

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:

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files in the 'AI ASSISTED' folder, including 'assignment-1.py', 'Assignment-3.3.py', 'lab-1', 'LAB-1.4.py', 'Lab-3.4.py', 'Lab-4.3.py', 'Lab-5.4.py', 'lab-6.3.py', 'Lab-6.4.py', 'Lab-7.3.py', 'LAB-8.3.py', and 'project.py'. The 'LAB-8.3.py' file is open in the main editor area. The code defines a class 'UserRegistration' with methods for initializing user details and validating email addresses using a regular expression. It also includes a 'display' method to print the user's name and email. Test cases at the bottom show valid and invalid email addresses being processed. The terminal below shows the execution of the program and its output, which matches the expected results.

```

File Edit Selection View Go Run Terminal Help ← → Q AI Assisted
EXPLORER AI ASSISTED assignment-1.py U Assignment-3.3.py U lab-1 LAB-1.4.py U Lab-3.4.py U Lab-4.3.py U Lab-5.4.py U lab-6.3.py U Lab-6.4.py U Lab-7.3.py U LAB-8.3.py U project.py U
Welcome Lab-7.3.py U project.py U LAB-8.3.py U Lab-3.4.py U
1 #TASK-1:
2 #write a python program to develop a user registration system that requires reliable email input validation
3 import re
4 class UserRegistration:
5     def __init__(self, name, email):
6         self.name = name
7         self.email = email
8     def validate_email(self):
9         pattern = r'^[a-zA-Z0-9_.%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
10        if re.match(pattern, self.email):
11            return True
12        else:
13            return False
14    def display(self):
15        print(f"Name: {self.name}")
16        print(f"Email: {self.email}")
17        if self.validate_email():
18            print("Email is valid.")
19        else:
20            print("Email is invalid.")
21    # Test Cases
22 user1 = UserRegistration("srinithya", "srinithya.polineni@gmail.com")
23 user2 = UserRegistration("Bob", "bob@example")
24 user1.display()
25 user2.display()
26
TERMINAL OUTPUT DEBUG CONSOLE PORTS GITLENS Python Debug Console + ×
hon.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher 61409 ... c:\users\sri...|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\sri...|LAB-8.3.py
In 18, Col 37 Spaces: 4 UFT-8 CR/LF {} Python 3.13.7 ⓘ Go Live

```

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

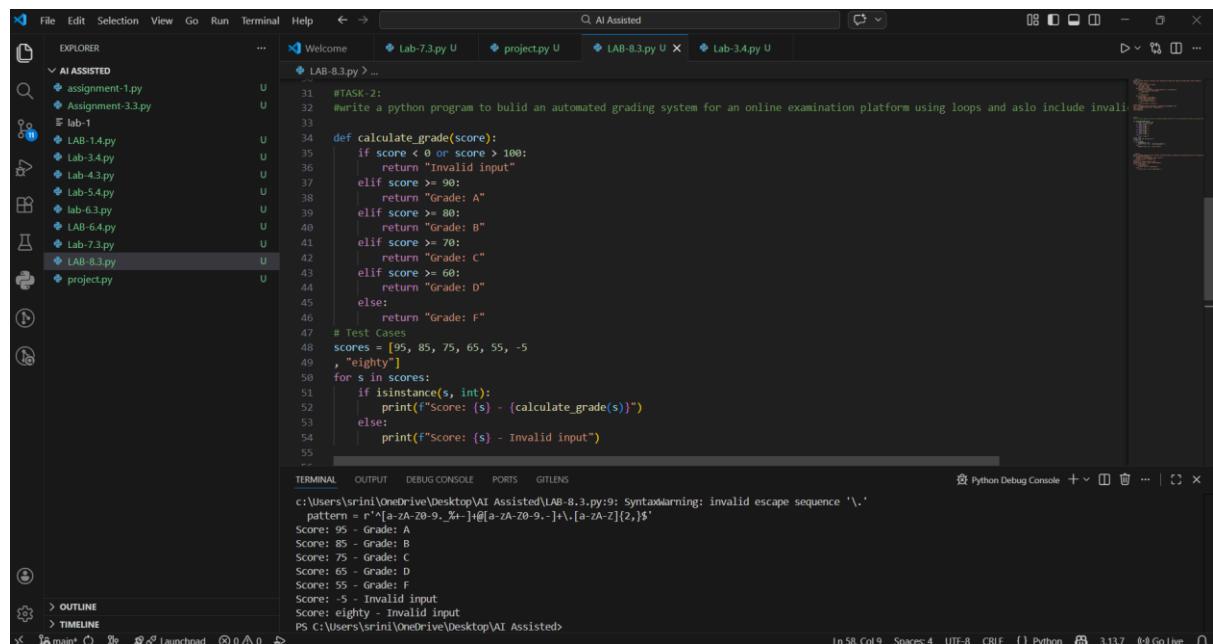
```

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:



The screenshot shows a code editor interface with several tabs open. The main tab contains Python code for calculating grades based on scores and running test cases. The code includes a function to calculate grades and a loop to print scores and their corresponding grades or invalid input messages. The code editor has a dark theme with syntax highlighting for Python keywords and comments.

```

File Edit Selection View Go Run Terminal Help ← → Q AI Assisted
EXPLORER ... Welcome Lab-7.3.py U project.py U LAB-8.3.py U Lab-3.4.py
AI ASSISTED assignment-1.py U Assignment-3.3.py U lab-1 LAB-1.4.py U Lab-3.4.py U Lab-4.3.py U Lab-5.4.py U Lab-6.4.py U Lab-7.3.py LAB-8.3.py U project.py U
TERMINAL OUTPUT DEBUG CONSOLE PORTS GITLENS
c:\Users\srujan\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\srujan\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:

```

LAB-8.3.py > ...
59
60     #TASK-3:
61     #write a python program to check sentence palindrome checker to develop a text-processing utility to analyze sentences
62     def is_palindrome(sentence):
63         cleaned = ''.join(sentence.lower().split())
64         return cleaned == cleaned[::-1]
65     # Test Cases
66     sentences = ["Madam In Eden Im Adam",
67                  "Hello World", "A man a plan a canal Panama"]
68     for s in sentences:
69         if is_palindrome(s):
70             print(f'"{s}" is a palindrome.')
71         else:
72             print(f'"{s}" is not a palindrome.')

```

TERMINAL OUTPUT DEBUG CONSOLE PORTS GITLENS

```

PS C:\Users\srini\OneDrive\Desktop\AI Assisted> & 'c:\Users\srini\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\srini\vscode\extensions\ms-vscode.cpptools\2025.18.0-win32-x64\bundled\libs\debug\launcher' '51507' '-l' '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>

```

Ln 54, Col 48 Spaces: 4 UTF-8 CRLF {} Python 3.13.7 ⓘ Go Live

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:

```

File Edit Selection View Go Run Terminal Help < > Q AI Assisted
EXPLORER ... Welcome Lab-7.3.py U project.py U LAB-8.3.py U Lab-3.4.py U
AI ASSISTED assignment-1.py U
Assignment-3.3.py U
lab-1
Lab-1.4.py U
Lab-3.4.py U
Lab-4.3.py U
Lab-5.4.py U
lab-6.3.py U
Lab-6.4.py U
Lab-7.3.py U
LAB-8.3.py U project.py U
74
75 #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
76 class ShoppingCart:
77     def __init__(self):
78         self.cart = {}
79     def add_item(self, item, price):
80         self.cart[item] = price
81         print(f"Added {item} to cart at ₹{price}")
82     def remove_item(self, item):
83         if item in self.cart:
84             del self.cart[item]
85             print(f"Removed {item} from cart")
86         else:
87             print(f"{item} not found in cart")
88     def total_cost(self):
89         return sum(self.cart.values())
90
# Test Cases
91 cart = ShoppingCart()
92 cart.add_item("Laptop", 50000)
93 cart.add_item("Headphones", 2000)
94 cart.add_item("Mouse", 500)
95 print("Total Cost: ₹", cart.total_cost())
96 cart.remove_item("Headphones")
97 print("Total Cost after removal: ₹", cart.total_cost())
98

```

TERMINAL OUTPUT DEBUG CONSOLE PORTS GITLENS Python Debug Console + ... | x

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: ₹ 50000
 PS C:\Users\sriniv\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:

The screenshot shows the Visual Studio Code interface with the 'AI Assisted' extension active. The Explorer sidebar on the left lists files like 'Assignment-1.py', 'Assignment-3.3.py', 'lab-1', 'LAB-1.4.py', 'Lab-3.4.py', 'Lab-4.3.py', 'Lab-5.4.py', 'lab-6.3.py', 'LAB-6.4.py', 'LAB-7.3.py', and 'LAB-8.3.py'. The 'projectpy' folder is also visible. The main editor area contains a Python script named 'LAB-8.3.py' with code for date conversion. The terminal at the bottom shows the output of running the script with various date strings. The status bar at the bottom right indicates the file is 3.13.7 and shows 'Go Live'.

```
102 #TASK-5:write a python program to create a utility function to convert date formats for reports
103 from datetime import datetime
104 def convert_date_format(date_str, current_format, desired_format):
105     try:
106         date_obj = datetime.strptime(date_str, current_format)
107         return date_obj.strftime(desired_format)
108     except ValueError:
109         return "Invalid date format"
110
111 # Test Cases
112 dates = [
113     ("2024-06-15", "%Y-%m-%d", "%d/%m/%Y"),
114     ("15/06/2024", "%d/%m/%Y", "%Y-%m-%d"),
115     ("06-15-2024", "%m-%d-%Y", "%Y/%m/%d"),
116     ("invalid-date", "%Y-%m-%d", "%d/%m/%Y")]
117
for date_str, current_fmt, desired_fmt in dates:
    print(f"Original: {date_str} - Converted: {convert_date_format(date_str, current_fmt, desired_fmt)}")
```

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.