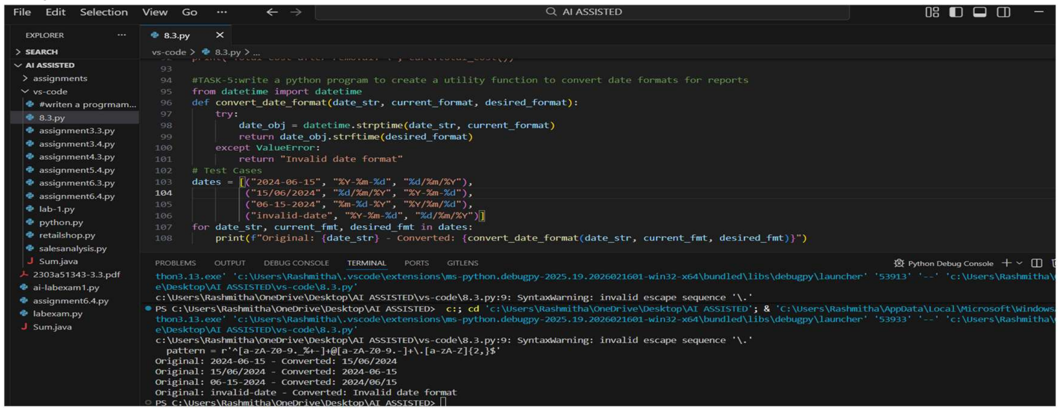
Test Cases

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:



```
#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)}")
```

```
thon3.13.exe' 'c:\Users\Rashmitha\.vscode\extensions\ms-python.debugpy-2025.19.2026021601-win32-x64\bundled\libs\debugpy\launcher' '53913' '--' 'c:\Users\Rashmitha\
e\Desktop\AI ASSISTED\vs-code\8.3.py'
c:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED\vs-code\8.3.py:9: SyntaxWarning: invalid escape sequence '\.'
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED>  c:; cd 'c:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED'; & 'C:\Users\Rashmitha\AppData\Local\Microsoft\Windows
thon3.13.exe' 'c:\Users\Rashmitha\.vscode\extensions\ms-python.debugpy-2025.19.2026021601-win32-x64\bundled\libs\debugpy\launcher' '53933' '--' 'c:\Users\Rashmitha\
e\Desktop\AI ASSISTED\vs-code\8.3.py'
c:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED\vs-code\8.3.py:9: SyntaxWarning: invalid escape sequence '\.'
  pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
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\Rashmitha\OneDrive\Desktop\AI ASSISTED>
```

Analysis: This program defines a function to convert dates from one format to another using Python's datetime module. It tries to parse the input date string with the current format and then outputs it in the desired format. If the input date doesn't match the format, it returns "Invalid date format." The test cases show successful conversions and correctly handle invalid inputs like "invalid-date"**.**