

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.

FileEditSelectionViewGo...1814 (Workspace)

trapy1814.code-workspace 1BLACKBOX

C:\Users\Mites> OneDrive > Desktop > 1814.code-workspace

1import unittest
2import re
3import 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.
10def 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
21class 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."))

Open Workspace

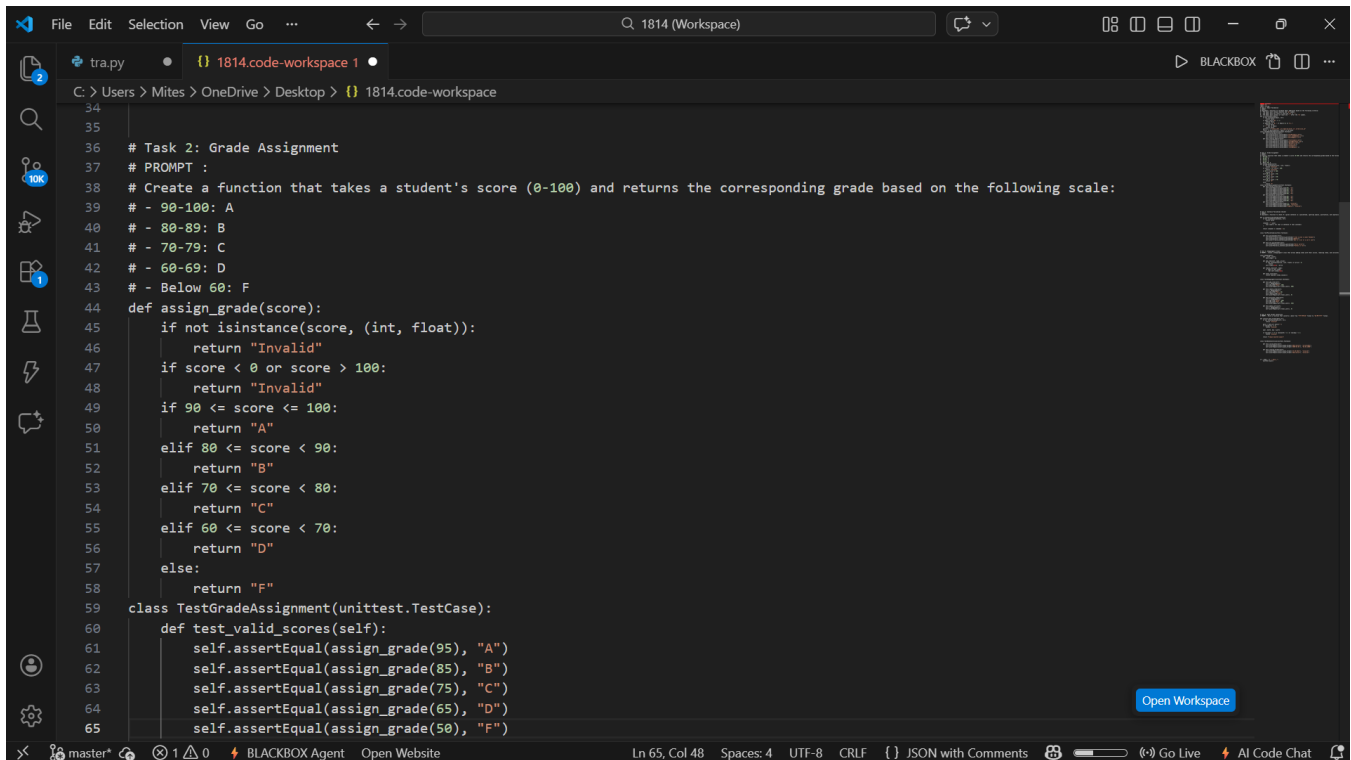
master*0BLACKBOX AgentOpen WebsiteLn 9, Col 66Spaces: 4UTF-8CRLFJSON with CommentsGo LiveAI Code Chat

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 Visual Studio Code editor window with a workspace named '1814 (Workspace)'. The editor is open to a file named 'tra.py'. The code in the file implements a function 'assign_grade(score)' that takes a student's score and returns a grade based on the specified scale. The function includes input validation for non-integer/float values and scores outside the 0-100 range. A test class 'TestGradeAssignment' is also present, with a 'test_valid_scores' method that uses 'unittest.TestCase' to verify the function's output for various scores.

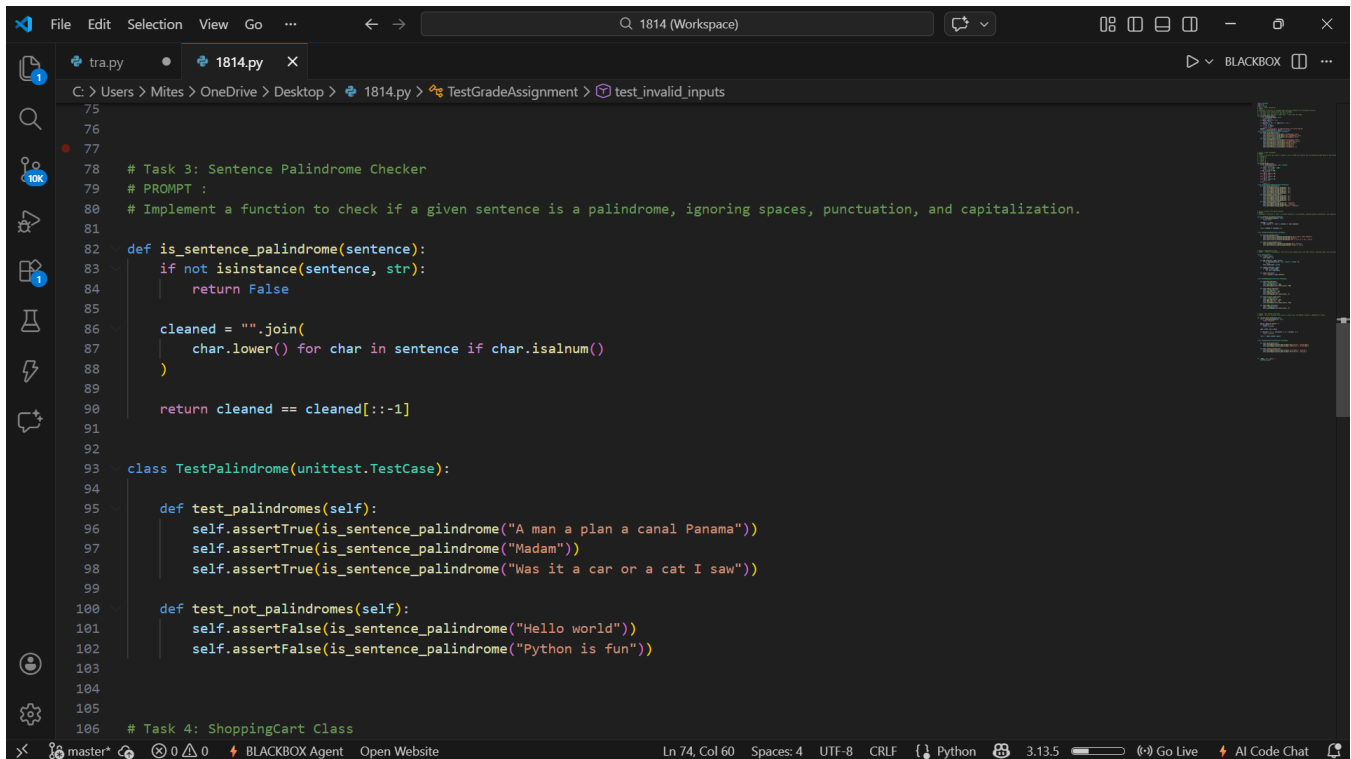
```
34
35
36 # Task 2: Grade Assignment
37 # PROMPT :
38 # Create a function that takes a student's score (0-100) and returns the corresponding grade based on the following scale:
39 # - 90-100: A
40 # - 80-89: B
41 # - 70-79: C
42 # - 60-69: D
43 # - Below 60: F
44 def assign_grade(score):
45     if 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(50), "F")
```

```
File Edit Selection View Go ... 1814 (Workspace) BLACKBOX ...
1814.py
C:\Users\Mites>OneDrive\Desktop>1814.py>TestGradeAssignment>test_invalid_inputs
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     def test_boundary_values(self):
66         self.assertEqual(assign_grade(90), "A")
67         self.assertEqual(assign_grade(80), "B")
68         self.assertEqual(assign_grade(70), "C")
69         self.assertEqual(assign_grade(60), "D")
70     def test_invalid_inputs(self):
71         self.assertEqual(assign_grade(-5), "Invalid")
72         self.assertEqual(assign_grade(105), "Invalid")
73         self.assertEqual(assign_grade("eighty"), "Invalid")
74
75
76
77
Ln 74, Col 60 Spaces: 4 UTF-8 CRLF Python 3.13.5 Go Live 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.



```
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
```

Task 4: ShoppingCart Class

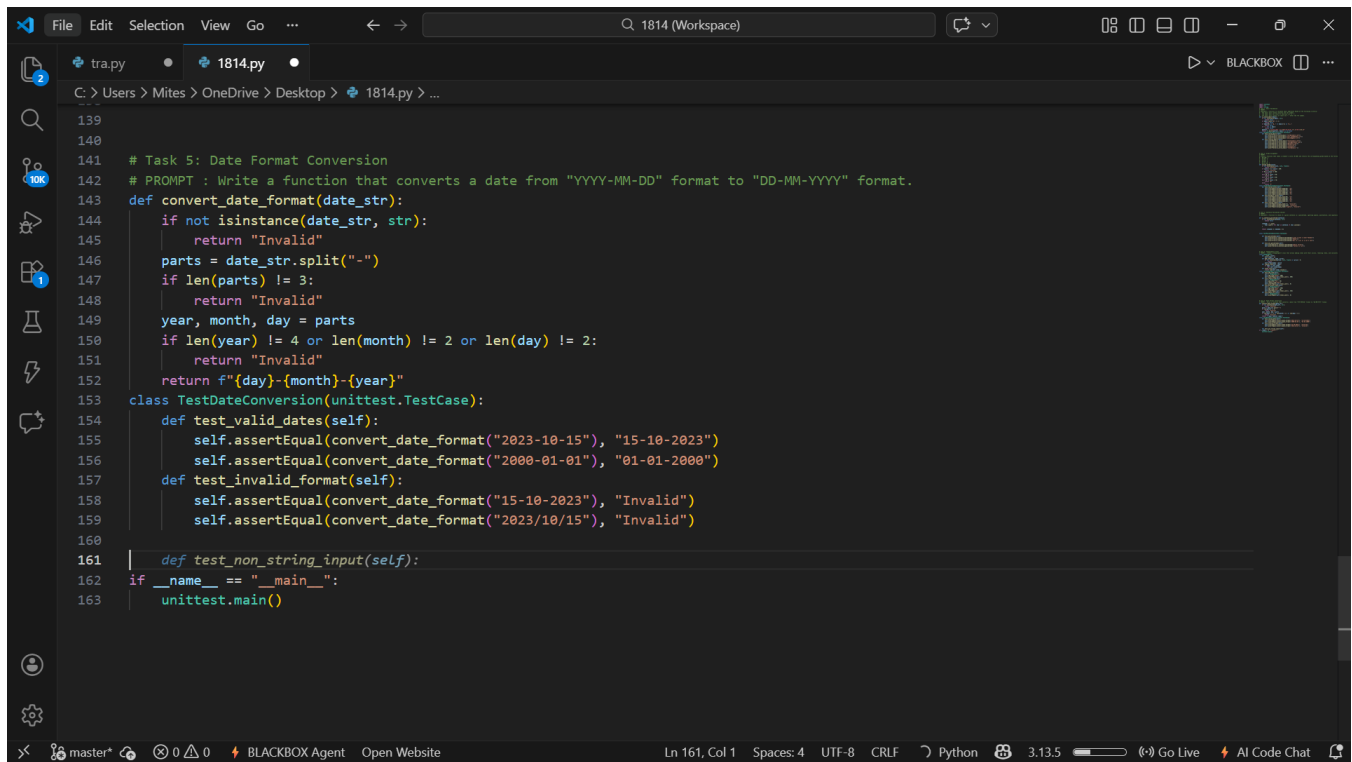
PROMPT : Create a ShoppingCart class that allows adding items with their prices, removing items, and calculating the total cost.

```
File Edit Selection View Go ... 1814 (Workspace) BLACKBOX ...
tra.py 1814.py
C:\Users\Mites>OneDrive\Desktop>1814.py>TestShoppingCart>test_multiple_items
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)
```

Ln 134, Col 49 Spaces: 4 UTF-8 CRLF Python 3.13.5 Go Live AI Code Chat

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 with a dark theme. The file explorer on the left shows a project named '1814' with a file '1814.py'. The editor displays the following Python code:

```
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
154 class TestDateConversion(unittest.TestCase):
155     def test_valid_dates(self):
156         self.assertEqual(convert_date_format("2023-10-15"), "15-10-2023")
157         self.assertEqual(convert_date_format("2000-01-01"), "01-01-2000")
158     def test_invalid_format(self):
159         self.assertEqual(convert_date_format("15-10-2023"), "Invalid")
160         self.assertEqual(convert_date_format("2023/10/15"), "Invalid")
161
162     def test_non_string_input(self):
163
164 if __name__ == "__main__":
165     unittest.main()
```

The status bar at the bottom shows the following information: master*, 0 0, BLACKBOX Agent, Open Website, Ln 161, Col 1, Spaces: 4, UTF-8, CRLF, Python, 3.13.5, Go Live, AI Code Chat.

