

Assignment-8.3

Name : Akshitha

Hall Ticket no : 2303A51330

Batch No : 20

Task -1:

Prompt: Email Validation using TDD

```
8.3-Assignment.py > ...
1  #Task-1 Email Validation using TDD Scenario
2  import re
3  def validate_email(email):
4      # Regular expression for validating an Email
5      regex = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
6
7      # If the email matches the regex pattern, it's valid
8      if re.match(regex, email):
9          return True
10     else:
11         return False
12     # # Test cases
13     # print(validate_email("test@example.com")) # Should return True
14     # print(validate_email("invalid.email"))    # Should return False
15     #user have to enter the input
16     user_input = input("Enter an email address to validate: ")
17     print(validate_email(user_input))
18
```

OUTPUT :

```
• (base) akshithakashireddy@Akshithas AI % python -u "/Users/akshithakashireddy/Desktop/AI /te
mpCodeRunnerFile.py"
Enter an email address to validate: akshithakashireddy@gmail.com
True
• (base) akshithakashireddy@Akshithas AI % python -u "/Users/akshithakashireddy/Desktop/AI /te
mpCodeRunnerFile.py"
Enter an email address to validate: akshitha@gmail.com
False
❖ (base) akshithakashireddy@Akshithas AI %
```

Observation:

- This program validates email addresses based on required conditions such as presence of '@' and '.' symbols.
- It ensures that the email does not start or end with special characters and prevents multiple '@' symbols.
- The function correctly distinguishes valid and invalid email formats using conditional logic and passes all test cases successfully.

Task 2:

Prompt: Grade Assignment using Loops

```
#Task 2 Grade Assignment using Loops Scenario
# You are building an automated grading system for an online examination
# Requirements
# • AI should generate test cases for assign_grade(score) where:
# 90-100 → A
# 80-89 → B
# 70-79 → C
# 60-69 → D
# Below 60 → F
def assign_grade(score):
    if 90 <= score <= 100:
        return 'A'
    elif 80 <= score < 90:
        return 'B'
    elif 70 <= score < 80:
        return 'C'
    elif 60 <= score < 70:
        return 'D'
    else:
        return 'F'
#user have to enter the input
score_input = int(input("Enter the score to assign a grade: "))
print(assign_grade(score_input))
```

Output:

```
• (base) akshithakashireddy@Akshithas AI % python -u "/Users/akshithakashireddy/Desktop/AI /tempCodeRunnerFile.py"
Enter an email address to validate: akshithakashireddy@gmail.com
True
• (base) akshithakashireddy@Akshithas AI % python -u "/Users/akshithakashireddy/Desktop/AI /tempCodeRunnerFile.py"
Enter an email address to validate: akshitha@gmail.com
False
❖ (base) akshithakashireddy@Akshithas AI %
```

Observation:

- This program assigns grades based on score ranges using conditional statements and loops for testing multiple cases.
- It correctly handles boundary values such as 60, 70, 80, and 90 and assigns appropriate grades.
- Invalid inputs like negative numbers, values above 100, or non-numeric inputs are handled properly to avoid errors.

Task 3:

Prompt : Sentence Palindrome Checker

```

#Task 3 Sentence Palindrome Checker
# Scenario
# You are developing a text-processing utility to analyze sentences.
# Requirements
# • AI should generate test cases for is_sentence_palindrome(sentence)
# • Ignore case, spaces, and punctuation
# • Test both palindromic and non-palindromic sentences
# • Example:
# - "A man a plan a canal Panama" → True
import re
def is_sentence_palindrome(sentence):
    # Remove spaces, punctuation, and convert to lowercase
    cleaned_sentence = re.sub(r'^A-Za-z0-9', '', sentence).lower()

    # Check if the cleaned sentence is equal to its reverse
    return cleaned_sentence == cleaned_sentence[::-1]
#user have to enter the input
sentence_input = input("Enter a sentence to check if it's a palindrome:")
print(is_sentence_palindrome(sentence_input))

```

Output :

```

(base) akshithakashireddy@Akshithas AI % python -u "/Users/akshithakashireddy/Desktop/AI /te
mpCodeRunnerFile.py"
Enter the score to assign a grade: 94
A
(base) akshithakashireddy@Akshithas AI % python -u "/Users/akshithakashireddy/Desktop/AI /te
mpCodeRunnerFile.py"
Enter the score to assign a grade: 80
B
(base) akshithakashireddy@Akshithas AI % 

```

Observation:

- This program checks whether a sentence is a palindrome by ignoring case, spaces, and punctuation marks.
- It preprocesses the input string to compare only meaningful characters.
- The function accurately returns True for palindromes and False for non-palindromes using clear string manipulation logic.

Task 4 :

Prompt : Shopping Cart Class

```

71 # Task 4: ShoppingCart Class Scenario
72 # You are designing a basic shopping cart module for an e-commerce application.
73 # Requirements
74 # • AI should generate test cases for the ShoppingCart class
75 # • Class must include the following methods:
76 # - add_item(name, price)
77 # - remove_item(name)
78 # - total_cost()
79 # • Validate correct addition, removal, and cost calculation
80 # • Handle empty cart scenarios
81 class ShoppingCart:
82     def __init__(self):
83         self.items = {}
84
85     def add_item(self, name, price):
86         self.items[name] = price
87
88     def remove_item(self, name):
89         if name in self.items:
90             del self.items[name]
91
92     def total_cost(self):
93         return sum(self.items.values())
94
95 #user have to enter the input
96 cart = ShoppingCart()
97 while True:
98     action = input("Enter action (add/remove/total/exit): ").lower()
99     if action == 'add':
100         name = input("Enter item name: ")
101         price = float(input("Enter item price: "))
102         cart.add_item(name, price)
103     elif action == 'remove':
104         name = input("Enter item name to remove: ")
105         cart.remove_item(name)
106     elif action == 'total':
107         print(f"Total cost: {cart.total_cost()}")
108     elif action == 'exit':
109         break
110     else:
111         print("Invalid action. Please try again.")

```

Output :

```

(base) akshithakashireddy@Akshithas AI % python -u "/Users/akshithakashireddy/Desktop/AI /te
mpCodeRunnerFile.py"
Enter action (add/remove/total/exit): add
Enter item name: pen
Enter item price: 30
Enter action (add/remove/total/exit): add
Enter item name: book
Enter item price: 50
Enter action (add/remove/total/exit): total
Total cost: 80.0
Enter action (add/remove/total/exit): exit
(base) akshithakashireddy@Akshithas AI %

```

Observation:

- This program implements a ShoppingCart class using objectoriented programming concepts.
- It allows adding, removing items, and calculating the total cost using defined class methods.
- The system correctly handles empty cart conditions and ensures accurate cost calculation.

Task 5 :

Prompt : Date Format Conversion

```
# Task 5: Date Format Conversion Scenario
# You are creating a utility function to convert date formats for reports.
# Requirements
# • AI should generate test cases for convert_date_format(date_str)
# • Input format must be "YYYY-MM-DD"
# • Output format must be "DD-MM-YYYY"
# • Example:
# - "2023-10-15" → "15-10-2023"
def convert_date_format(date_str):
    try:
        year, month, day = date_str.split('-')
        return f"{day}-{month}-{year}"
    except ValueError:
        return "Invalid date format. Please use YYYY-MM-DD."
#user have to enter the input
date_input = input("Enter a date in YYYY-MM-DD format to convert: ")
print(convert_date_format(date_input))
```

Output :

```
(base) akshithakashireddy@Akshithas AI % python -u "/Users/akshithakashireddy/Desktop/AI /te
mpCodeRunnerFile.py"
Enter action (add/remove/total/exit): add
Enter item name: pen
Enter item price: 30
Enter action (add/remove/total/exit): add
Enter item name: book
Enter item price: 50
Enter action (add/remove/total/exit): total
Total cost: 80.0
Enter action (add/remove/total/exit): exit
(base) akshithakashireddy@Akshithas AI %
```

Observation:

- This program converts date format from "YYYY-MM-DD" to "DD-MM-YYYY" using string manipulation techniques.
- It validates the input format before performing the conversion to ensure correctness.
- The function successfully produces accurate formatted dates and passes all test cases.