

Assignment - 8

Name : K Mitesh Reddy

Hall Ticket No :

2303A51814

Task 1: Email Validation

PROMPT :

Implement a function to validate email addresses based on the following criteria:

The email must contain exactly one "@" symbol. # -

The email must not start or end with "@" or ".".

The email must contain at least one "." after the "@" symbol.

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, ...
- Title Bar:** Q_ 1814 (Workspace)
- Toolbar:** Includes icons for file operations (New, Open, Save, etc.), search, and navigation.
- Code Area:** The main workspace displays the following Python code:

```
C:\Users\Users> Mites> OneDrive> Desktop> {} 1814.code-workspace
1 import unittest
2 import re
3 import string
4 # Task 1: Email Validation
5 # PROMPT :
6 # Implement a function to validate email addresses based on the following criteria:
7 # - The email must contain exactly one "@" symbol.
8 # - The email must not start or end with "@" or ..
9 # - The email must contain at least one "." after the "@" symbol.
10 def is_valid_email(email):
11     if not isinstance(email, str):
12         return False
13     if email.count("@") != 1:
14         return False
15     if email[0] in "@._" or email[-1] in "@._":
16         return False
17     if "." not in email:
18         return False
19     pattern = r"^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$"
20     return re.match(pattern, email) is not None
21 class TestEmailValidation(unittest.TestCase):
22     def test_valid_emails(self):
23         self.assertTrue(is_valid_email("test@example.com"))
24         self.assertTrue(is_valid_email("user.name@domain.co"))
25         self.assertTrue(is_valid_email("abc123@gmail.in"))
26     def test_invalid_emails(self):
27         self.assertFalse(is_valid_email("testexample.com"))
28         self.assertFalse(is_valid_email("test@example.com"))
29         self.assertFalse(is_valid_email("@example.com"))
30         self.assertFalse(is_valid_email("test@.com"))
31         self.assertFalse(is_valid_email("test@com"))
32         self.assertFalse(is_valid_email("test@domain."))

```

Right Panel: Shows a sidebar with "10K" notifications, a file tree, and a preview area for the current file.

Bottom Right: Open Workspace button.

Task 2: Grade Assignment

PROMPT :

Create a function that takes a student's score (0-100) and returns the corresponding grade based on the following scale:

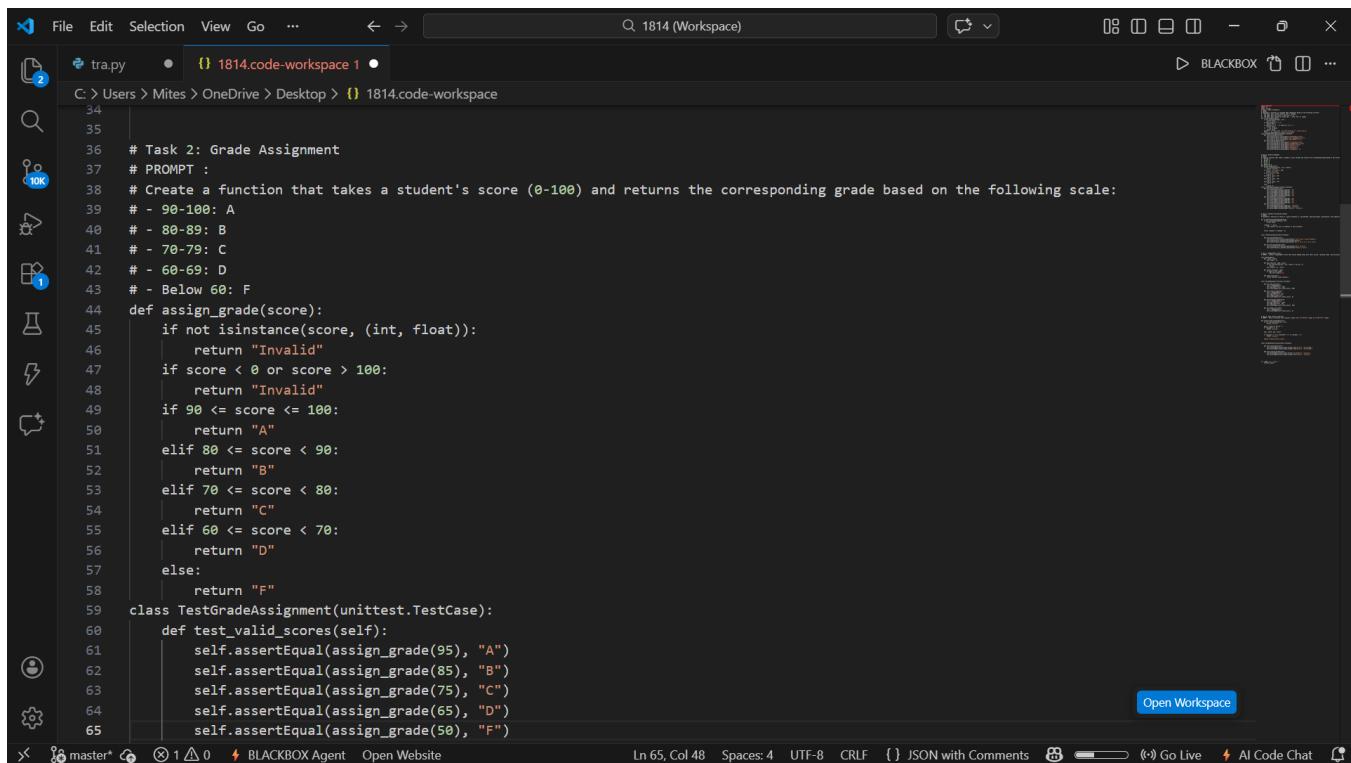
- 90-100: A

- 80-89: B

- 70-79: C

- 60-69: D

- Below 60: F



The screenshot shows a code editor interface with the following details:

- File Path:** C:\Users\ Mites\OneDrive\Desktop\1814.code-workspace
- File Name:** trappy.py
- Code Content:**

```
34
35
36 # Task 2: Grade Assignment
37 # PROMPT :
38 # Create a function that takes a student's score (0-100) and returns the corresponding grade based on the following scale:
39 # - 90-100: A
40 # - 80-89: B
41 # - 70-79: C
42 # - 60-69: D
43 # - Below 60: F
44 def assign_grade(score):
45     if not isinstance(score, (int, float)):
46         return "Invalid"
47     if score < 0 or score > 100:
48         return "Invalid"
49     if 90 <= score <= 100:
50         return "A"
51     elif 80 <= score < 90:
52         return "B"
53     elif 70 <= score < 80:
54         return "C"
55     elif 60 <= score < 70:
56         return "D"
57     else:
58         return "F"
59 class TestGradeAssignment(unittest.TestCase):
60     def test_valid_scores(self):
61         self.assertEqual(assign_grade(95), "A")
62         self.assertEqual(assign_grade(85), "B")
63         self.assertEqual(assign_grade(75), "C")
64         self.assertEqual(assign_grade(65), "D")
65         self.assertEqual(assign_grade(55), "F")
```
- Editor Features:** The interface includes standard file operations (File, Edit, Selection, View, Go), a search bar, and a sidebar with various icons (document, search, refresh, etc.).
- Bottom Status Bar:** Shows the current file path (1814.code-workspace), line count (Ln 65), column count (Col 48), and encoding (UTF-8). It also includes buttons for JSON with Comments, Go Live, and AI Code Chat.

The screenshot shows a code editor interface with a dark theme. The main area displays Python code for a grade assignment function and its unit tests. The code is as follows:

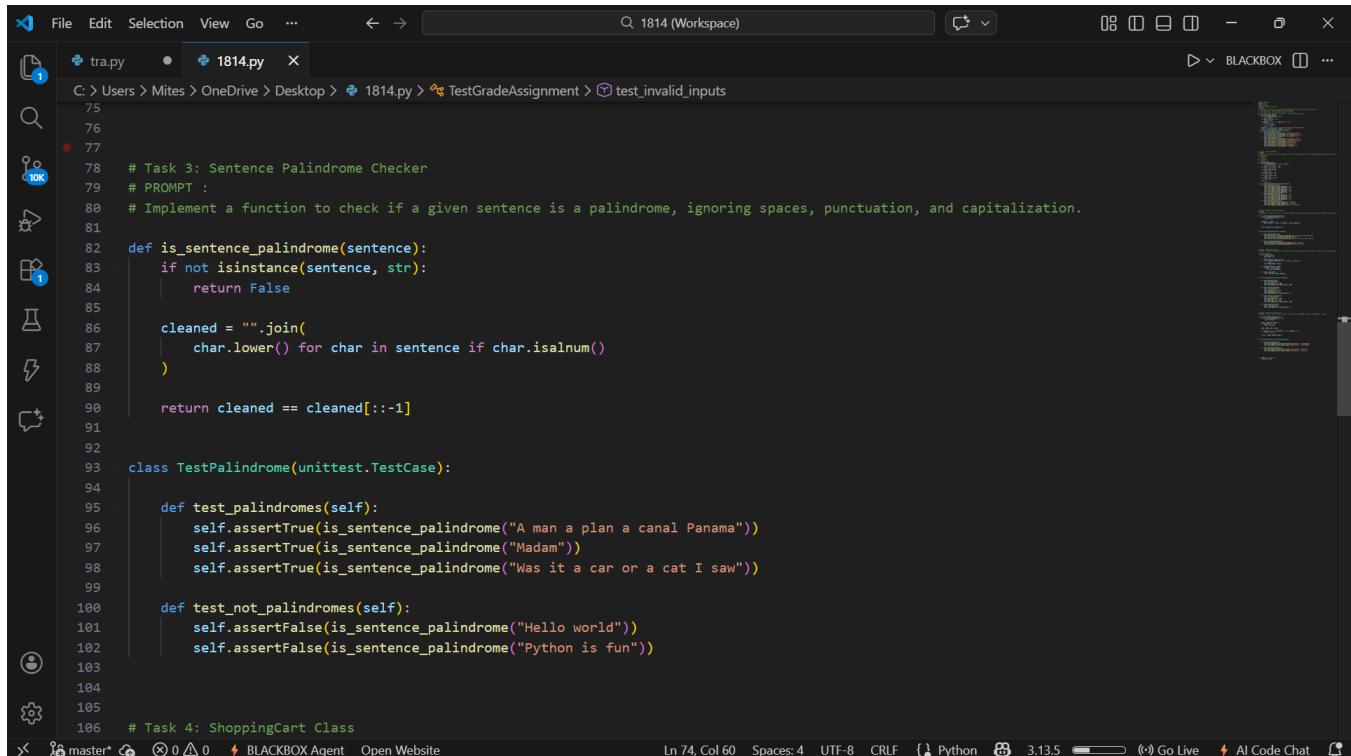
```
44 def assign_grade(score):
45     if score < 0 or score > 100:
46         return "Invalid"
47     if 90 <= score <= 100:
48         return "A"
49     elif 80 <= score < 90:
50         return "B"
51     elif 70 <= score < 80:
52         return "C"
53     elif 60 <= score < 70:
54         return "D"
55     else:
56         return "F"
57
58 class TestGradeAssignment(unittest.TestCase):
59     def test_valid_scores(self):
60         self.assertEqual(assign_grade(95), "A")
61         self.assertEqual(assign_grade(85), "B")
62         self.assertEqual(assign_grade(75), "C")
63         self.assertEqual(assign_grade(65), "D")
64         self.assertEqual(assign_grade(50), "F")
65
66     def test_boundary_values(self):
67         self.assertEqual(assign_grade(90), "A")
68         self.assertEqual(assign_grade(80), "B")
69         self.assertEqual(assign_grade(70), "C")
70         self.assertEqual(assign_grade(60), "D")
71
72     def test_invalid_inputs(self):
73         self.assertEqual(assign_grade(-5), "Invalid")
74         self.assertEqual(assign_grade(105), "Invalid")
75         self.assertEqual(assign_grade("eighty"), "Invalid")
```

The code editor includes various status icons on the left margin, such as a file icon with a 10x mark, a search icon, and a refresh icon. The bottom status bar shows the file path as C:/Users/Mites/OneDrive/Desktop/1814.py, the line number as Ln 74, column 60, and other details like Python version 3.13.5 and AI Code Chat.

Task 3: Sentence Palindrome Checker

PROMPT :

Implement a function to check if a given sentence is a palindrome, ignoring spaces, punctuation, and capitalization.



The screenshot shows a code editor window with the following details:

- File:** 1814.py
- Content:**

```
75
76
77
78 # Task 3: Sentence Palindrome Checker
79 # PROMPT :
80 # Implement a function to check if a given sentence is a palindrome, ignoring spaces, punctuation, and capitalization.
81
82 def is_sentence_palindrome(sentence):
83     if not isinstance(sentence, str):
84         return False
85
86     cleaned = "".join(
87         char.lower() for char in sentence if char.isalnum()
88     )
89
90     return cleaned == cleaned[::-1]
91
92
93 class TestPalindrome(unittest.TestCase):
94
95     def test_palindromes(self):
96         self.assertTrue(is_sentence_palindrome("A man a plan a canal Panama"))
97         self.assertTrue(is_sentence_palindrome("Madam"))
98         self.assertTrue(is_sentence_palindrome("Was it a car or a cat I saw"))
99
100    def test_not_palindromes(self):
101        self.assertFalse(is_sentence_palindrome("Hello world"))
102        self.assertFalse(is_sentence_palindrome("Python is fun"))
103
104
105
106 # Task 4: ShoppingCart Class
```
- Toolbars and Status Bar:** The status bar at the bottom shows "Ln 74, Col 60" and "Spaces: 4". Other icons include "BLACKBOX Agent", "Open Website", "Python 3.13.5", "Go Live", and "AI Code Chat".

Task 4: ShoppingCart Class

PROMPT : Create a ShoppingCart class that allows adding items with their prices, removing items, and calculating the total cost.

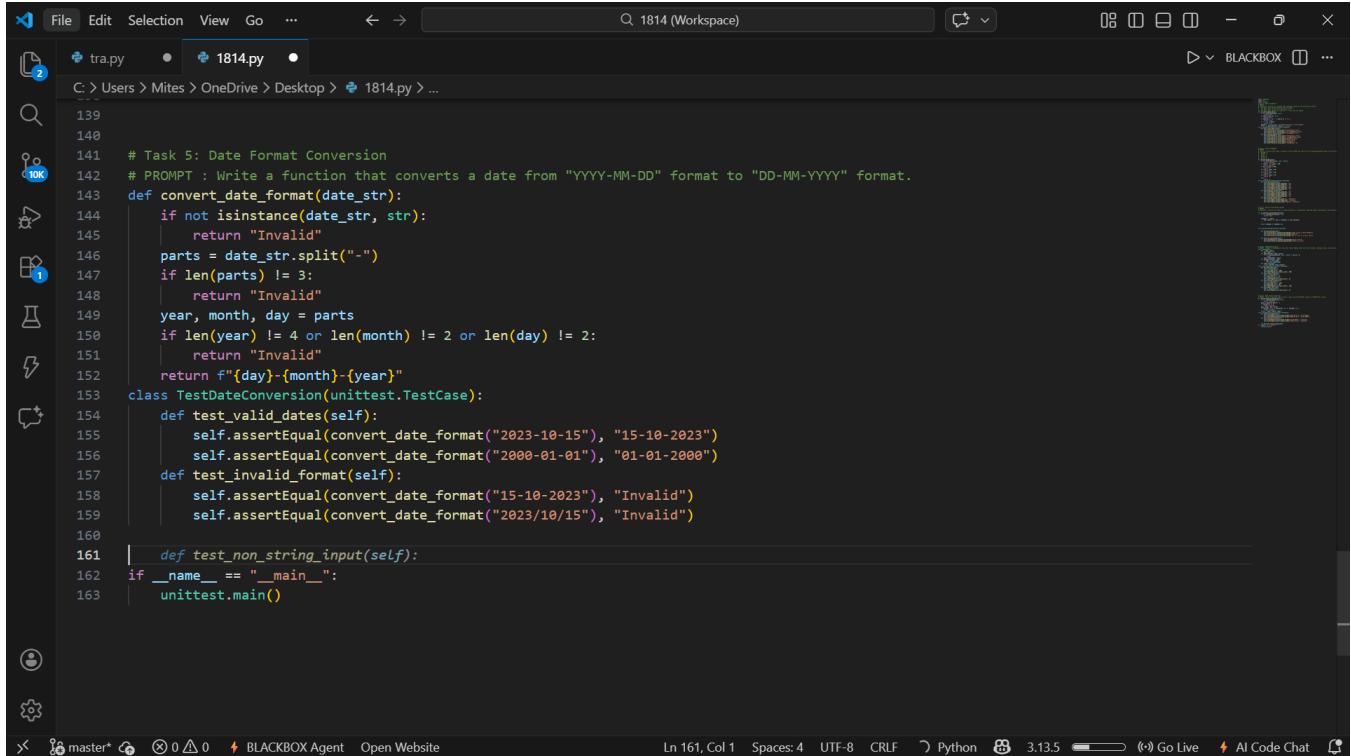
The screenshot shows a code editor interface with a dark theme. The top bar includes standard menu items like File, Edit, Selection, View, Go, and a search bar labeled 'Q 1814 (Workspace)'. On the right side, there are various icons for file operations and a 'BLACKBOX' dropdown. The main area displays Python code for a ShoppingCart class and its unit tests.

```
106 # Task 4: ShoppingCart Class
107 # PROMPT : Create a ShoppingCart class that allows adding items with their prices, removing items, and calculating the total cost.
108 class ShoppingCart:
109     def __init__(self):
110         self.items = {}
111     def add_item(self, name, price):
112         if not isinstance(price, (int, float)) or price < 0:
113             return
114         self.items[name] = price
115     def remove_item(self, name):
116         if name in self.items:
117             del self.items[name]
118     def total_cost(self):
119         return sum(self.items.values())
120 class TestShoppingCart(unittest.TestCase):
121     def test_add_item(self):
122         cart = ShoppingCart()
123         cart.add_item("Book", 500)
124         self.assertEqual(cart.total_cost(), 500)
125     def test_remove_item(self):
126         cart = ShoppingCart()
127         cart.add_item("Pen", 20)
128         cart.remove_item("Pen")
129         self.assertEqual(cart.total_cost(), 0)
130     def test_multiple_items(self):
131         cart = ShoppingCart()
132         cart.add_item("Book", 500)
133         cart.add_item("Pen", 20)
134         self.assertEqual(cart.total_cost(), 520)
135     def test_empty_cart(self):
136         cart = ShoppingCart()
137         self.assertEqual(cart.total_cost(), 0)
```

At the bottom of the editor, status information includes 'Ln 134, Col 49', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', '3.13.5', 'Go Live', 'AI Code Chat', and a refresh icon.

Task 5: Date Format Conversion

PROMPT : Write a function that converts a date from "YYYY-MM-DD" format to "DD-MM-YYYY" format.



The screenshot shows a code editor interface with a dark theme. The left sidebar has icons for file operations like Open, Save, Find, and Refresh. The top bar includes File, Edit, Selection, View, Go, and a tab for '1814 (Workspace)'. The main area displays Python code:

```
C: > Users > Mites > OneDrive > Desktop > 1814.py > ...
139
140
141 # Task 5: Date Format Conversion
142 # PROMPT : Write a function that converts a date from "YYYY-MM-DD" format to "DD-MM-YYYY" format.
143 def convert_date_format(date_str):
144     if not isinstance(date_str, str):
145         return "Invalid"
146     parts = date_str.split("-")
147     if len(parts) != 3:
148         return "Invalid"
149     year, month, day = parts
150     if len(year) != 4 or len(month) != 2 or len(day) != 2:
151         return "Invalid"
152     return f"{day}-{month}-{year}"
153 class TestDateConversion(unittest.TestCase):
154     def test_valid_dates(self):
155         self.assertEqual(convert_date_format("2023-10-15"), "15-10-2023")
156         self.assertEqual(convert_date_format("2000-01-01"), "01-01-2000")
157     def test_invalid_format(self):
158         self.assertEqual(convert_date_format("15-10-2023"), "Invalid")
159         self.assertEqual(convert_date_format("2023/10/15"), "Invalid")
160
161     def test_non_string_input(self):
162 if __name__ == "__main__":
163     unittest.main()
```

The status bar at the bottom shows: Ln 161, Col 1 | Spaces: 4 | UTF-8 | CRLF | Python | 3.13.5 | Go Live | AI Code Chat.

