39.Sudoku Validator Code

```
def is_valid_sudoku(board):
    def is_valid_unit(unit):
        nums = [num for num in unit if num != "."]
        return len(nums) == len(set(nums))  # Check for duplicates

    # Check rows
    for row in board:
        if not is_valid_unit(row):
            return False

    # Check columns
    for col in zip(*board):  # Transpose to check columns
        if not is_valid_unit(col):
            return False

    # Check 3x3 subgrids
    for i in range(0, 9, 3):
        for j in range(0, 9, 3):
            subgrid = [board[x][y] for x in range(i, i+3) for y in range(j, j+3)]
            if not is_valid_unit(subgrid):
                return False

    return True

# Example Sudoku Board
board = [
    ["5","3",".",".","7",".",".",".","."],
    ["6",".",".","1","9","5",".",".","."],
    [".","9","8",".",".",".",".","6","."],
    ["8",".",".",".","6",".",".",".","3"],
    ["4",".",".","8",".","3",".",".","1"],
    ["7",".",".",".","2",".",".",".","6"],
    [".","6",".",".",".",".","2","8","."],
    [".",".",".","4","1","9",".",".","5"],
    [".",".",".",".","8",".",".","7","9"]
]

print(is_valid_sudoku(board))  # Output: True
```

```
True
```

Generate

40.Word Frequency in Text

Close

```
def word_frequency(text):
    words = text.lower().split()  # Convert to lowercase and split into words
    freq = {}  # Dictionary to store word counts

    for word in words:
        word = word.strip(".,!?")  # Remove punctuation
        freq[word] = freq.get(word, 0) + 1  # Count occurrences

    return freq

# Example Input
text = "Hello world! Hello everyone. Welcome to the world of Python."

# Output the word frequency dictionary
print(word_frequency(text))
```

```
{'hello': 2, 'world': 2, 'everyone': 1, 'welcome': 1, 'to': 1, 'the': 1, 'of': 1, 'python': 1}
```

41.0/1 Knapsack Problem using dynamic programming:

```
def knapsack(weights, values, capacity):
    n = len(weights)
    dp = [[0] * (capacity + 1) for _ in range(n + 1)]  # DP table

    for i in range(1, n + 1):
        for w in range(capacity + 1):
            if weights[i - 1] <= w:
                dp[i][w] = max(dp[i - 1][w], values[i - 1] + dp[i - 1][w - weights[i - 1]])
            else:
                dp[i][w] = dp[i - 1][w]
    return dp[n][capacity]  # Max value that can be carried

# Example Input
weights = [2, 3, 4, 5]
values = [3, 4, 5, 6]
capacity = 5

# Output the maximum value
print(knapsack(weights, values, capacity))
```

```
7
```

### 42. Merge Intervals

```
def merge_intervals(intervals):
    if not intervals:
        return []

    # Step 1: Sort intervals based on the start time
    intervals.sort(key=lambda x: x[0])
    merged = [intervals[0]]

    for start, end in intervals[1:]:
        last_end = merged[-1][1]

        # Step 2: Check if intervals overlap
        if start <= last_end:
            merged[-1][1] = max(last_end, end)  # Merge overlapping intervals
        else:
            merged.append([start, end])  # Add non-overlapping interval

    return merged

# Example Input
intervals = [[1, 3], [2, 6], [8, 10], [15, 18]]

# Output the merged intervals
print(merge_intervals(intervals))
```

```
[[1, 6], [8, 10], [15, 18]]
```

### 43. Find the Median of Two Sorted Arrays

```
def find_median_sorted_arrays(nums1, nums2):
    # Merge the two sorted arrays
    merged = sorted(nums1 + nums2)
    n = len(merged)

    # Find the median
    if n % 2 == 1:
        return merged[n // 2]  # Odd length case
    else:
        return (merged[n // 2 - 1] + merged[n // 2]) / 2  # Even length case

# Example Input
nums1 = [1, 3]
nums2 = [2]

# Output the median
print(find_median_sorted_arrays(nums1, nums2))
```

```
2
```

### 44. Maximal Rectangle in Binary Matri

```
def maximalRectangle(matrix):
    if not matrix or not matrix[0]:
        return 0

    # Get the number of rows and columns
    rows, cols = len(matrix), len(matrix[0])

    # Heights array to store histogram heights
    heights = [0] * cols
    max_area = 0

    for row in matrix:
        for j in range(cols):
            heights[j] = heights[j] + 1 if row[j] == "1" else 0  # Update heights

        max_area = max(max_area, largestRectangleArea(heights))  # Compute max area

    return max_area

# Helper function to find the largest rectangle in a histogram
def largestRectangleArea(heights):
    stack = []
    max_area = 0
    heights.append(0)  # Add a zero-height bar to pop remaining elements

    for i, h in enumerate(heights):
        while stack and heights[stack[-1]] > h:
            height = heights[stack.pop()]
            width = i if not stack else i - stack[-1] - 1
            max_area = max(max_area, height * width)

        stack.append(i)

    heights.pop()  # Remove the zero-height bar
    return max_area

# Example Input
matrix = [
    ["1", "0", "1", "0", "0"],
    ["1", "0", "1", "1", "1"],
    ["1", "1", "1", "1", "1"],
    ["1", "0", "0", "1", "0"]
]

# Output the maximal rectangle area
print(maximalRectangle(matrix))
```

```
6
```

## 45. Largest Sum Contiguous Subarray (Kadane's Algorithm)

```
def maxSubArray(nums):
    max_sum = float('-inf')  # Initialize max_sum as negative infinity
    current_sum = 0

    for num in nums:
        current_sum += num  # Add the current number to current_sum
        max_sum = max(max_sum, current_sum)  # Update max_sum if needed
        if current_sum < 0:  # Reset current_sum if it goes negative
            current_sum = 0

    return max_sum

# Example Input
nums = [-2, 1, -3, 4, -1, 2, 1, -5, 4]

# Output the maximum subarray sum
print(maxSubArray(nums))
```

```
6
```

## 46. Word Ladder Problem

```
from collections import deque

def word_ladder(beginWord, endWord, wordList):
    wordSet = set(wordList)  # Convert list to a set for quick lookup
    if endWord not in wordSet:
        return 0  # If the endWord is not in the dictionary, return 0

    queue = deque([(beginWord, 1)])  # BFS queue with (word, transformation count)

    while queue:
        word, length = queue.popleft()

        if word == endWord:
            return length  # Return the transformation count when we reach the endWord

        for i in range(len(word)):
            for c in 'abcdefghijklmnopqrstuvwxyz':  # Try replacing each letter
                newWord = word[:i] + c + word[i+1:]

                if newWord in wordSet:
                    queue.append((newWord, length + 1))  # Add new word to the queue
                    wordSet.remove(newWord)  # Remove from set to prevent revisiting

    return 0  # No transformation found

# Example Input
beginWord = "hit"
endWord = "cog"
wordList = ["hot", "dot", "dog", "lot", "log", "cog"]

# Output the shortest transformation sequence length
print(word_ladder(beginWord, endWord, wordList))
```

```
5
```

6. Command-Line RPG Game

Generate randomly select 5 items from a list Close

```
import json
import random

class Character:
    def __init__(self, name, health, attack, defense):
        self.name = name
        self.health = health
        self.attack = attack
        self.defense = defense
        self.inventory = []

    def take_damage(self, damage):
        actual_damage = max(damage - self.defense, 0)
        self.health -= actual_damage
        print(f"{self.name} took {actual_damage} damage. Health: {self.health}")

    def is_alive(self):
        return self.health > 0

    def add_item(self, item):
        self.inventory.append(item)
        print(f"{item} added to inventory.")

class Enemy(Character):
    pass

class Game:
    def __init__(self):
        self.player = None
        self.locations = {
            "forest": "A dark and mysterious forest.",
            "cave": "A deep cave with strange noises.",
            "village": "A peaceful village with shops.",
        }
        self.current_location = "village"

    def start_game(self):
        print("Welcome to the RPG Adventure!")
        name = input("Enter your character's name: ")
        self.player = Character(name, health=100, attack=10, defense=5)
        self.main_menu()

    def main_menu(self):
        while self.player.is_alive():
            print("\nMain Menu:")
            print("1. Explore")
            print("2. Check Inventory")
            print("3. Save Game")
            print("4. Load Game")
            print("5. Exit")
            choice = input("Choose an action: ")

            if choice == "1":
                self.explore()
                self.explore()
            elif choice == "2":
                self.check_inventory()
            elif choice == "3":
                self.save_game()
            elif choice == "4":
                self.load_game()
            elif choice == "5":
                print("Goodbye!")
                break
            else:
                print("Invalid choice. Try again.")

    def explore(self):
        print("\nWhere would you like to go?")
        for loc in self.locations:
            print(f"- {loc}: {self.locations[loc]}")
        choice = input("Enter location: ").lower()

        if choice in self.locations:
            self.current_location = choice
            print(f"\nYou arrive at the {choice}.")
            self.trigger_event()
        else:
            print("Invalid location.")

    def trigger_event(self):
        event = random.choice(["enemy", "item", "nothing"])
        if event == "enemy":
            enemy = Enemy("Goblin", health=30, attack=5, defense=2)
            print(f"A wild {enemy.name} appears!")
            self.fight(enemy)
        elif event == "item":
            item = random.choice(["Sword", "Shield", "Potion"])
            print(f"You found a {item}!")
            self.player.add_item(item)
        else:
            print("Nothing happened.")

    def fight(self, enemy):
        while self.player.is_alive() and enemy.is_alive():
            print("\n1. Attack")
            print("2. Run")
            choice = input("Choose an action: ")

            if choice == "1":
                damage = random.randint(5, self.player.attack)
                enemy.take_damage(damage)

                if enemy.is_alive():
                    enemy_damage = random.randint(2, enemy.attack)
                    self.player.take_damage(enemy_damage)
            elif choice == "2":
                print("You ran away!")
                return
            else:
                print("Invalid choice.")

        if self.player.is_alive():
            print(f"You defeated the {enemy.name}!")
        else:
            print("You have been defeated.")

    def check_inventory(self):
        print("\nInventory:", self.player.inventory if self.player.inventory else "Empty")

    def save_game(self):
        data = {
            "name": self.player.name,
            "health": self.player.health,
            "attack": self.player.attack,
            "defense": self.player.defense,
            "inventory": self.player.inventory,
            "location": self.current_location
        }
        with open("savegame.json", "w") as file:
            json.dump(data, file)
        print("Game saved successfully!")

    def load_game(self):
        try:
            with open("savegame.json", "r") as file:
                data = json.load(file)
                self.player = Character(data["name"], data["health"], data["attack"], data["defense"])
                self.player.inventory = data["inventory"]
                self.current_location = data["location"]
            print("Game loaded successfully!")
        except FileNotFoundError:
            print("No saved game found.")

# Start the game
game = Game()
game.start_game()
```

```
Welcome to the RPG Adventure!
Enter your character's name:
```