Assignment – 8

Name : K Mitesh Reddy

Hall Ticket No :

2303A51814

Task 1: Email Validation

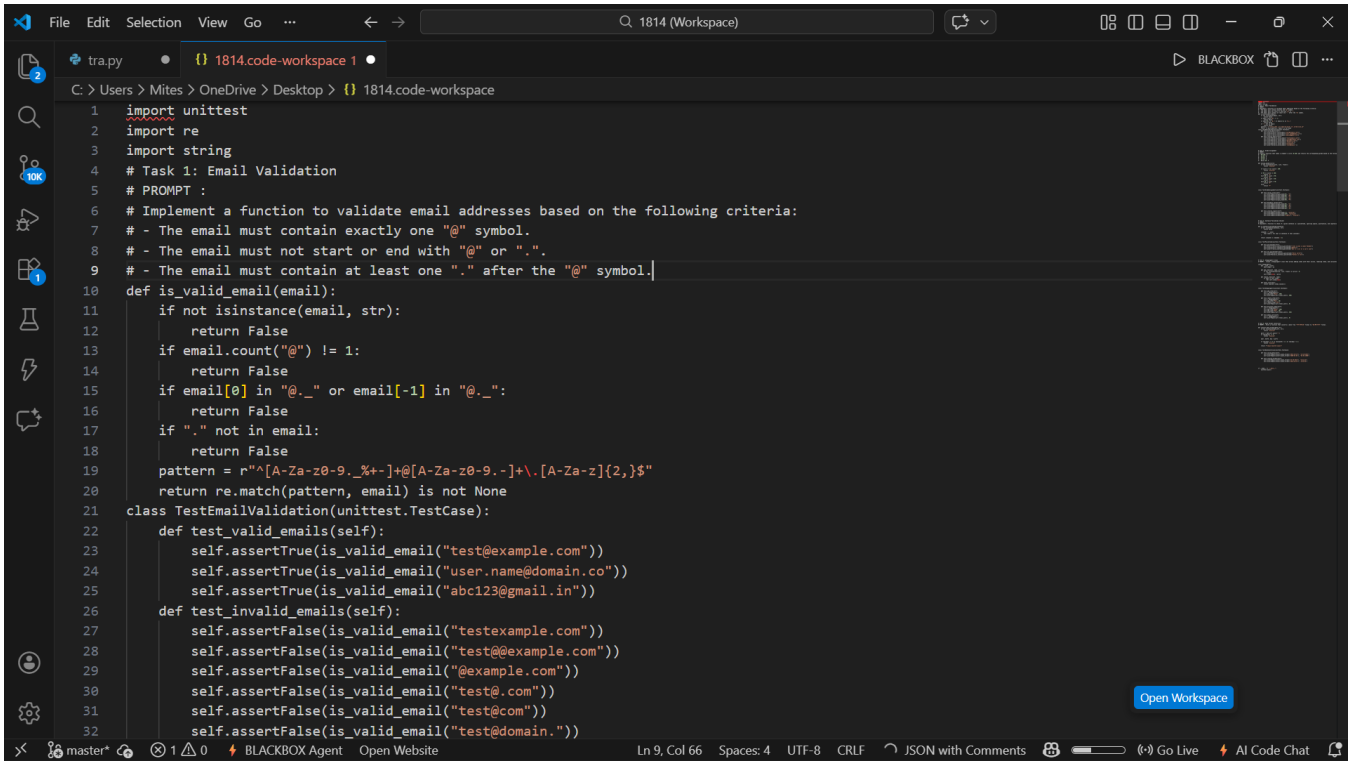
PROMPT :

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.


File Edit Selection View Go
1814 (Workspace)
tra.py
1814.code-workspace 1
BLACKBOX
C: > Users > Mites > OneDrive > Desktop > {} 1814.code-workspace
import unittest
import re
import string
# 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.
def is_valid_email(email):
if not isinstance(email, str):
return False
if email.count("@") != 1:
return False
if email[0] in "@._" or email[-1] in "@._":
return False
if "." not in email:
return False
pattern = r"^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$"
return re.match(pattern, email) is not None
class TestEmailValidation(unittest.TestCase):
def test_valid_emails(self):
self.assertTrue(is_valid_email("test@example.com"))
self.assertTrue(is_valid_email("user.name@domain.co"))
self.assertTrue(is_valid_email("abc123@gmail.in"))
def test_invalid_emails(self):
self.assertFalse(is_valid_email("testexample.com"))
self.assertFalse(is_valid_email("test@@example.com"))
self.assertFalse(is_valid_email("@example.com"))
self.assertFalse(is_valid_email("test@.com"))
self.assertFalse(is_valid_email("test@com"))
self.assertFalse(is_valid_email("test@domain."))
Open Workspace
master*
1 0
BLACKBOX Agent
Open Website
Ln 9, Col 66
Spaces: 4
UTF-8
CRLF
JSON with Comments
Go Live
AI 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:

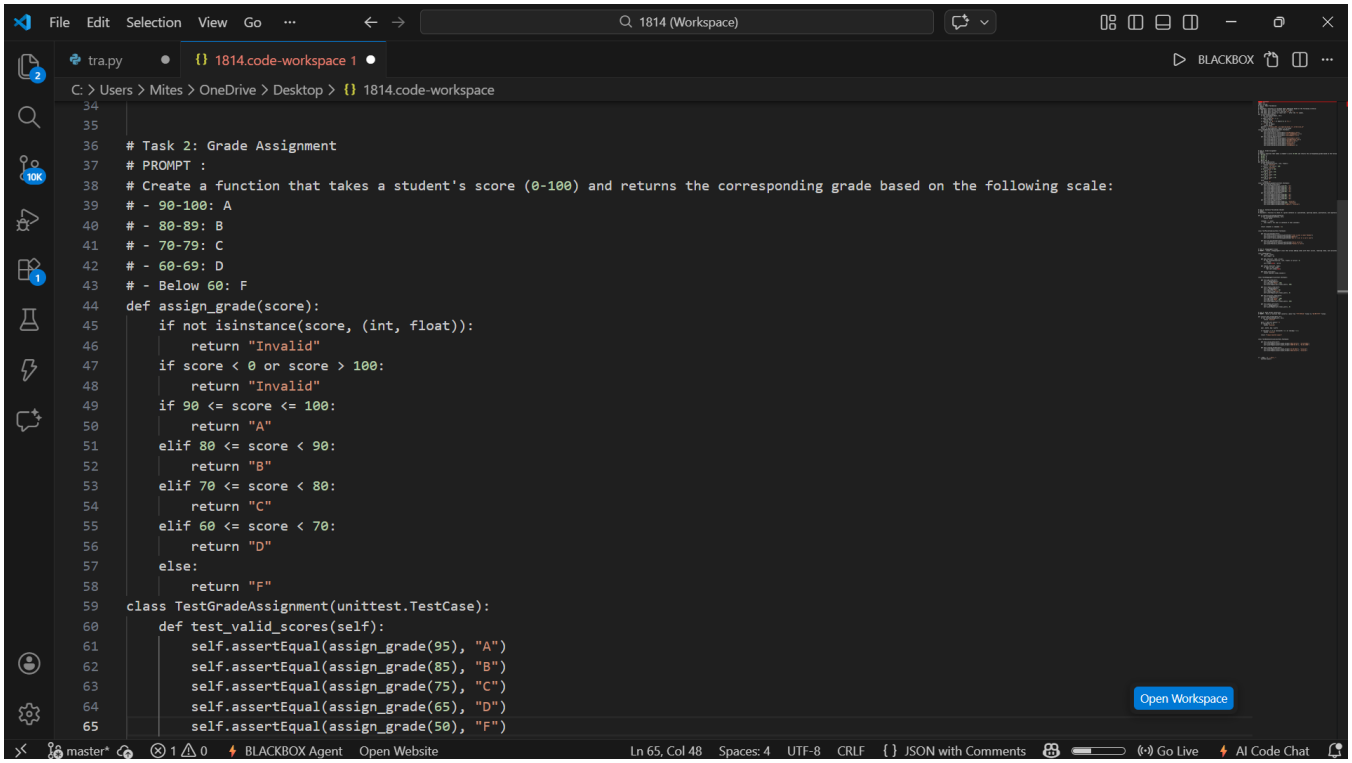
- 90-100: A

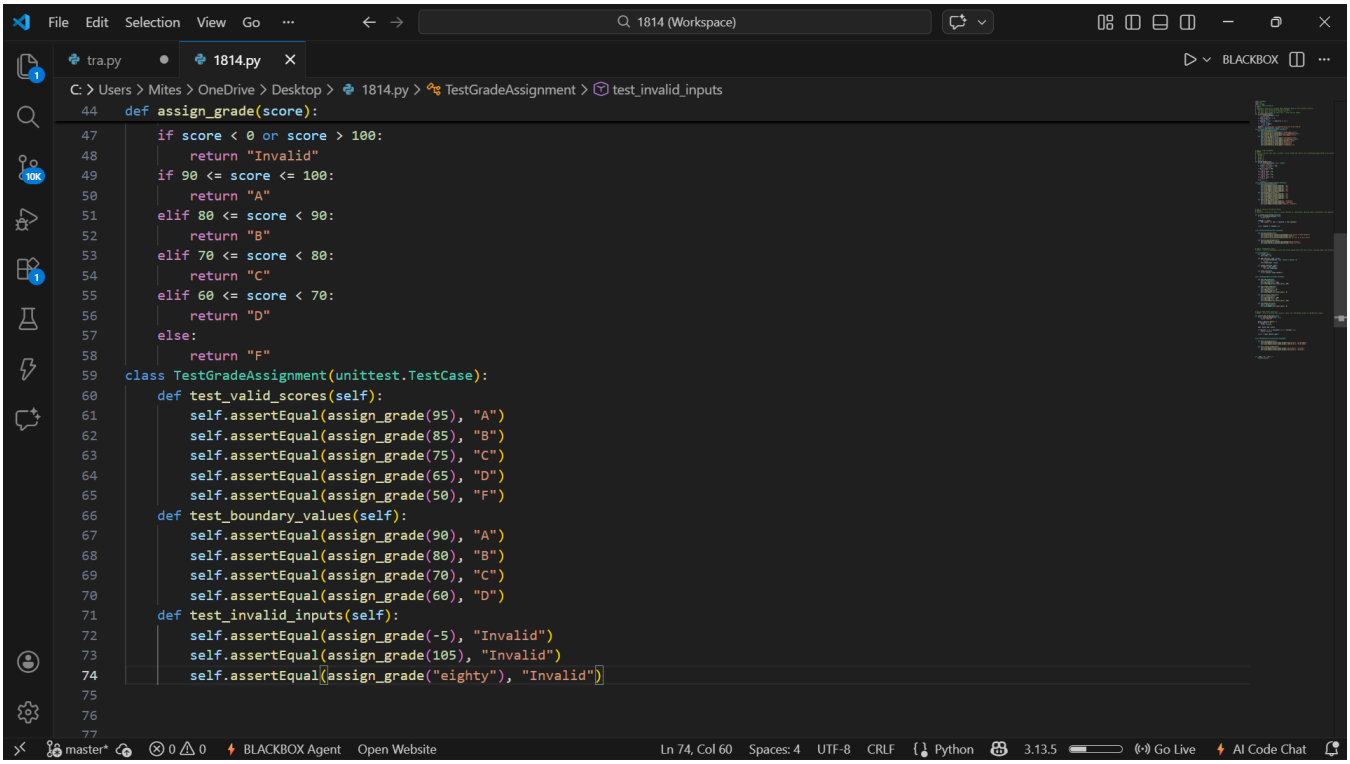
- 80-89: B

- 70-79: C

- 60-69: D

- Below 60: F

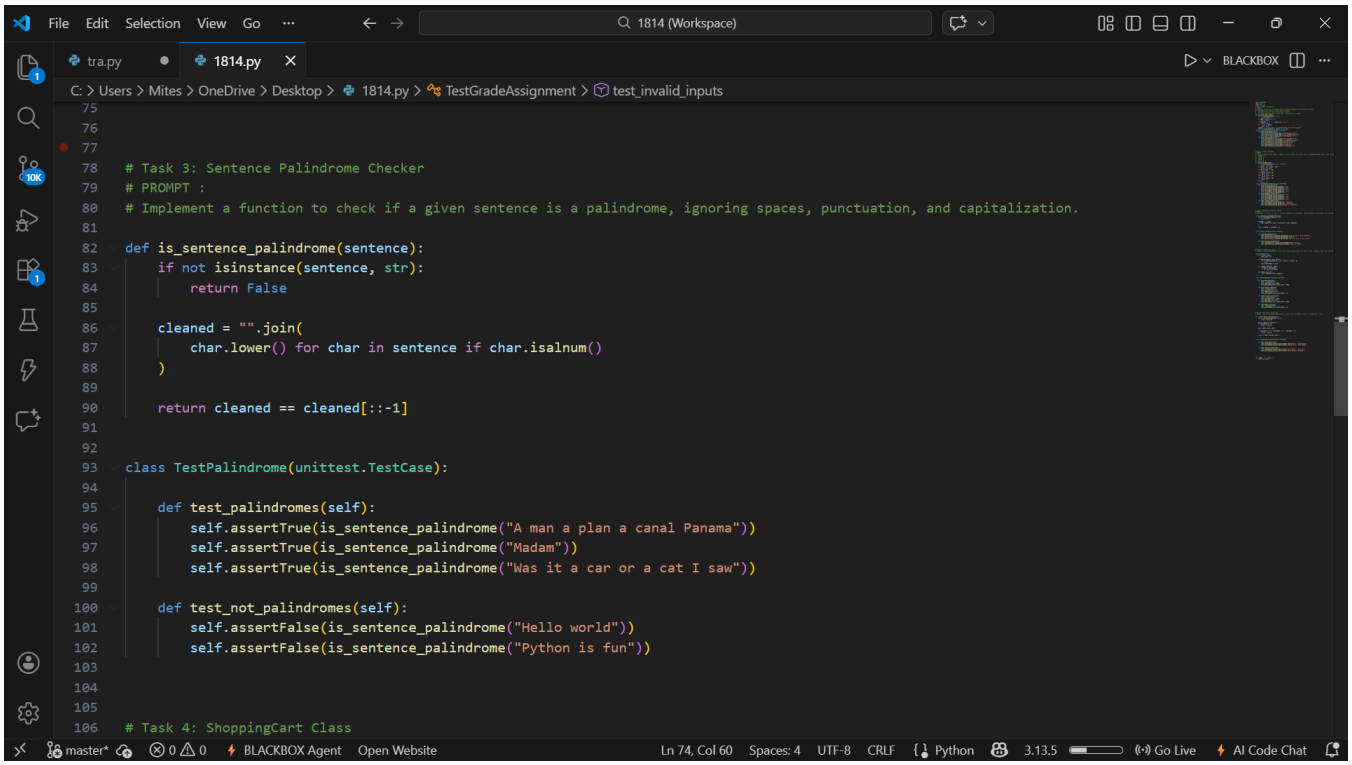



File Edit Selection View Go
1814 (Workspace)
tra.py
1814.py
BLACKBOX
C: > Users > Mites > OneDrive > Desktop > 1814.py > TestGradeAssignment > test_invalid_inputs
def assign_grade(score):
if score < 0 or score > 100:
return "Invalid"
if 90 <= score <= 100:
return "A"
elif 80 <= score < 90:
return "B"
elif 70 <= score < 80:
return "C"
elif 60 <= score < 70:
return "D"
else:
return "F"
class TestGradeAssignment(unittest.TestCase):
def test_valid_scores(self):
self.assertEqual(assign_grade(95), "A")
self.assertEqual(assign_grade(85), "B")
self.assertEqual(assign_grade(75), "C")
self.assertEqual(assign_grade(65), "D")
self.assertEqual(assign_grade(50), "F")
def test_boundary_values(self):
self.assertEqual(assign_grade(90), "A")
self.assertEqual(assign_grade(80), "B")
self.assertEqual(assign_grade(70), "C")
self.assertEqual(assign_grade(60), "D")
def test_invalid_inputs(self):
self.assertEqual(assign_grade(-5), "Invalid")
self.assertEqual(assign_grade(105), "Invalid")
self.assertEqual(assign_grade("eighty"), "Invalid")
master*
BLACKBOX Agent
Open Website
Ln 74, Col 60
Spaces: 4
UTF-8
CRLF
Python
3.13.5
Go Live
AI Code Chat

Task 3: Sentence Palindrome Checker #

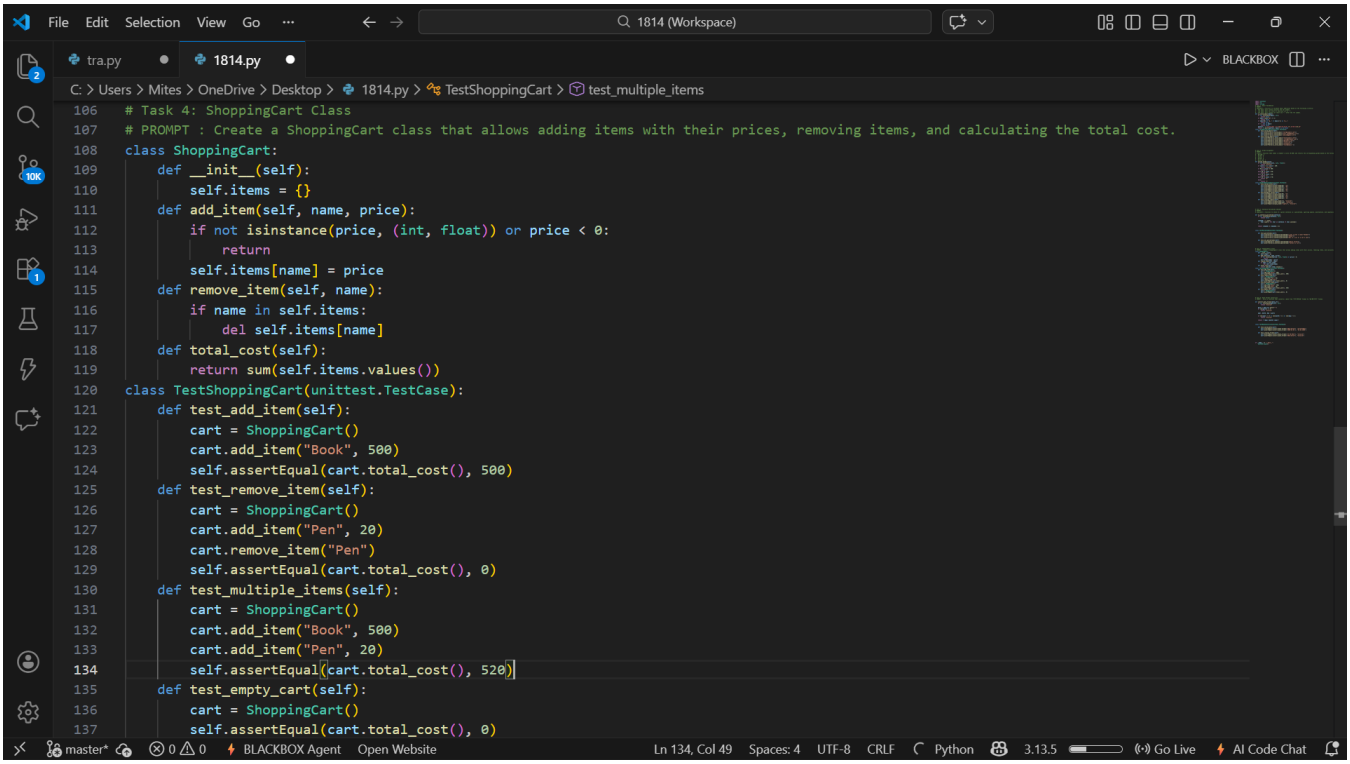
PROMPT :

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



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)
tra.py
1814.py
BLACKBOX
C: > Users > Mites > OneDrive > Desktop > 1814.py > TestShoppingCart > test_multiple_items
# Task 4: ShoppingCart Class
# PROMPT : Create a ShoppingCart class that allows adding items with their prices, removing items, and calculating the total cost.
class ShoppingCart:
def __init__(self):
self.items = {}
def add_item(self, name, price):
if not isinstance(price, (int, float)) or price < 0:
return
self.items[name] = price
def remove_item(self, name):
if name in self.items:
del self.items[name]
def total_cost(self):
return sum(self.items.values())
class TestShoppingCart(unittest.TestCase):
def test_add_item(self):
cart = ShoppingCart()
cart.add_item("Book", 500)
self.assertEqual(cart.total_cost(), 500)
def test_remove_item(self):
cart = ShoppingCart()
cart.add_item("Pen", 20)
cart.remove_item("Pen")
self.assertEqual(cart.total_cost(), 0)
def test_multiple_items(self):
cart = ShoppingCart()
cart.add_item("Book", 500)
cart.add_item("Pen", 20)
self.assertEqual(cart.total_cost(), 520)
def test_empty_cart(self):
cart = ShoppingCart()
self.assertEqual(cart.total_cost(), 0)
master*
BLACKBOX Agent
Open Website
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.