ASSIGNMENT – 8.3

2303A51355

Batch-10

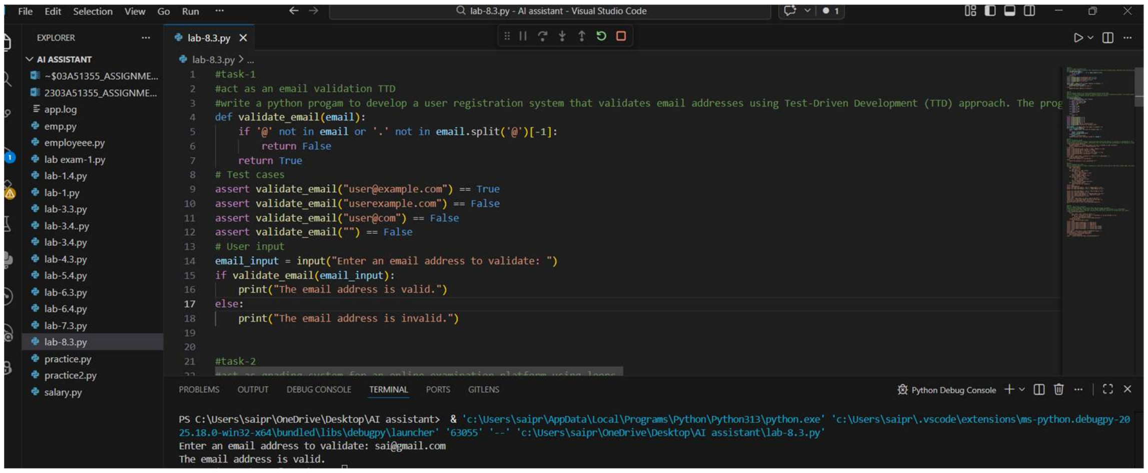
Task-1

Prompt: act as an email validation TTD

#write a python progam to develop a user registration system that validates email addresses using TTD approach. The program should include a function to validate email addresses based on common email formatting rules (e.g., presence of '@' and domain).

code :

```
def validate_email(email):
        if '@' not in email or '.' not in
email.split('@')[-1]:
       return False
    return True
# Test cases
assert validate_email("user@example.com")
== True
assert validate_email("userexample.com") ==
False
assert validate_email("user@com") == False
assert validate_email("") == False
# User input
email_input = input("Enter an email address
to validate: ")
if validate_email(email_input):
    print("The email address is valid.")
else:
    print("The email address is invalid.")
```



```python
#task-1
#act as an email validation TTD
#write a python progam to develop a user registration system that validates email addresses using Test-Driven Development (TTD) approach. The prog
def validate_email(email):
    if '@' not in email or '.' not in email.split('@')[-1]:
        return False
    return True
# Test cases
assert validate_email("user@example.com") == True
assert validate_email("userexample.com") == False
assert validate_email("user@com") == False
assert validate_email("") == False
# User input
email_input = input("Enter an email address to validate: ")
if validate_email(email_input):
    print("The email address is valid.")
else:
    print("The email address is invalid.")


#task-2
```

```
PS C:\Users\saipr\OneDrive\Desktop\AI assistant>  & 'c:\Users\saipr\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\saipr\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '63055' '--' 'c:\Users\saipr\OneDrive\Desktop\AI assistant\lab-8.3.py'
Enter an email address to validate: sai@gmail.com
The email address is valid.
```

Code Analysis:

- The program defines a function validate_email() to check whether an email contains @ and a valid domain.
- Test cases are written using assert to verify correct and incorrect email formats.
- The function returns True for valid emails and False for invalid ones.
- User input is taken and checked using the same validation function.
- This approach follows Test-Driven Development (TDD) by testing before final usage.

Task-2

Prompt: act as grading system for an online examination platform .

# Write a Python program that simulates a grading system for an online examination platform.

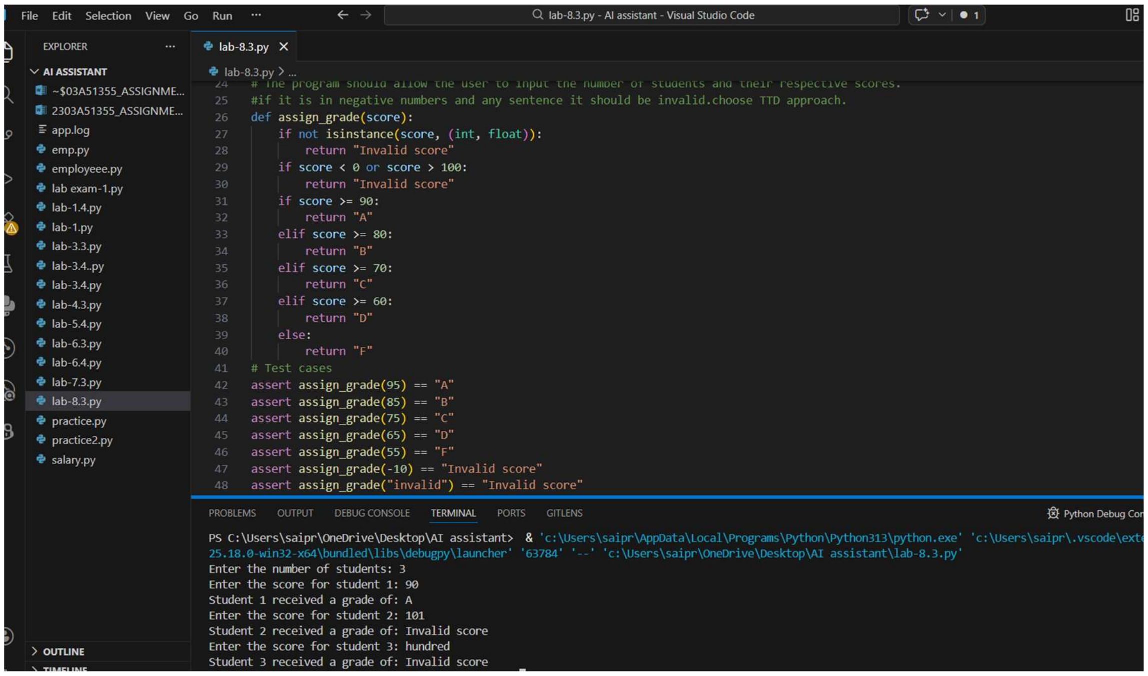
# The program should allow the user to input the number of students and their respective scores.

#if it is in negative numbers and any sentence it should be invalid.choose TTD approach.

Code :

```python
def assign_grade(score):
    if not isinstance(score, (int, float)):
        return "Invalid score"
    if score < 0 or score > 100:
        return "Invalid score"
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "C"
    elif score >= 60:
        return "D"
    else:
        return "F"
# Test cases
assert assign_grade(95) == "A"
assert assign_grade(85) == "B"
assert assign_grade(75) == "C"
assert assign_grade(65) == "D"
assert assign_grade(55) == "F"
assert assign_grade(-10) == "Invalid score"
assert assign_grade("invalid") == "Invalid score"
# User input
num_students = int(input("Enter the number of students: "))
for i in range(num_students):
    score = input(f"Enter the score for student {i+1}: ")
    try:
        score = float(score)
        grade = assign_grade(score)
    except ValueError:
        grade = "Invalid score"
```

```python
print(f"Student {i+1} received a grade of: {grade}")
```



```python
# The program should allow the user to input the number of students and their respective scores.
#if it is in negative numbers and any sentence it should be invalid.choose TTD approach.
def assign_grade(score):
    if not isinstance(score, (int, float)):
        return "Invalid score"
    if score < 0 or score > 100:
        return "Invalid score"
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "C"
    elif score >= 60:
        return "D"
    else:
        return "F"
# Test cases
assert assign_grade(95) == "A"
assert assign_grade(85) == "B"
assert assign_grade(75) == "C"
assert assign_grade(65) == "D"
assert assign_grade(55) == "F"
assert assign_grade(-10) == "Invalid score"
assert assign_grade("invalid") == "Invalid score"
```

```
PS C:\Users\saipr\OneDrive\Desktop\AI assistant>  & 'c:\Users\saipr\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\saipr\.vscode\exte
25.18.0-win32-x64\bundled\libs\debugpy\launcher' '63784' '--' 'c:\Users\saipr\OneDrive\Desktop\AI assistant\lab-8.3.py'
Enter the number of students: 3
Enter the score for student 1: 90
Student 1 received a grade of: A
Enter the score for student 2: 101
Student 2 received a grade of: Invalid score
Enter the score for student 3: hundred
Student 3 received a grade of: Invalid score
```

Code Analysis:

- The function assign_grade() assigns grades based on the student's score.
- Invalid inputs such as negative values or non-numeric inputs are handled safely.
- Test cases verify all grading conditions including invalid inputs.
- A loop is used to accept scores of multiple students.
- The program prints grades for each student, ensuring accurate evaluation.

Task-3

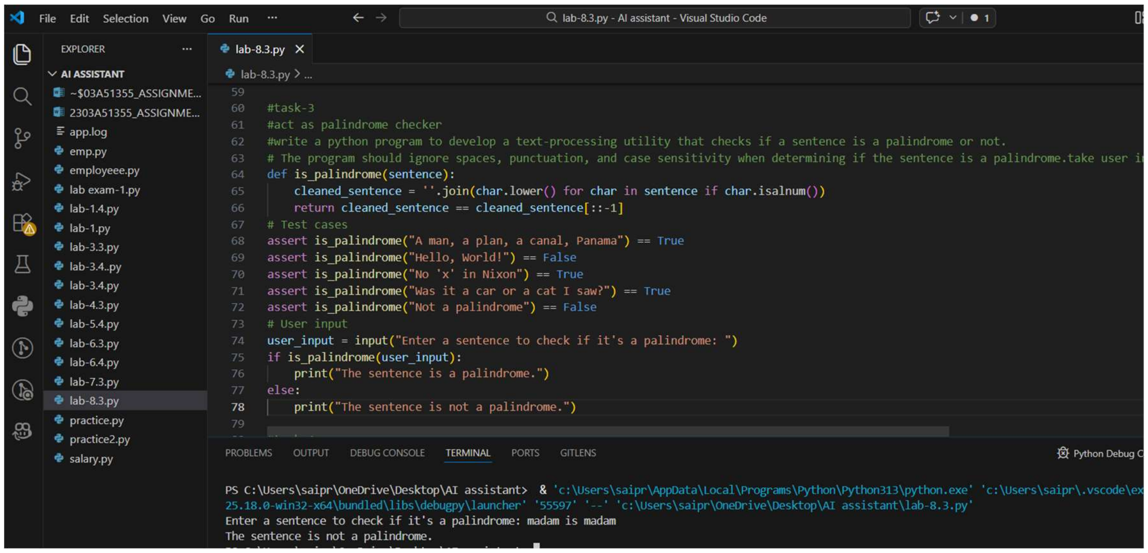
Prompt: act as palindrome checker

#write a python program to develop a text-processing utility that checks if a sentence is a palindrome or not.

The program should ignore spaces, punctuation, and case sensitivity when determining if the sentence is a palindrome.take user input and display appropriate messages based on the palindrome check results.

Code :

```python
def is_palindrome(sentence):
    cleaned_sentence = ''.join(char.lower() for char in sentence if char.isalnum())
    return cleaned_sentence == cleaned_sentence[::-1]
# Test cases
assert is_palindrome("A man, a plan, a canal, Panama") == True
assert is_palindrome("Hello, World!") == False
assert is_palindrome("No 'x' in Nixon") == True
assert is_palindrome("Was it a car or a cat I saw?") == True
assert is_palindrome("Not a palindrome") == False
# User input
user_input = input("Enter a sentence to check if it's a palindrome: ")
if is_palindrome(user_input):
    print("The sentence is a palindrome.")
else:
    print("The sentence is not a palindrome.")
```



```python
#task-3
#act as palindrome checker
#write a python program to develop a text-processing utility that checks if a sentence is a palindrome or not.
# The program should ignore spaces, punctuation, and case sensitivity when determining if the sentence is a palindrome.take user i
def is_palindrome(sentence):
    cleaned_sentence = ''.join(char.lower() for char in sentence if char.isalnum())
    return cleaned_sentence == cleaned_sentence[::-1]
# Test cases
assert is_palindrome("A man, a plan, a canal, Panama") == True
assert is_palindrome("Hello, World!") == False
assert is_palindrome("No 'x' in Nixon") == True
assert is_palindrome("Was it a car or a cat I saw?") == True
assert is_palindrome("Not a palindrome") == False
# User input
user_input = input("Enter a sentence to check if it's a palindrome: ")
if is_palindrome(user_input):
    print("The sentence is a palindrome.")
else:
    print("The sentence is not a palindrome.")
```

```
PS C:\Users\saipr\OneDrive\Desktop\AI assistant>  & 'c:\Users\saipr\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\saipr\.vscode\ex
25.18.0-win32-x64\bundled\libs\debugpy\launcher' '55597' '--' 'c:\Users\saipr\OneDrive\Desktop\AI assistant\lab-8.3.py'
Enter a sentence to check if it's a palindrome: madam is madam
The sentence is not a palindrome.
```

Code Analysis :

 The function is_palindrome() removes spaces, punctuation, and converts text to lowercase.

 It checks whether the processed string is equal to its reverse.

 Test cases verify multiple palindrome and non-palindrome sentences.

 The program accepts a sentence from the user.

 It displays whether the given sentence is a palindrome or not.

Task-4

Prompt: act as shopping cart class

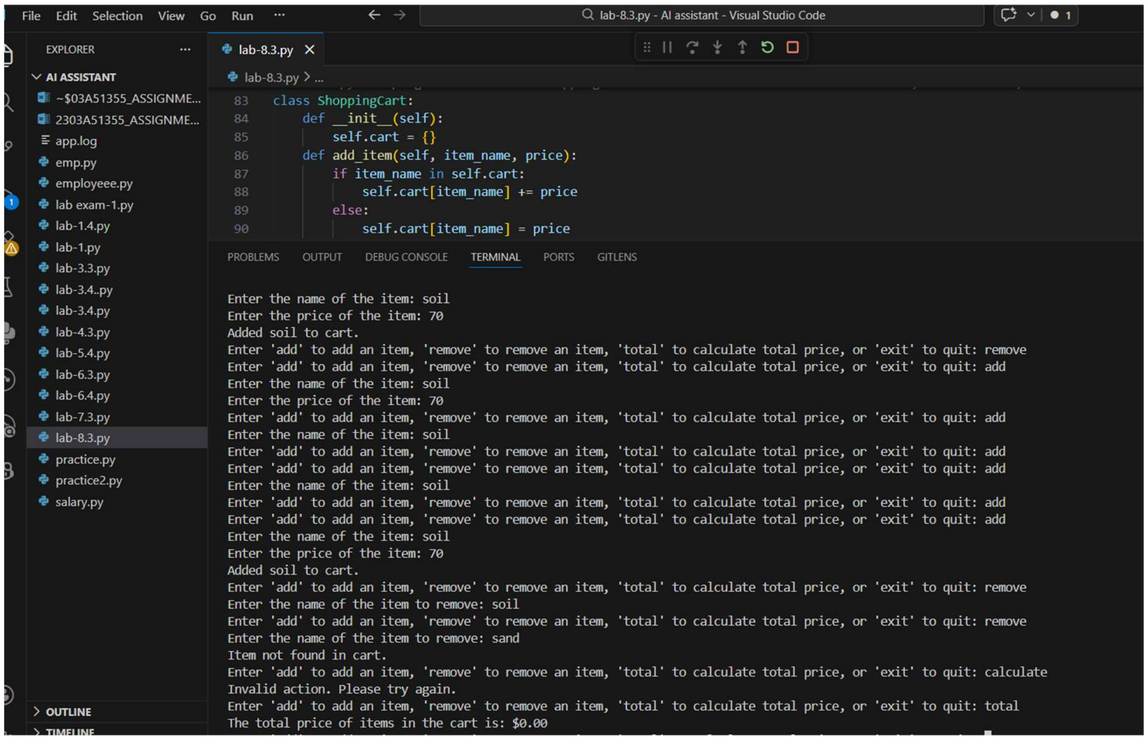
#write a python program to create a shopping cart class that allows users to add items, remove items, and calculate the total price of the items in the cart. The class should have methods for each of these functionalities. take user input.

Code :

```
class ShoppingCart:
  def __init__(self):
    self.cart = {}
  def add_item(self, item_name, price):
    if item_name in self.cart:
      self.cart[item_name] += price
    else:
      self.cart[item_name] = price
  def remove_item(self, item_name):
    if item_name in self.cart:
      del self.cart[item_name]
    else:
      print("Item not found in cart.")
  def calculate_total(self):
    return sum(self.cart.values())
```

```
# User input
cart = ShoppingCart()
while True:
    action = input("Enter 'add' to add an item, 'remove' to
remove an item, 'total' to calculate total price, or 'exit' to quit:
")
    if action == 'add':
        item_name = input("Enter the name of the item: ")
        try:
            price = float(input("Enter the price of the item: "))
            cart.add_item(item_name, price)
            print(f"Added {item_name} to cart.")
        except ValueError:
            print("Invalid price. Please enter a number.")
    elif action == 'remove':
        item_name = input("Enter the name of the item to
remove: ")
        cart.remove_item(item_name)
    elif action == 'total':
        total_price = cart.calculate_total()
        print(f"The total price of items in the cart is:
${total_price:.2f}")
    elif action == 'exit':
        print("Exiting the shopping cart program.")
        break
    else:
        print("Invalid action. Please try again.")
```

Output:



```python
class ShoppingCart:
    def __init__(self):
        self.cart = {}
    def add_item(self, item_name, price):
        if item_name in self.cart:
            self.cart[item_name] += price
        else:
            self.cart[item_name] = price
```

```
Enter the name of the item: soil
Enter the price of the item: 70
Added soil to cart.
Enter 'add' to add an item, 'remove' to remove an item, 'total' to calculate total price, or 'exit' to quit: remove
Enter 'add' to add an item, 'remove' to remove an item, 'total' to calculate total price, or 'exit' to quit: add
Enter the name of the item: soil
Enter the price of the item: 70
Enter 'add' to add an item, 'remove' to remove an item, 'total' to calculate total price, or 'exit' to quit: add
Enter the name of the item: soil
Enter 'add' to add an item, 'remove' to remove an item, 'total' to calculate total price, or 'exit' to quit: add
Enter 'add' to add an item, 'remove' to remove an item, 'total' to calculate total price, or 'exit' to quit: add
Enter the name of the item: soil
Enter 'add' to add an item, 'remove' to remove an item, 'total' to calculate total price, or 'exit' to quit: add
Enter 'add' to add an item, 'remove' to remove an item, 'total' to calculate total price, or 'exit' to quit: add
Enter the name of the item: soil
Enter the price of the item: 70
Added soil to cart.
Enter 'add' to add an item, 'remove' to remove an item, 'total' to calculate total price, or 'exit' to quit: remove
Enter the name of the item to remove: soil
Enter 'add' to add an item, 'remove' to remove an item, 'total' to calculate total price, or 'exit' to quit: remove
Enter the name of the item to remove: sand
Item not found in cart.
Enter 'add' to add an item, 'remove' to remove an item, 'total' to calculate total price, or 'exit' to quit: calculate
Invalid action. Please try again.
Enter 'add' to add an item, 'remove' to remove an item, 'total' to calculate total price, or 'exit' to quit: total
The total price of items in the cart is: $0.00
```

Code Analysis :

- The ShoppingCart class manages items using a dictionary.
- Users can add items, remove items, and calculate the total price.
- The program uses a loop to repeatedly take user actions.
- Input validation prevents invalid price entries.
- The system provides an interactive shopping experience.

Task-5

Prompt: act as a date format converter

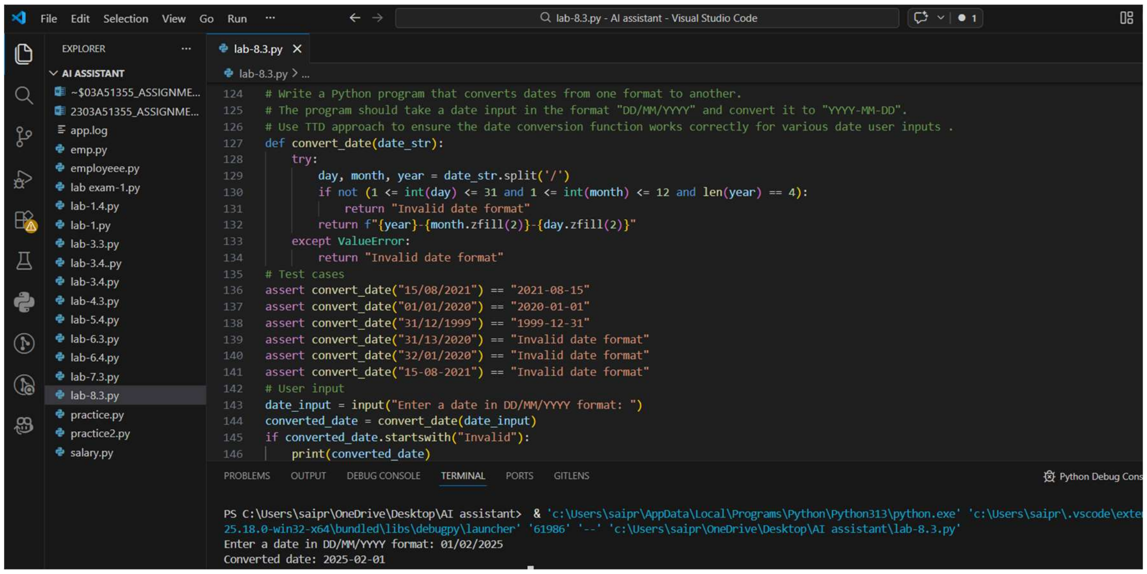
# Write a Python program that converts dates.The program should take a date input in the format "DD/MM/YYYY" and convert it to "YYYY-MM-DD".

# Use TTD approach

Code:

```python
def convert_date(date_str):
    try:
        day, month, year = date_str.split('/')
        if not (1 <= int(day) <= 31 and 1 <= int(month) <= 12 and len(year) == 4):
            return "Invalid date format"
        return f"{year}-{month.zfill(2)}-{day.zfill(2)}"
    except ValueError:
        return "Invalid date format"
# Test cases
assert convert_date("15/08/2021") == "2021-08-15"
assert convert_date("01/01/2020") == "2020-01-01"
assert convert_date("31/12/1999") == "1999-12-31"
assert convert_date("31/13/2020") == "Invalid date format"
assert convert_date("32/01/2020") == "Invalid date format"
assert convert_date("15-08-2021") == "Invalid date format"
# User input
date_input = input("Enter a date in DD/MM/YYYY format: ")
converted_date = convert_date(date_input)
if converted_date.startswith("Invalid"):
    print(converted_date)
else:    print(f"Converted date: {converted_date}")
```

Output :



```python
# Write a Python program that converts dates from one format to another.
# The program should take a date input in the format "DD/MM/YYYY" and convert it to "YYYY-MM-DD".
# Use TTD approach to ensure the date conversion function works correctly for various date user inputs .
def convert_date(date_str):
    try:
        day, month, year = date_str.split('/')
        if not (1 <= int(day) <= 31 and 1 <= int(month) <= 12 and len(year) == 4):
            return "Invalid date format"
        return f"{year}-{month.zfill(2)}-{day.zfill(2)}"
    except ValueError:
        return "Invalid date format"
# Test cases
assert convert_date("15/08/2021") == "2021-08-15"
assert convert_date("01/01/2020") == "2020-01-01"
assert convert_date("31/12/1999") == "1999-12-31"
assert convert_date("31/13/2020") == "Invalid date format"
assert convert_date("32/01/2020") == "Invalid date format"
assert convert_date("15-08-2021") == "Invalid date format"
# User input
date_input = input("Enter a date in DD/MM/YYYY format: ")
converted_date = convert_date(date_input)
if converted_date.startswith("Invalid"):
    print(converted_date)
```

```
PS C:\Users\saipr\OneDrive\Desktop\AI assistant>  & 'c:\Users\saipr\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\saipr\.vscode\exte
25.18.0-win32-x64\bundled\libs\debugpy\launcher' '61986' '--' 'c:\Users\saipr\OneDrive\Desktop\AI assistant\lab-8.3.py'
Enter a date in DD/MM/YYYY format: 01/02/2025
Converted date: 2025-02-01
```

Code Analysis :

- The function convert_date() converts date format from DD/MM/YYYY → YYYY-MM-DD.
- It validates day, month, and year before conversion.
- Test cases ensure correct handling of valid and invalid dates.
- User input is processed safely using exception handling.
- The program prints the converted date or an error message.