

ASSIGNMENT- 8.3

Name: POOJITHA EDDE

HT.No: 2303A51356

Batch: 20

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:

```
File Edit Selection View Go Run Terminal Help < → Q ai ass coding
EXPLORER ... assignment-8.3.py • Assignment-6.3.py • Assignment-6.5.py • Assignment-7.5.py • Assignment-8.3.py • .py × Dv ...
Assignment-1.5.py
Assignment-2.5.py
Assignment-3.5.py
Assignment-4.5.py
Assignment-5.5.py
Assignment-6.3.py
Assignment-6.5.py
Assignment-7.5.py
Assignment-8.3.py
Assignment-9.3.py
exp-02_A.py
exp-02_B.py

import re
def is_valid_email(email):
    # Check for exactly one @ symbol
    if email.count('@') != 1:
        return False
    # Check for presence of . character
    if '.' not in email:
        return False
    # Check for valid characters and structure using regex
    pattern = r'^[a-zA-Z0-9._+]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    return re.match(pattern, email) is not None

# Test cases
test_emails = [
    "user@example.com",           # Valid
    "invalid.email",              # Invalid - no @
    "user@.com",                  # Invalid - no domain name
    "@example.com",               # Invalid - no local part
    "user@example",               # Invalid - no TLD
    "user@example.com",           # Invalid - multiple dots
    "user@example.com",           # Invalid - multiple @ symbols
    ".user@example.com",          # Invalid - starts with special character
    "user@example.com.",          # Invalid - ends with special character
    "user@example ple.com",       # Invalid - space in domain name
    "user@exam ple.com",          # Valid - underscore in domain name
    "user@example.co.uk",          # Valid - multi-level domain
    "user+tag@example.com",        # Valid - plus in local part
    "user_name@example-domain.com" # Valid - underscore and hyphen in local part and domain name
]
for email in test_emails:
    print(f"{email}: {is_valid_email(email)}")
#Expected Output: True for valid emails, False for invalid ones.
```

Ln 38, Col 65 Spaces:4 UTF-8 CRLF () Python 3.13.12 (Microsoft Store)

OUTPUT:

```

File Edit Selection View Go Run Terminal Help < > ai ass coding
EXPLORER ... Assignment-5.5.py Assignment-6.3.py Assignment-6.5.py Assignment-7.5.py Assignment-8.3.py .py ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v ... | x ...
CHAT + v ... | x ...
SESSIONS ... Fibonacci sequence generation up to n terms... Completed in 5s. 1 mo ago
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding & c:/users/pooji/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/users/pooji/OneDrive/Desktop/ai ass coding/.py"
user@example.com: True
invalid_email: False
user@.com: False
@example.com: False
user@example.: False
user@example..com: True
user@example.com: False
.user@example.com: True
user@example.com.: False
@example.com: False
user@example..com: True
user@example.com: False
.user@example.com: True
user@example..com: False
user@example..com: True
user@example.com: False
.user@example.com: True
user@example..com: False
user@example..com: True
user@example.com: False
.user@example.com: True
user@example..com: True
user@example..com: False
user@example.co.uk: True
user@agileexample.com: True
user@agileexample.com: True
user@agileexample.com: True
user_name@example-domain.com: True
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding>

```

Ln 38, Col 65 Spaces:4 UTRF-8 CRLF () Python 3.13.12 (Microsoft Store) ⚡

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:

```
File Edit Selection View Go Run Terminal Help < > ai ass coding
EXPLORER
AI ASS CODING
Assignment-1.5.py
Assignment-2.5.py
Assignment-3.5.py
Assignment-4.5.py
Assignment-5.5.py
Assignment-6.3.py
Assignment-6.5.py
Assignment-7.5.py
Assignment-8.3.py
exp-02_A.py
exp-02_B.py

.py > ...
1 #generate automated grading system for an online examination platform.
2 # AI should generate test cases for assign_grade(score) where:
3 # 90-100 => A
4 # 80-89 => B
5 # 70-79 => C
6 # 60-69 => D
7 # Below 60 => F
8 # Include boundary values (60, 70, 80, 90)
9 # Include invalid inputs such as -5, 105, "eighty"
10 # Implement the function using a test-driven approach
11 def assign_grade(score):
12     if isinstance(score, str) or score < 0 or score > 100:
13         return "Invalid input"
14     if score >= 90:
15         return 'A'
16     elif score >= 80:
17         return 'B'
18     elif score >= 70:
19         return 'C'
20     elif score >= 60:
21         return 'D'
22     else:
23         return 'F'
24 # Test cases
25 test_scores = [95, 85, 75, 65, 55, 60, 70, 80, 90, -5, 105, "eighty"]
26 for score in test_scores:
27     print(f"({score}): {assign_grade(score)}")
28 #Expected Output: Correct grade for valid scores, "Invalid input" for invalid ones.
29

Chat
SESSIONS
Fibonacci sequence generation up to n terms... Completed in 5s. 1 mo ago
```

OUTPUT:

```
File Edit Selection View Go Run Terminal Help < > ai ass coding
EXPLORER
AI ASS CODING
Assignment-1.5.py
Assignment-2.5.py
Assignment-3.5.py
Assignment-4.5.py
Assignment-5.5.py
Assignment-6.3.py
Assignment-6.5.py
Assignment-7.5.py
Assignment-8.3.py
exp-02_A.py
exp-02_B.py

.py > ...
1 #generate automated grading system for an online examination platform.
2 # AI should generate test cases for assign_grade(score) where:
3 # 90-100 => A
4 # 80-89 => B
5 # 70-79 => C
6 # 60-69 => D
7 # Below 60 => F
8 # Include boundary values (60, 70, 80, 90)
9 # Include invalid inputs such as -5, 105, "eighty"
10 # Implement the function using a test-driven approach
11 def assign_grade(score):
12     if isinstance(score, str) or score < 0 or score > 100:
13         return "Invalid input"
14     if score >= 90:
15         return 'A'
16     elif score >= 80:
17         return 'B'
18     elif score >= 70:
19         return 'C'
20     elif score >= 60:
21         return 'D'
22     else:
23         return 'F'

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding & c:/users/pooji/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/users/pooji/Desktop/ai ass coding/.py"
95: A
85: B
75: C
65: D
55: F
60: D
70: C
80: B
90: A
-5: Invalid input
105: Invalid input
eighty: Invalid input

Chat
SESSIONS
Fibonacci sequence generation up to n terms... Completed in 5s. 1 mo ago
```

Task 3: Sentence Palindrome Checker

Scenario

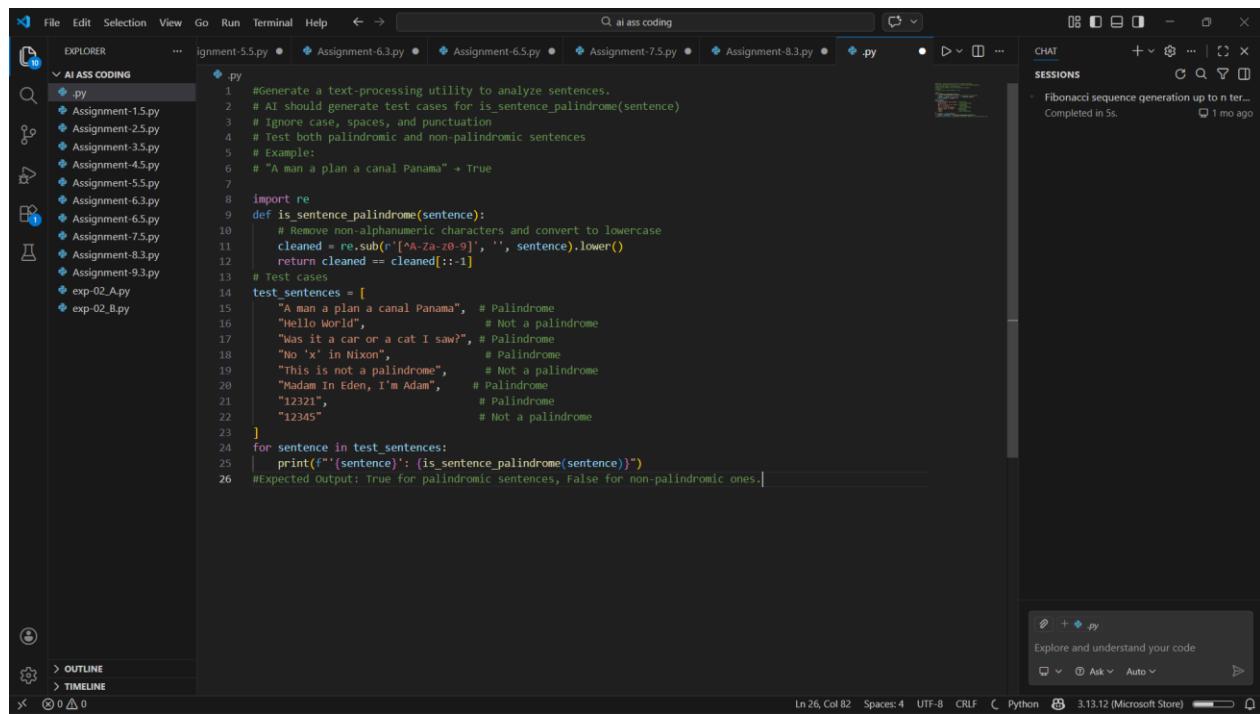
You are developing a text-processing utility to analyze sentences.

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:



```

File Edit Selection View Go Run Terminal Help < > ai ass coding
EXPLORER .py Assignment-1.5.py Assignment-2.5.py Assignment-3.5.py Assignment-4.5.py Assignment-5.5.py Assignment-6.3.py Assignment-6.5.py Assignment-7.5.py Assignment-8.3.py .py
1 #Generate a text-processing utility to analyze sentences.
2 # AI should generate test cases for is_sentence_palindrome(sentence)
3 # Ignore case, spaces, and punctuation
4 # Test both palindromic and non-palindromic sentences
5 # Example:
6 # "A man a plan a canal Panama" + True
7
8 import re
9 def is_sentence_palindrome(sentence):
10     # Remove non-alphanumeric characters and convert to lowercase
11     cleaned = re.sub(r"[^A-Za-z0-9]", "", sentence).lower()
12     return cleaned == cleaned[::-1]
13 # Test cases
14 test_sentences = [
15     "A man a plan a canal Panama", # Palindrome
16     "Hello World", # Not a palindrome
17     "Was it a car or a cat I saw?", # Palindrome
18     "No 'x' in Nixon", # Palindrome
19     "This is not a palindrome", # Not a palindrome
20     "Madam In Eden, I'm Adam", # Palindrome
21     "12321", # Palindrome
22     "12345" # Not a palindrome
23 ]
24 for sentence in test_sentences:
25     print(f"'{sentence}': {is_sentence_palindrome(sentence)}")
26 #Expected Output: True for palindromic sentences, False for non-palindromic ones.

```

CHAT: Fibonacci sequence generation up to n terms... Completed in 5s.

SESSIONS: Fibonacci sequence generation up to n terms... Completed in 5s.

OUTLINE: EXPLORE AND UNDERSTAND YOUR CODE

Ln 26, Col 82 Spaces:4 UTF-8 CRLF Python 3.13.12 (Microsoft Store)

OUTPUT:

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The left sidebar has 'EXPLORER' and 'AI ASS CODING' sections. The main area displays a Python script named 'Assignment-5.5.py'. The script defines a function to check if a sentence is a palindrome, ignoring punctuation and case. It includes several test cases. The terminal below shows the script running and printing its results. The status bar at the bottom indicates the file is 'Assignment-5.5.py'.

```
File Edt Selection View Go Run Terminal Help < > ai ass coding .py
```

```
Assignment-5.5.py ● Assignment-6.3.py ● Assignment-6.5.py ● Assignment-7.5.py ● Assignment-8.3.py ● .py
```

```
#Generate a text-processing utility to analyze sentences.
# 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

import re
def is_sentence_palindrome(sentence):
    # Remove non-alphanumeric characters and convert to lowercase
    cleaned = re.sub("[^A-Za-z0-9]", "", sentence).lower()
    return cleaned == cleaned[::-1]

# test cases
test_sentences = [
    "A man a plan a canal Panama", # Palindrome
    "Hello World", # Not a palindrome
    "Was it a car or a cat I saw?", # Palindrome
]

for sentence in test_sentences:
    print(f'{sentence}: {is_sentence_palindrome(sentence)}')

# Output from terminal
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding> & C:/Users/pooji/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/pooji/OneDrive/Desktop/ai ass coding/.py"
'A man a plan a canal Panama': True
'Hello World': False
'Was it a car or a cat I saw?': True
'No 'x' in Nixon': True
'This is not a palindrome': False
'Madam In Eden, I'm Adam': True
'12321': True
'12345': False
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding>
```

```
OUTLINE > TIMELINE
```

```
CHAT SESSIONS + Fibonacci sequence generation up to n terms... Completed in 5s. 1 ms ago
```

```
Explore and understand your code
```

```
In 26, Col 82 Spaces: 4 UTF-8 CRLF () Python 3.13.12 (Microsoft Store)
```

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:

A screenshot of the Microsoft Visual Studio Code interface. The main area shows a Python file named 'it-6.3.py' containing code for a ShoppingCart class. The code includes methods for adding items, removing items, and calculating total cost. A series of print statements are used to verify the correctness of these methods. The terminal at the bottom shows the command 'python it-6.3.py' being run and the output of the program.

```
#Generate a basic shopping cart module for an e-commerce application.
# 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

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()
cart.add_item("book", 12.99)
cart.add_item("Pen", 1.50)
print(f"Total cost after adding items: {cart.total_cost()}") # Expected: 14.49
cart.remove_item("Pen")
print(f"Total cost after removing Pen: {cart.total_cost()}") # Expected: 12.99
cart.remove_item("Notebook") # Removing non-existing item
print(f"Total cost after trying to remove non-existing item: {cart.total_cost()}") # Expected: 12.99
cart.remove_item("Book")
print(f"Total cost after removing Book: {cart.total_cost()}") # Expected: 0.0
```

OUTPUT:

A screenshot of the Microsoft Visual Studio Code interface, similar to the previous one but with the terminal window open. The terminal shows the command 'python it-6.3.py' being run and the output of the program, which is the same as the code editor output above.

```
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding> & c:/users/pooji/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/users/pooji/OneDrive/Desktop/ai ass coding/it-6.3.py"
Total cost after adding items: 14.49
Total cost after removing Pen: 12.99
Total cost after trying to remove non-existing item: 12.99
Total cost after removing Book: 0
```

Task 5: Date Format Conversion

Scenario

You are creating a utility function to convert date formats for reports.

Requirements

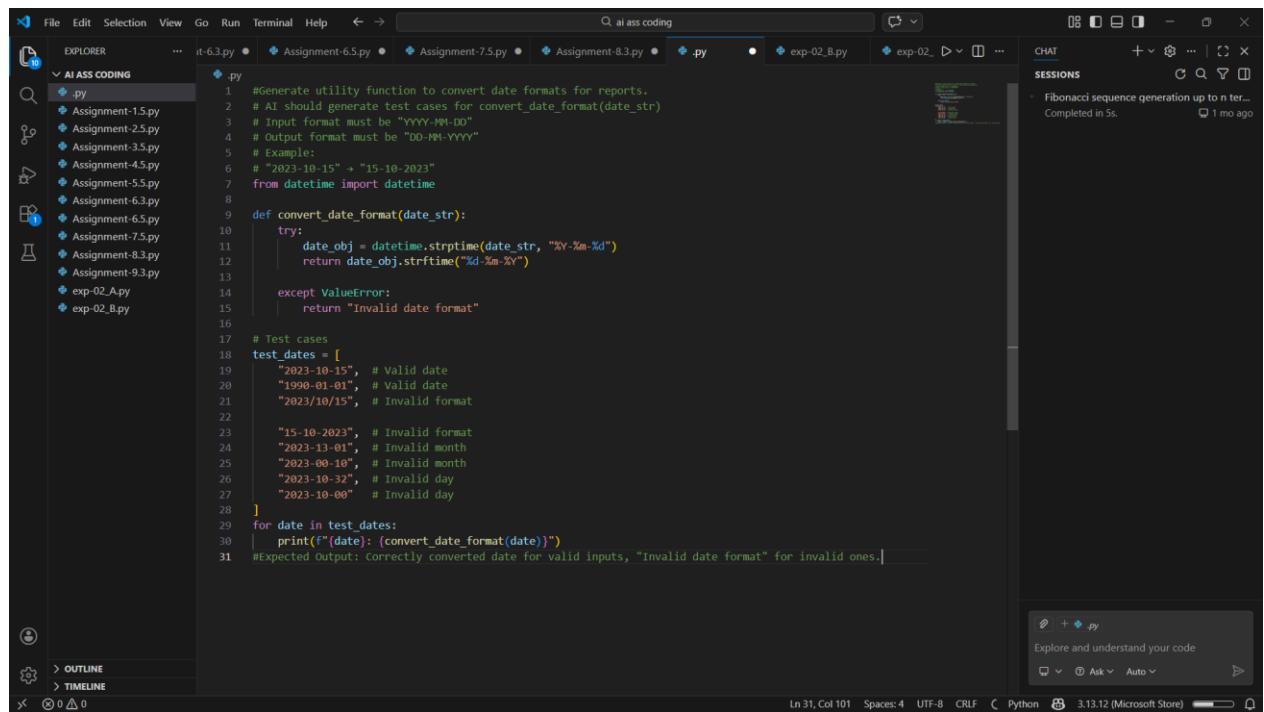
- AI should generate test cases for convert_date_format(date_str)

- 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:



The screenshot shows the Microsoft Visual Studio Code interface. The left sidebar has a tree view titled 'AI ASS CODING' containing files like 'it-6.3.py', 'Assignment-1.5.py', etc. The main code editor window contains the following Python script:

```

1 #Generate utility function to convert date formats for reports.
2 # AI should generate test cases for convert_date_format(date_str)
3 # Input format must be "YYYY-MM-DD"
4 # Output format must be "DD-MM-YYYY"
5 # Example:
6 # "2023-10-15" → "15-10-2023"
7 from datetime import datetime
8
9 def convert_date_format(date_str):
10     try:
11         date_obj = datetime.strptime(date_str, "%Y-%m-%d")
12         return date_obj.strftime("%d-%m-%Y")
13     except ValueError:
14         return "Invalid date format"
15
16 # Test cases
17 test_dates = [
18     "2023-10-15", # Valid date
19     "1990-01-01", # Valid date
20     "2023/10/15", # Invalid format
21
22     "15-10-2023", # Invalid format
23     "2023-13-01", # Invalid month
24     "2023-00-10", # Invalid month
25     "2023-10-32", # Invalid day
26     "2023-10-00" # Invalid day
27 ]
28 for date in test_dates:
29     print(f"{date}: {convert_date_format(date)}")
30
#Expected Output: Correctly converted date for valid inputs, "Invalid date format" for invalid ones.

```

The status bar at the bottom indicates: Ln 31, Col 101 Spaces: 4 UFT-8 CRLF C Python 3.13.12 (Microsoft Store)

OUTPUT:

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "AI ASS CODING" containing several Python files: Assignment-1.5.py, Assignment-2.5.py, Assignment-3.5.py, Assignment-4.5.py, Assignment-5.5.py, Assignment-6.3.py, Assignment-6.5.py, Assignment-7.5.py, Assignment-8.3.py, exp-02_A.py, and exp-02_B.py.
- Terminal:** The terminal window displays the following Python code and its execution results:

```
it-6.3.py > ...
1 #Generate utility function to convert date formats for reports.
2 # AI should generate test cases for convert_date_format(date_str)
3 # Input format must be "YYYY-MM-DD"
4 # Output format must be "DD-MM-YYYY"
5 # Example:
6 # "2023-10-15" + "15-10-2023"
7 from datetime import datetime
8
9 def convert_date_format(date_str):
10     try:
11         date_obj = datetime.strptime(date_str, "%Y-%m-%d")
12         return date_obj.strftime("%d %m %Y")
13     except ValueError:
14         return "Invalid date format"
15
16 # Test cases
17
PS C:\Users\pooji\OneDrive\Desktop\ai ass coding> & c:/Users/pooji/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/pooji/OneDrive/Desktop/ai ass coding/.py"
2023-10-15: 15-10-2023
1990-01-01: 01-01-1990
2023/10/15: Invalid date format
15-10-2023: Invalid date format
2023-13-01: Invalid date format
2023-00-10: Invalid date format
2023-10-32: Invalid date format
2023-10-00: Invalid date format
```
- Output:** The output pane shows the command run in the terminal: "PS C:\Users\pooji\OneDrive\Desktop\ai ass coding> & c:/Users/pooji/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/pooji/OneDrive/Desktop/ai ass coding/.py"
- Chat:** A sidebar titled "SESSIONS" shows a single session: "Fibonacci sequence generation up to n ter... Completed in 5s. 1 mo ago".
- Status Bar:** The status bar at the bottom shows: Ln 31, Col 101, Spaces:4, UTF-8, CRLF, Python 3.13.12 (Microsoft Store).