9/10

Group_08_Exercise_03

December 14, 2020

1 Exercice 1

Write a function called **string_to_array** to split a string and convert it into an array of words. The function should take one str argument and does not return anything, just print the array of words.

For example:

```
"Robin Singh" should return ["Robin", "Singh"]

"I love arrays they are my favorite" should return ["I", "love", "arrays", "they", "are", "my"
```

1.0.1 check your answer with the following strings

```
string_to_array("Robin Singh") should print ["Robin", "Singh"])
string_to_array("CodeWars") should print ["CodeWars"] string_to_array("I love
arrays they are my favorite") should print ["I", "love", "arrays", "they",
"are", "my", "favorite"] string_to_array("1 2 3") should print ["1", "2", "3"]
string_to_array("") should print [""]
```

[8]:
```
# NOTE! "pass" is a keyword for an empty function. It tells the compiler that␣
↪it is not an error if a function is empty
def string_to_array(s):
    if s.split() == []:
        l = []
        l.append(s)
        print(l)
    else:
        s = str(s)
        print(s.split())
```

[9]:
```
string_to_array("Robin Singh")
string_to_array("CodeWars")
string_to_array("I love arrays they are my favorite")
string_to_array("1 2 3")
string_to_array("")
```

```
['Robin', 'Singh']
['CodeWars']
['I', 'love', 'arrays', 'they', 'are', 'my', 'favorite']
```

```
['1', '2', '3']
['']
```

2 Exercice 2:

Usually when you buy something, you're asked whether your credit card number, phone number or answer to your most secret question is still correct. However, since someone could look over your shoulder, you don't want that shown on your screen. Instead, we mask it.

Your task is to write a function called **maskify**, which changes all but the last four characters into '#' and return the value as a string (str).

Output example: "What was the name of your first pet?"

```
maskify("Skippy")                                    == "##ippy"
maskify("Nananananananananananananananananana Batman!") == "####################################ma
```

2.0.1 check your answer with the following strings

```
maskify("4556364607935616") == "############5616"
maskify(     "64607935616") ==      "#######5616"
maskify(               "1") ==                "1"
maskify(                "") ==                 ""
```

[77]:
```
# your code here
def maskify(s):
    s = str(s)
    if len(s) <=4:
        return s
    else:
        return "#" * (len(s) - 4) + s[-4:]
```

[78]:
```
# test your function here
maskify("4556364607935616")
```

[78]:
```
'############5616'
```

[79]:
```
maskify(     "64607935616")
```

[79]:
```
'#######5616'
```

[80]:
```
maskify(               "1")
```

[80]:
```
'1'
```

[81]:
```
maskify(                "")
```

[81]:
```
''
```

3 Exercice 3

Your task is to create a function that does four basic mathematical operations.

The function should take three arguments -

operation(string/char),

value1(number),

value2(number).

The function should return result of numbers after applying the chosen operation.

Examples

```
basic_op('+', 4, 7)          # Output: 11
basic_op('-', 15, 18)        # Output: -3
basic_op('*', 5, 5)          # Output: 25
basic_op('/', 49, 7)         # Output: 7
```

[82]:
```
# your code here

def basic_op(s,a, b):
    if s == '+':
        return a+b
    elif s == '-':
        return a-b
    elif s == '*':
        return a*b
    elif s =='/':
        return a/b
```

[83]:
```
#test
basic_op('+', 4, 7)          # Output: 11
```

[83]: 11

[84]:
```
basic_op('-', 15, 18)        # Output: -3
```

[84]: -3

[85]:
```
basic_op('*', 5, 5)          # Output: 25
```

[85]: 25

[86]:
```
basic_op('/', 49, 7)         # Output: 7
```

[86]: 7.0

4 Exercice 4

Who remembers back to their time in the schoolyard, when you would take a flower and tear its petals, saying each of the following phrases each time a petal was torn:

`I like you a little a lot passionately madly not at all`

When the last petal was torn there were cries of excitement, dreams, surging thoughts and emotions.

Your goal is to determine which phrase you would say for a flower of a given number of petals, where **nb_petals > 0**.

Examples:

```
how_much_i_like_you(7)          # Output: "I like you"
how_much_i_like_you(2)          # Output: "a little"
how_much_i_like_you(5)          # Output: "madly"
how_much_i_like_you(9)          # Output: "a lot"
```

[87]:
```
# your code here
def how_much_i_like_you(n):
    p = ["I like you","a little","a lot","passionately","madly","not at all"]
    if n <= len(p):
        return p[n-1]
    else:
        return p[(n % len(p))-1]
```

[88]:
```
# Test
how_much_i_like_you(7)
```

[88]: 'I like you'

[89]:
```
how_much_i_like_you(2)
```

[89]: 'a little'

[90]:
```
how_much_i_like_you(5)
```

[90]: 'madly'

[91]:
```
how_much_i_like_you(9)
```

[91]: 'a lot'

5 Exercice 5 (slicing) - 1 pt

You are going to be given an array of integers. Your job is to take that array and find an index N where the sum of the integers to the left of N is equal to the sum of the integers to the right of N. If there is no index that would make this happen, return -1.

For example:

Let's say you are given the array {1,2,3,4,3,2,1}: Your function will return the index 3, because at the 3rd position of the array, the sum of left side of the index ({1,2,3}) and the sum of the right side of the index ({3,2,1}) both equal 6.

Let's look at another one. You are given the array {1,100,50,-51,1,1}: Your function will return the index 1, because at the 1st position of the array, the sum of left side of the index ({1}) and the sum of the right side of the index ({50,-51,1,1}) both equal 1.

Last one: You are given the array {20,10,-80,10,10,15,35} At index 0 the left side is {} The right side is {10,-80,10,10,15,35} They both are equal to 0 when added. (Empty arrays are equal to 0 in this problem) Index 0 is the place where the left side and right side are equal.

Note: Please remember that in python the index of an array starts at 0.

Input: An integer array of length $0 < \text{arr} < 1000$. The numbers in the array can be any integer positive or negative.

Output: The lowest index N where the side to the left of N is equal to the side to the right of N. If you do not find an index that fits these rules, then you will return -1.

Note: If you are given an array with multiple answers, return the lowest correct index.

Check your answer with the following arrays

```
find_even_index([1,2,3,4,3,2,1]) should return 3 find_even_index([1,100,50,-51,1,1])
should return 1 find_even_index([1,2,3,4,5,6]) should return
-1 find_even_index([20,10,30,10,10,15,35]) should return 3
find_even_index([20,10,-80,10,10,15,35]) should return 0 find_even_index([10,-80,10,10,15,35,20
should return 6 find_even_index(range(1,100)) should return
-1 find_even_index([-1,-2,-3,-4,-3,-2,-1]) should return 3
find_even_index(range(-100,-1)) should return -1 find_even_index([0,0,0,0,0])
should return 0 because it Should pick the first index if more cases are valid
```

[92]:
```
def find_even_index(l):
    m = l[::-1]
    res = {'index':-1}
    for a,b  in enumerate(m):
        if sum(m[:a+1]) == 0:
            res['index'] = 0
            break
    else:
        for i,j in enumerate(l):

            for x,y in enumerate(m):
                #print(l[:i+1], m[:x+1])
                if sum(l[:i+1]) == sum(m[:x+1]) and (i+1<len(l)-i) and
↪(x+1<len(m)-x):
                    right_rev = m[:x+1]
                    left = l[:i+1]
                    res['index'] = len(left)
```

```
                    break


    if res['index'] ==0:
        return res['index']
    elif res['index'] >=1:
        #print(f"{left} and {right_rev[::-1]} are equal")
        return res['index']
    else:
        return -1
```

[93]:
```
# Test
find_even_index([1,100,50,-51,1,1])
```

[93]: 1

[94]:
```
find_even_index([10,-80,10,10,15,35,20])
```

[94]: -1

[95]:
```
find_even_index([20,10,-80,10,10,15,35])
```

[95]: 0

6 Exercice 6

You are given an input string.

For each symbol in the string if it's the first character occurence, replace it with a '1', else replace it with the amount of times you've already seen it...

Examples:

```
input   =  "Hello, World!"
result  =  "1112111121311"

input   =  "aaaaaaaaaaaa"
result  =  "123456789101112"
```

Test your code with the following strings

`numericals("Hello, World!") should return "1112111121311" numericals("Hello, World! It's me, JomoPipi!") should return "1112111121311211113122413241122" numericals("hello hello") should return "11121122342" numericals("Hello") should return "11121" numericals("aaaaaaaaaaaa") should return "123456789101112"`

[2]:
```
# your code here
def numericals(s):
    res = ""
    for i in range(len(s)):
```

```
        res = res + str(s[:i+1].count(s[i]))
    return res
```

[3]:
```
numericals("Hello, World!")
```

[3]: '1112111121311'

[4]:
```
numericals("Hello, World! It's me, JomoPipi!")
```

[4]: '1112111121311211111312241324111 22'

[5]:
```
numericals("hello hello")
```

[5]: '11121122342'

[6]:
```
numericals("Hello")
```

[6]: '11121'

[7]:
```
numericals("aaaaaaaaaaaa")
```

[7]: '123456789101112'

7 Exercice 7

Write a function called 'solve'. It shoud take two arrays of string and return the number of times each string of the second array appears in the first array.

Example

array1 = ['abc', 'abc', 'xyz', 'cde', 'uvw']

array2 = ['abc', 'cde', 'uap']

How many times do the elements in array2 appear in array1?

'abc' appears twice in the first array (2)

'cde' appears only once (1)

'uap' does not appear in the first array (0)

Therefore, solve(array1, array2) = [2, 1, 0]

[98]:
```
# your code here
array1 = ['abc', 'abc', 'xyz', 'cde', 'uvw']
array2 = ['abc', 'cde', 'uap']

def solve(array1, array2):
    l = []
    for i in array2:
        l.append(array1.count(i))
```

```
    return l

solve(array1, array2)
```

[98]:
```
[2, 1, 0]
```

[99]:
```
# test your function with those values
solve(['abc', 'abc','xyz','abcd','cde'], ['abc', 'cde', 'uap']) # should give␣
↪[2, 1, 0]
```

[99]:
```
[2, 1, 0]
```

[100]:
```
solve(['abc', 'xyz','abc', 'xyz','cde'], ['abc', 'cde', 'xyz']) # should give ␣
↪[2, 1, 2])
```

[100]:
```
[2, 1, 2]
```

[101]:
```
solve(['quick', 'brown', 'fox', 'is', 'quick'], ['quick', 'abc', 'fox']) #␣
↪should give [2, 0, 1])
```

[101]:
```
[2, 0, 1]
```

[ ]:
```

```