Daniel Gamboa

danielgamboa10@gmail.com

BASIC INFO

Test: Computer Science - Python II - Module Project

Solved: 6/6

Similarity: none

Score: 1800/1800

Finished On: 06 Apr 2021

Time Taken: 125m/168h

Labels: -

Task	Solve Time	Score	Similarity
csWhereIsBob	3min	300/300	none
csSchoolYearsAndGroups	11min	300/300	none
csMakeItJazzy	7min	300/300	none
csShortestWord	8min	300/300	none
csSumOfPositive	2min	300/300	none
csAnythingButFive	8min	300/300	none

Task details: csWhereIsBob

Description:

Write a function that searches a list of names (unsorted) for the name "Bob" and returns the location in the list. If Bob is not in the array, return -1.

Examples:

- `csWhereIsBob(["Jimmy", "Layla", "Bob"]) → 2`
- `csWhereIsBob(["Bob", "Layla", "Kaitlyn", "Patricia"]) → 0`
- `csWhereIsBob(["Jimmy", "Layla", "James"]) → -1`

Notes:

- Assume all names start with a capital letter and are lowercase thereafter (i.e. don't worry about finding "BOB" or "bob").

Solution (main.py3):

```
def csWhereIsBob(names):
    return names.index("Bob") if "Bob" in names else -1

```

Task details: csSchoolYearsAndGroups

Description:

Imagine a school that children attend for `years`. In each year, there are a certain number of `groups` started, marked with the `letters`. So if `years = 7` and `groups = 4`For the first year, the groups are 1a, 1b, 1c, 1d, and for the last year, the groups are 7a, 7b, 7c, 7d.

Write a function that returns the groups in the school by year (as a string), separated with a comma and space in the form of "1a, 1b, 1c, 1d, 2a, 2b (....) 6d, 7a, 7b, 7c, 7d".

Examples:

- csSchoolYearsAndGroups(years = 7, groups = 4) → "1a, 1b, 1c, 1d, 2a, 2b, 2c, 2d, 3a, 3b, 3c, 3d, 4a, 4b, 4c, 4d, 5a, 5b, 5c, 5d, 6a, 6b, 6c, 6d, 7a, 7b, 7c, 7d"

Notes:

- `1 <= years <= 10`
- `1 <= groups <=26`

Solution (main.py3):

```
def csSchoolYearsAndGroups(years, groups):
    store = []
    letters = list(string.ascii_lowercase)
    for y in range(1, years + 1, 1):
        for l in letters[:groups:1]:
            store.append(str(y) + l)
    return ", ".join(store)


```

Task details: csMakeItJazzy

Description:

Create a function that concatenates the number 7 to the end of every chord in a list. If a chord already ends with a 7, ignore that chord.

Examples:

- `csMakeItJazzy(["G", "F", "C"]) → ["G7", "F7", "C7"]`
- `csMakeItJazzy(["Dm", "G", "E", "A"]) → ["Dm7", "G7", "E7", "A7"]`
- `csMakeItJazzy(["F7", "E7", "A7", "Ab7", "Gm7", "C7"]) → ["F7", "E7", "A7", "Ab7", "Gm7", "C7"]`
- `csMakeItJazzy([]) → []`

Notes:

- Return an empty list if the given list is empty.
- You can expect all the tests to have valid chords.

Solution (main.py3):

```
def csMakeItJazzy(chords):
    store = []
    for i in chords:
        store.append(i) if i[-1] == '7' else store.append(i + "7")

    return store



```

Task details: csShortestWord

Description:

Given a string of words, return the length of the shortest word(s).

Notes:

- The input string will never be empty and you do not need to validate for different data types.

Solution (main.py3):

```
def csShortestWord(input_str):
    return len(min(input_str.split(), key=len))

    # shortest = len(input_str)

    # words = input_str.split()
    # for w in words:
    #     if len(w) < shortest:
    #         shortest = len(w)

    # return shortest
```

Task details: csSumOfPositive

Description:

Given an array of integers, return the sum of all the positive integers in the array.

Examples:

- csSumOfPositive([1, 2, 3, -4, 5]) -> 1 + 2 + 3 + 5 = 11
- csSumOfPositive([-3, -2, -1, 0, 1]) -> 1
- csSumOfPositive([-3, -2]) -> 0

Notes:

- If the `input_arr` does not contain any positive integers, the default sum should be `0`.

Solution (main.py3):

```
def csSumOfPositive(input_arr):
    accum = 0
    for n in input_arr:
        if n > 0: accum += n

    return accum

```

Task details: csAnythingButFive

Description:

Given a start integer and an ending integer (both inclusive), write a function that returns the count (not the sum) of all integers in the range (except integers that contain the digit 5).

Examples:

- csAnythingButFive(1, 5) -> 1, 2, 3, 4, -> 4 (there are 4 integers in the range that do not contain the digit 5)
- csAnythingButFive(1, 9) -> 1, 2, 3, 4, 6, 7, 8, 9 -> 8
- csAnythingButFive(4, 17) -> 4,6,7,8,9,10,11,12,13,14,16,17 -> 12

Notes:

- The output can contain the digit 5.
- The start number will always be less than the end number (both numbers can also be negative).

Solution (main.py3):

```
def csAnythingButFive(start, end):
    return len(list(filter(lambda n : '5' not in str(n), range(start, end + 1))))

    # i = 0
    # for n in range(start, end+1):
    #     if '5' not in str(n):
    #         i += 1

    # return i

```