Python questions

03 March 2025 00:34

LONGEST COMMON PREFIX

code.py

code.py > longest_common_prefix

```python
def longest_common_prefix(strs):
    if not strs:
        return ""

    prefix = strs[0]  # Assume the first string is the prefix

    for string in strs[1:]:
        while not string.startswith(prefix):
            prefix = prefix[:-1]  # Remove the last character from prefix
            if not prefix:
                return ""  # If prefix becomes empty, return ""

    return prefix


# Test cases
print(longest_common_prefix(["flower", "flow", "flight"]))  # Output: "fl"
print(longest_common_prefix(["dog", "racecar", "car"]))  # Output: ""
print(longest_common_prefix(["interspecies", "interstellar", "interstate"]))  # Output: "inter"
print(longest_common_prefix([""]))  # Output: ""
print(longest_common_prefix(["a"]))  # Output: "a"
print(longest_common_prefix(["abc", "abcde", "abcdef"]))  # Output: "abc"
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
sandy@sandy:~/new$ python3 code.py
fl

inters

a
abc
sandy@sandy:~/new$
```

LARGEST ODD NUMBER

code.py code1.py

code1.py > ...

```python
def largest_odd_number(s):
    for i in range(len(s) - 1, -1, -1):  # Iterate from the last character to the first
        if int(s[i]) % 2 == 1:  # Check if the digit is odd
            return s[:i+1]  # Return the substring up to the last odd digit
    return ""  # Return empty string if no odd digit is found

# Test cases
print(largest_odd_number("51"))  # "51"
print(largest_odd_number("52"))  # "5"
print(largest_odd_number("35428"))  # "354"
print(largest_odd_number("2468"))  # "" (No odd digit)
print(largest_odd_number("13579"))  # "13579" (Already odd)
print(largest_odd_number("8642"))  # "" (No odd digit)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash - n

```
sandy@sandy:~/new$ python3 code1.py
51
5
35

13579

sandy@sandy:~/new$
```

REMOVE OUTERMOST PARENTHESIS

code.py code1.py code2.py

code2.py > ...

```python
def remove_outermost_parenthesis(s):
    # Check if input is valid
    if not s or s[0] != '(' or s[-1] != ')' or s.count('(') != s.count(')'):
        return "Invalid input: Unbalanced or incorrectly formatted parentheses"

    result = []
    count = 0

    for char in s:
        if char == '(' and count > 0:
            result.append(char)
        elif char == ')' and count > 1:
            result.append(char)

        if char == '(':
            count += 1
        elif char == ')':
            count -= 1

    return ''.join(result)

# Test cases
print(remove_outermost_parenthesis("(()())"))  # "()()"
print(remove_outermost_parenthesis("(())"))    # "()"
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
sandy@sandy:~/new$ python3 code2.py
()()
()
sandy@sandy:~/new$
```

ANAGRAMS : An **anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, using all the original letters exactly once, examples like rat <--> art , top <--> pot, etc..

Implementation can be done in several ways

1) Using dictionaries in python

code.py | code1.py | code2.py | code3.py

code3.py > are_anagrams

```python
def are_anagrams(s1, s2):
    """Check if two strings are anagrams using a dictionary."""
    if len(s1) != len(s2):
        return False  # Anagrams must have the same length

    char_count = {}

    # Count characters in the first string
    for char in s1:
        char_count[char] = char_count.get(char, 0) + 1

    # Subtract character counts using the second string
    for char in s2:
        if char not in char_count or char_count[char] == 0:
            return False
        char_count[char] -= 1

    return all(value == 0 for value in char_count.values())

# Test cases
print(are_anagrams("listen", "silent"))  # True
print(are_anagrams("triangle", "integral"))  # True
print(are_anagrams("hello", "world"))  # False
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
sandy@sandy:~/new$ python3 code3.py
True
True
False
sandy@sandy:~/new$
```

2) Using sorted method or counters in collections

code.py | code1.py | code2.py | code3.py

code3.py > ...

```python
print ("----- SORT -----")
def are_anagrams(s1, s2):
    """Check if two strings are anagrams."""
    return sorted(s1) == sorted(s2)

# Test cases
print(are_anagrams("listen", "silent"))  # True
print(are_anagrams("hello", "world"))    # False

print ("\n----- COUNTERS -----")
from collections import Counter

def are_anagrams(s1, s2):
    """Check if two strings are anagrams using a dictionary (Counter)."""
    return Counter(s1) == Counter(s2)

# Test cases
print(are_anagrams("listen", "silent"))  # True
print(are_anagrams("triangle", "integral"))  # True
print(are_anagrams("hello", "world"))  # False
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
sandy@sandy:~/new$ python3 code3.py
----- SORT -----
True
False

----- COUNTERS -----
True
True
False
sandy@sandy:~/new$
```