TESTING / Python / pytoning coding questions.ipynb

Udaykumaryellaboina 14022025 f60700e · now

771 lines (771 loc) · 17.2 KB

Preview Code Blame Raw

Q1. Reverse a String Problem: Reverse a given string.

Input: "hello"

Output: "olleh"

Solution 1: Using Slicing

In [1]:
```
def reverse_string(s):
    return s[::-1]

# Test
print(reverse_string("hello"))  # Output: olleh
```

```
olleh
```

In [2]:
```
#Solution 2: Using Loop

def reverse_string(s):
    reversed_str = ""
    for char in s:
        reversed_str = char + reversed_str
    return reversed_str

# Test
print(reverse_string("hello"))  # Output: olleh
```

```
olleh
```

Check if a String is a Palindrome

Problem: Check if a string is the same forward and backward.

Input: "madam"

Output: True

Solution:

In [3]:
```
def is_palindrome(s):
    s = s.lower()  # Case insensitive
    return s == s[::-1]

# Test
print(is_palindrome("Madam"))  # Output: True
print(is_palindrome("Hello"))  # Output: False
```

```
True
False
```

Find the First Non-Repeating Character

Problem: Find the first character that does not repeat.

Input: "swiss"

Output: "w"

Solution:

In [8]:
```
from collections import Counter

def first_non_repeating(s):
    count = Counter(s)
    for char in s:
        if count[char] == 1:
            return char
    return None

# Test
print(first_non_repeating("swiss"))  # Output: w
print(first_non_repeating("aabbcc")) # Output: None
```

```
w
None
```

Count the Occurrences of Each Character

Problem: Count the number of occurrences of each character in a string.

Input: "apple"

Output: {'a': 1, 'p': 2, 'l': 1, 'e': 1}

Solution:

In [9]:
```
from collections import Counter

def char_count(s):
    return dict(Counter(s))

# Test
print(char_count("apple"))  # Output: {'a': 1, 'p': 2, 'l': 1, 'e': 1}
```

```
{'a': 1, 'p': 2, 'l': 1, 'e': 1}
```

Check if Two Strings are Anagrams

Problem: Check if two strings are anagrams of each other.

Input: "listen", "silent"

Output: True

In [11]:
```
def are_anagrams(s1, s2):
    return sorted(s1) == sorted(s2)

# Test
print(are_anagrams("listen", "silent"))  # Output: True
print(are_anagrams("hello", "world"))    # Output: False
```

```
True
False
```

Find the Maximum Product of Two Integers

Problem: Given an array of integers, find the maximum product of two integers.

Input: [1, 20, -1, 3, 10]

Output: 20

In [12]:
```
def max_product(nums):
    nums.sort()
    return max(nums[-1] * nums[-2], nums[0] * nums[1])

# Test
print(max_product([1, 20, -1, 3, 10]))  # Output: 200
```

```
200
```

Find the Second Largest Element

Problem: Find the second largest element in an array.

Input: [10, 20, 4, 45, 99]

Output: 45

In [14]:
```
def second_largest(nums):
    first, second = float('-inf'), float('-inf')
    for num in nums:
        if num > first:
            second = first
            first = num
        elif num > second and num != first:
            second = num
    return second

# Test
print(second_largest([10, 20, 4, 45, 99]))  # Output: 45
```

```
45
```

Find Duplicates in an Array

Problem: Find all duplicates in an array.

Input: [1, 2, 3, 2, 4, 3, 5]

Output: [2, 3]

In [15]:
```
def find_duplicates(nums):
    seen = set()
    duplicates = set()
    for num in nums:
        if num in seen:
            duplicates.add(num)
        else:
            seen.add(num)
    return list(duplicates)

# Test
print(find_duplicates([1, 2, 3, 2, 4, 3, 5]))  # Output: [2, 3]
```

```
[2, 3]
```

Rotate Array by K Positions

Problem: Rotate an array to the right by k positions.

Input: [1, 2, 3, 4, 5], k=2

Output: [4, 5, 1, 2, 3]

In [16]:
```python
def rotate_array(nums, k):
    k = k % len(nums)  # In case k > len(nums)
    return nums[-k:] + nums[:-k]

# Test
print(rotate_array([1, 2, 3, 4, 5], 2))  # Output: [4, 5, 1, 2, 3]
```

```
[4, 5, 1, 2, 3]
```

FizzBuzz Problem: Print numbers from 1 to 100. If divisible by 3, print "Fizz". If divisible by 5, print "Buzz". If divisible by both, print "FizzBuzz".

In [18]:
```python
def fizz_buzz():
    for i in range(1, 10):
        if i % 3 == 0 and i % 5 == 0:
            print("FizzBuzz")
        elif i % 3 == 0:
            print("Fizz")
        elif i % 5 == 0:
            print("Buzz")
        else:
            print(i)

# Test
fizz_buzz()
```

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
```

Fibonacci Sequence

Problem: Print the Fibonacci sequence up to n numbers.

Input: n = 10

Output: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34

In [19]:
```python
def fibonacci(n):
    a, b = 0, 1
    for _ in range(n):
        print(a, end=" ")
        a, b = b, a + b

# Test
fibonacci(10)
```

```
0 1 1 2 3 5 8 13 21 34
```

Find Missing Number in Array

Problem: Find the missing number in an array of n integers ranging from 1 to n.

Input: [1, 2, 4, 5, 6]

Output: 3

In [20]:
```python
def find_missing_number(nums):
    n = len(nums) + 1
    total_sum = n * (n + 1) // 2
    return total_sum - sum(nums)

# Test
print(find_missing_number([1, 2, 4, 5, 6]))  # Output: 3
```

3

Longest Substring Without Repeating Characters

Problem: Find the length of the longest substring without repeating characters.

Input: "abcabcbb"

Output: 3

In [21]:
```python
def length_of_longest_substring(s):
    char_set = set()
    left = 0
    max_len = 0

    for right in range(len(s)):
        while s[right] in char_set:
            char_set.remove(s[left])
            left += 1
        char_set.add(s[right])
        max_len = max(max_len, right - left + 1)

    return max_len

# Test
print(length_of_longest_substring("abcabcbb"))  # Output: 3
```

3

String Compression

Problem: Given a string, compress it such that "aaabbbcc" becomes "a3b3c2".

If the compressed string is longer than the original string, return the original string.
Input: "aabccccaaa"

Output: "a2b1c5a3"

In [23]:
```python
def compress_string(s):
    result = []
    count = 1
    for i in range(1, len(s)):
        if s[i] == s[i - 1]:
            count += 1
        else:
            result.append(s[i - 1] + str(count))
            count = 1
    result.append(s[-1] + str(count))

    compressed_str = ''.join(result)
    return compressed_str if len(compressed_str) < len(s) else s

# Test
print(compress_string("aabccccaaa"))  # Output: a2b1c5a3
```

```
a2b1c5a3
```

Find the Intersection of Two Arrays

Problem: Find the intersection of two arrays (elements that are common to both arrays).

Input: [1, 2, 2, 1], [2, 2]

Output: [2, 2]

In [24]:
```python
from collections import Counter

def intersect(nums1, nums2):
    counter1 = Counter(nums1)
    counter2 = Counter(nums2)
    intersection = counter1 & counter2
    return list(intersection.elements())

# Test
print(intersect([1, 2, 2, 1], [2, 2]))  # Output: [2, 2]
```

```
[2, 2]
```

Move Zeros to the End

Problem: Move all zero elements in an array to the end while maintaining the order of non-zero elements.

Input: [0, 1, 0, 3, 12]

Output: [1, 3, 12, 0, 0]

In [25]:
```python
def move_zeros_to_end(nums):
    non_zero_count = 0

    # Move all non-zero elements to the front
    for i in range(len(nums)):
        if nums[i] != 0:
            nums[non_zero_count] = nums[i]
            non_zero_count += 1

    # Fill remaining positions with 0s
    for i in range(non_zero_count, len(nums)):
        nums[i] = 0

    return nums

# Test
print(move_zeros_to_end([0, 1, 0, 3, 12]))  # Output: [1, 3, 12, 0, 0]
```

```
[1, 3, 12, 0, 0]
```

Find the Missing Number in an Array of 1 to N

Problem: Given an array of size n containing numbers from 1 to n+1, find the missing number.

Input: [3, 7, 1, 2, 8, 4, 5]

Output: 6

In [26]:

```python
def find_missing_number(nums):
    n = len(nums) + 1  # Expected size
    total_sum = n * (n + 1) // 2
    actual_sum = sum(nums)

    return total_sum - actual_sum

# Test
print(find_missing_number([3, 7, 1, 2, 8, 4, 5]))  # Output: 6
```

```
6
```