

2303A51563

batch=10

Task 1: Email Validation using TDD

Scenario

You are developing a user registration system that requires reliable email input validation.

Requirements

- Must contain @ and . characters
- Must not start or end with special characters
- Should not allow multiple @ symbols
- AI should generate test cases covering valid and invalid email formats
- Implement is_valid_email(email) to pass all AI-generated test cases

Expected Output

- Python function for email validation
- All AI-generated test cases pass successfully
- Invalid email formats are correctly rejected
- Valid email formats return True

Code

```
import unittest
```

```
# -----
```

```
# Email Validation Function
```

```
# -----
```

```
def is_valid_email(email):
```

```
    # Must contain exactly one '@'
```

```
    if email.count("@") != 1:
```

```
        return False
```

```
    # Must contain at least one '.'
```

```
    if "." not in email:
```

```
        return False
```

```
    # Must not start or end with special characters
```

```
    special_chars = "@._-"
```

```
    if email[0] in special_chars or email[-1] in special_chars:
```

```
        return False
```

```
    # Split into local part and domain part
```

```
    local, domain = email.split("@")
```

```
    # Local and domain must not be empty
```

```
    if local == "" or domain == "":
```

```
        return False
```

```
# Domain must contain a dot (example: gmail.com)
if "." not in domain:
    return False
```

```
# Domain must not start or end with dot
if domain[0] == "." or domain[-1] == ".":
    return False
```

```
return True
```

```
# -----
# TDD Test Cases
# -----
```

```
class TestEmailValidation(unittest.TestCase):
```

```
#  Valid Emails
```


```
def test_valid_emails(self):
    self.assertTrue(is_valid_email("user@gmail.com"))
    self.assertTrue(is_valid_email("john.doe@yahoo.in"))
    self.assertTrue(is_valid_email("student123@university.edu"))
    self.assertTrue(is_valid_email("my_mail@domain.co"))
```

```
#  Invalid Emails
```

```
def test_invalid_emails(self):
    self.assertFalse(is_valid_email("usergmail.com"))    # Missing @
    self.assertFalse(is_valid_email("user@gmailcom"))    # Missing dot
    self.assertFalse(is_valid_email("@gmail.com"))       # Starts with special char
    self.assertFalse(is_valid_email("user@gmail.com@abc")) # Multiple @
    self.assertFalse(is_valid_email("user@.com"))         # Domain starts with dot
    self.assertFalse(is_valid_email("user@gmail.com."))  # Ends with dot
    self.assertFalse(is_valid_email(".user@gmail.com"))  # Starts with dot
    self.assertFalse(is_valid_email("user@domain"))      # No dot in domain
```

```
# -----
# Run Tests
# -----
```

```
if __name__ == "__main__":
    unittest.main()
```

**OnlineGDB**
online compiler and debugger for c/c++

Welcome, **2303A51563** ▲

Create New Project

My Projects

Classroom **new**

Learn Programming

Programming Questions

Upgrade

Logout ▼

About • FAQ • Blog • Terms of Use •
Contact Us • GDB Tutorial • Credits •
Privacy
© 2016 - 2026 GDB Online

main.py

```
1 import unittest
2
3 # -----
4 # Email Validation Function
5 # -----
6
7 def is_valid_email(email):
8     # Must contain exactly one '@'
9     if email.count("@") != 1:
10         return False
11
12     # Must contain at least one '.'
13     if "." not in email:
14         return False
15
16     # Must not start or end with special characters
17     special_chars = "@._-"
18     if email[0] in special_chars or email[-1] in special_chars:
19         return False
20
21     # Split into local part and domain part
22     local, domain = email.split("@")
23
24     # Local and domain must not be empty
```


input

..

Ran 2 tests in 0.000s

OK

...Program finished with exit code 0
Press ENTER to exit console.

**OnlineGDB**
online compiler and debugger for c/c++

Welcome, **2303A51563** ▲

Create New Project

My Projects

Classroom **new**

Learn Programming

Programming Questions

Upgrade

Logout ▼

About • FAQ • Blog • Terms of Use •
Contact Us • GDB Tutorial • Credits •
Privacy
© 2016 - 2026 GDB Online

main.py

```
22 local, domain = email.split("@")
23
24 # Local and domain must not be empty
25 if local == "" or domain == "":
26     return False
27
28 # Domain must contain a dot (example: gmail.com)
29 if "." not in domain:
30     return False
31
32 # Domain must not start or end with dot
33 if domain[0] == "." or domain[-1] == ".":
34     return False
35
36 return True
37
38 # -----
39 # TDD Test Cases
40 # -----
41
42 class TestEmailValidation(unittest.TestCase):
43
44     # Valid Email
```

input

..

Ran 2 tests in 0.000s

OK

...Program finished with exit code 0
Press ENTER to exit console.

OnlineGDB

online compiler and debugger for c/c++

Welcome, 2303A51563 ▲

Create New Project

My Projects

Classroom new

Learn Programming

Programming Questions

Upgrade

Logout ▼

About • FAQ • Blog • Terms of Use •

Contact Us • GDB Tutorial • Credits •

Privacy

© 2016 - 2026 GDB Online

Run

Debug

Stop

Share

Save

{ } Beautify

⬇

Language Python3 ⌵ ⚙

main.py

```

45 # ✓ Valid Emails
46 def test_valid_emails(self):
47     self.assertTrue(is_valid_email("user@gmail.com"))
48     self.assertTrue(is_valid_email("john.doe@yahoo.in"))
49     self.assertTrue(is_valid_email("student123@university.edu"))
50     self.assertTrue(is_valid_email("my_mail@domain.co"))
51
52 # ✗ Invalid Emails
53 def test_invalid_emails(self):
54     self.assertFalse(is_valid_email("usergmail.com"))           # Missing @
55     self.assertFalse(is_valid_email("user@gmailcom"))           # Missing dot
56     self.assertFalse(is_valid_email("@gmail.com"))              # Starts with special char
57     self.assertFalse(is_valid_email("user@gmail.com@abc"))       # Multiple @
58     self.assertFalse(is_valid_email("user@.com"))                # Domain starts with dot
59     self.assertFalse(is_valid_email("user@gmail.com."))          # Ends with dot
60     self.assertFalse(is_valid_email(".user@gmail.com"))          # Starts with dot
61     self.assertFalse(is_valid_email("user@domain"))              # No dot in domain
62
63
64 # -----
65 # Run Tests
66 # -----
67

```

input

```

..
-----
Ran 2 tests in 0.000s
OK
...Program finished with exit code 0
Press ENTER to exit console.

```

OnlineGDB

online compiler and debugger for c/c++

Welcome, 2303A51563 ▲

Create New Project

My Projects

Classroom new

Learn Programming

Programming Questions

Upgrade

Logout ▼

About • FAQ • Blog • Terms of Use •

Contact Us • GDB Tutorial • Credits •

Privacy

© 2016 - 2026 GDB Online

Run

Debug

Stop

Share

Save

{ } Beautify

⬇

Language Python3 ⌵ ⚙

main.py

```

47 self.assertTrue(is_valid_email("user@gmail.com"))
48 self.assertTrue(is_valid_email("john.doe@yahoo.in"))
49 self.assertTrue(is_valid_email("student123@university.edu"))
50 self.assertTrue(is_valid_email("my_mail@domain.co"))
51
52 # ✗ Invalid Emails
53 def test_invalid_emails(self):
54     self.assertFalse(is_valid_email("usergmail.com"))           # Missing @
55     self.assertFalse(is_valid_email("user@gmailcom"))           # Missing dot
56     self.assertFalse(is_valid_email("@gmail.com"))              # Starts with special char
57     self.assertFalse(is_valid_email("user@gmail.com@abc"))       # Multiple @
58     self.assertFalse(is_valid_email("user@.com"))                # Domain starts with dot
59     self.assertFalse(is_valid_email("user@gmail.com."))          # Ends with dot
60     self.assertFalse(is_valid_email(".user@gmail.com"))          # Starts with dot
61     self.assertFalse(is_valid_email("user@domain"))              # No dot in domain
62
63
64 # -----
65 # Run Tests
66 # -----
67
68 if __name__ == "__main__":
69     unittest.main()
70

```

input

```

..
-----
Ran 2 tests in 0.000s
OK
...Program finished with exit code 0
Press ENTER to exit console.

```

Task 2: Grade Assignment using Loops

Scenario

You are building an automated grading system for an online examination platform.

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

- Include boundary values (60, 70, 80, 90)
- Include invalid inputs such as -5, 105, "eighty"
- Implement the function using a test-driven approach

Expected Output

- Grade assignment function implemented in Python
- Boundary values handled correctly
- Invalid inputs handled gracefully
- All AI-generated test cases pass

Code

```
def assign_grade(score):
    if not isinstance(score, int) or score < 0 or score > 100:
        return "Invalid"

    if score >= 90: return "A"
    if score >= 80: return "B"
    if score >= 70: return "C"
    if score >= 60: return "D"
    return "F"
```

Test Cases

```
print(assign_grade(90)) # A
print(assign_grade(80)) # B
print(assign_grade(70)) # C
print(assign_grade(60)) # D
print(assign_grade(50)) # F
print(assign_grade(-5)) # Invalid
print(assign_grade(105)) # Invalid
print(assign_grade("eighty")) # Invalid
```

Task 2: Grade Assignment using Loops

Scenario

You are building an automated grading system for an online examination platform.

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
- Include boundary values (60, 70, 80, 90)
- Include invalid inputs such as -5, 105, "eighty"
- Implement the function using a test-driven approach

Expected Output

- Grade assignment function implemented in Python
- Boundary values handled correctly
- Invalid inputs handled gracefully
- All AI-generated test cases pass

Code

```
def assign_grade(score):  
    if not isinstance(score, int) or score < 0 or score > 100:  
        return "Invalid"  
  
    if score >= 90: return "A"  
    if score >= 80: return "B"  
    if score >= 70: return "C"  
    if score >= 60: return "D"  
    return "F"
```

Test Cases

```
print(assign_grade(90)) # A  
print(assign_grade(80)) # B  
print(assign_grade(70)) # C  
print(assign_grade(60)) # D  
print(assign_grade(50)) # F  
print(assign_grade(-5)) # Invalid  
print(assign_grade(105)) # Invalid  
print(assign_grade("eighty")) # Invalid
```

The screenshot shows the OnlineGDB web interface. On the left is a sidebar with navigation links: 'Welcome, 2303A51563', 'Create New Project', 'My Projects', 'Classroom new', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area displays a Python script named 'main.py' with the following code:

```
1 def assign_grade(score):
2     if not isinstance(score, int) or score < 0 or score > 100:
3         return "Invalid"
4
5     if score >= 90: return "A"
6     if score >= 80: return "B"
7     if score >= 70: return "C"
8     if score >= 60: return "D"
9     return "F"
10
11
12 # Test Cases
13 print(assign_grade(90)) # A
14 print(assign_grade(80)) # B
15 print(assign_grade(70)) # C
16 print(assign_grade(60)) # D
17 print(assign_grade(50)) # F
18 print(assign_grade(-5)) # Invalid
19 print(assign_grade(105)) # Invalid
20 print(assign_grade("eighty")) # Invalid
21
```

Below the code editor is an input/output console. It shows the output of the test cases: 'D', 'F', 'Invalid', 'Invalid', 'Invalid', and 'Invalid'. At the bottom, it states '...Program finished with exit code 0' and 'Press ENTER to exit console.'

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

Expected Output

- Function correctly identifies sentence palindromes
- Case and punctuation are ignored
- Returns True or False accurately
- All AI-generated test cases pass

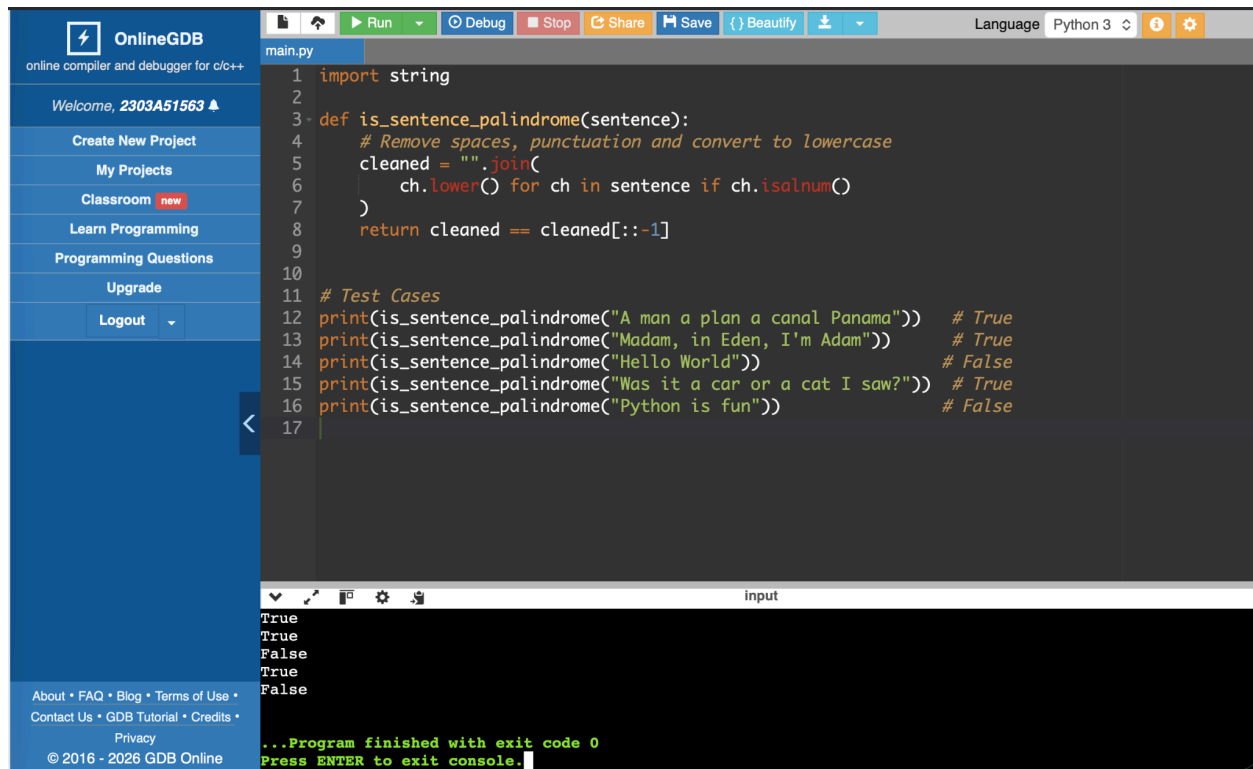
Code

import string

```
def is_sentence_palindrome(sentence):
    # Remove spaces, punctuation and convert to lowercase
    cleaned = "".join(
        ch.lower() for ch in sentence if ch.isalnum()
    )
    return cleaned == cleaned[::-1]
```

Test Cases

```
print(is_sentence_palindrome("A man a plan a canal Panama")) # True
print(is_sentence_palindrome("Madam, in Eden, I'm Adam"))    # True
print(is_sentence_palindrome("Hello World"))                  # False
print(is_sentence_palindrome("Was it a car or a cat I saw?")) # True
print(is_sentence_palindrome("Python is fun"))                 # False
```



The screenshot shows the OnlineGDB web interface. On the left is a sidebar with navigation links: 'Welcome, 2303A51563', 'Create New Project', 'My Projects', 'Classroom' (marked 'new'), 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area displays a Python file named 'main.py' with the following code:

```
1 import string
2
3 def is_sentence_palindrome(sentence):
4     # Remove spaces, punctuation and convert to lowercase
5     cleaned = "".join(
6         ch.lower() for ch in sentence if ch.isalnum()
7     )
8     return cleaned == cleaned[::-1]
9
10
11 # Test Cases
12 print(is_sentence_palindrome("A man a plan a canal Panama")) # True
13 print(is_sentence_palindrome("Madam, in Eden, I'm Adam"))    # True
14 print(is_sentence_palindrome("Hello World"))                  # False
15 print(is_sentence_palindrome("Was it a car or a cat I saw?")) # True
16 print(is_sentence_palindrome("Python is fun"))                 # False
17
```

Below the code editor is an 'input' field and an output console. The console shows the output of the program:

```
True
True
False
True
False
...Program finished with exit code 0
Press ENTER to exit console.
```

Task 4: ShoppingCart Class

Scenario

You are designing a basic shopping cart module for an e-commerce application.

Requirements

- AI should generate test cases for the ShoppingCart class
- Class must include the following methods:
 - add_item(name, price)
 - remove_item(name)
 - total_cost()
- Validate correct addition, removal, and cost calculation
- Handle empty cart scenarios

Expected Output

- Fully implemented ShoppingCart class
- All methods pass AI-generated test cases
- Total cost is calculated accurately
- Items are added and removed correctly

Code

```
class ShoppingCart:
    def __init__(self):
        self.items = {}

    def add_item(self, name, price):
        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())

# ----- Test Cases -----

cart = ShoppingCart()

# Empty cart test
print(cart.total_cost()) # 0

# Add items
cart.add_item("Book", 200)
cart.add_item("Pen", 50)

print(cart.total_cost()) # 250

# Remove item
cart.remove_item("Pen")
print(cart.total_cost()) # 200

# Remove non-existing item (no error)
cart.remove_item("Laptop")

print(cart.total_cost()) # 200
```

OnlineGDB
online compiler and debugger for c/c++

Welcome, 2303A51563 ▲

Create New Project

My Projects

Classroom **new**

Learn Programming

Programming Questions

Upgrade

Logout ▾

About • FAQ • Blog • Terms of Use •
Contact Us • GDB Tutorial • Credits •
Privacy
© 2016 - 2026 GDB Online

```
1 class ShoppingCart:
2     def __init__(self):
3         self.items = {}
4
5     def add_item(self, name, price):
6         self.items[name] = price
7
8     def remove_item(self, name):
9         if name in self.items:
10            del self.items[name]
11
12    def total_cost(self):
13        return sum(self.items.values())
14
15
16 # ----- Test Cases -----
17
18 cart = ShoppingCart()
19
20 # Empty cart test
21 print(cart.total_cost()) # 0
22
23 # Add items
24 cart.add_item("Book", 200)
```

input

0
250
200
200

...Program finished with exit code 0
Press ENTER to exit console.

OnlineGDB
online compiler and debugger for c/c++

Welcome, 2303A51563 ▲

Create New Project

My Projects

Classroom **new**

Learn Programming

Programming Questions

Upgrade

Logout ▾

About • FAQ • Blog • Terms of Use •
Contact Us • GDB Tutorial • Credits •
Privacy
© 2016 - 2026 GDB Online

```
15
16 # ----- Test Cases -----
17
18 cart = ShoppingCart()
19
20 # Empty cart test
21 print(cart.total_cost()) # 0
22
23 # Add items
24 cart.add_item("Book", 200)
25 cart.add_item("Pen", 50)
26
27 print(cart.total_cost()) # 250
28
29 # Remove item
30 cart.remove_item("Pen")
31 print(cart.total_cost()) # 200
32
33 # Remove non-existing item (no error)
34 cart.remove_item("Laptop")
35
36 print(cart.total_cost()) # 200
37
38
```

input

0
250
200
200

...Program finished with exit code 0
Press ENTER to exit console.

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"

Expected Output

- Date conversion function implemented in Python
- Correct format conversion for all valid inputs
- All AI-generated test cases pass successfully

Code

```
def convert_date_format(date_str):
```

```
    # Split YYYY-MM-DD into parts
```

```
    y, m, d = date_str.split("-")
```

```
    return f"{d}-{m}-{y}"
```

```
# ----- Test Cases -----
```

```
print(convert_date_format("2023-10-15")) # 15-10-2023
```

```
print(convert_date_format("2024-01-01")) # 01-01-2024
```

```
print(convert_date_format("1999-12-31")) # 31-12-1999
```

The screenshot shows the OnlineGDB web interface. On the left is a sidebar with navigation links: 'Welcome, 2303A51563', 'Create New Project', 'My Projects', 'Classroom new', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area displays a Python script in a dark-themed editor. The script defines a function `convert_date_format` that splits a date string by hyphens and returns the components in reverse order. Below the function, three test cases are printed. The bottom panel shows the output of the program, which matches the expected results. The status bar at the bottom indicates the program finished successfully.

```

1 def convert_date_format(date_str):
2     # Split YYYY-MM-DD into parts
3     y, m, d = date_str.split("-")
4     return f"{d}-{m}-{y}"
5
6
7 # ----- Test Cases -----
8 print(convert_date_format("2023-10-15")) # 15-10-2023
9 print(convert_date_format("2024-01-01")) # 01-01-2024
10 print(convert_date_format("1999-12-31")) # 31-12-1999
11

```

input

```

15-10-2023
01-01-2024
31-12-1999

```

...Program finished with exit code 0
Press ENTER to exit console.