

Add-One

May 14, 2020

0.0.1 Problem Statement

You are given a non-negative number in the form of list elements. For example, the number 123 would be provided as `arr = [1, 2, 3]`. Add one to the number and return the output in the form of a new list.

Example 1: *input = [1, 2, 3] *output = [1, 2, 4]

Example 2: *input = [9, 9, 9] *output = [1, 0, 0, 0]

Challenge:

One way to solve this problem is to convert the input array into a number and then add one to it. For example, if we have `input = [1, 2, 3]`, you could solve this problem by creating the number 123 and then separating the digits of the output number 124.

But can you solve it in some other way?

```
In [ ]: def add_one(arr):
        """
        :param: arr - list of digits representing some number x
        return a list with digits representing (x + 1)
        """
        num = 0
        output = []
        reverse = []
        count = len(arr) - 1
        for item in arr:
            num += item * 10 ** count
            count -= 1
        num += 1
        while num > 0:
            digit = int(num % 10)
            num = int(num / 10)
            print(digit, num)
            output.append(digit)
        for item in range(len(output), 0, -1):
            print(item)
            reverse.append(output[item])
            print(reverse)
        print(output)
        pass
add_one([1, 2, 3])
```

Hide Solution

```
In [ ]: # Solution
```

```
'''
```

The Logic

1. The idea is to start checking the array from the right end, in a FOR loop.

2. Add 1 to the digit, and check if it lies in the range 0-9 OR becomes 10.

3. If the updated digit is between 0-9, quit the FOR loop. (Example, original array is [

4. Otherwise update the current position in the array, and carry over the "borrow" to the

5. Once, we finish iterating over all the digits of the original array, we will be left with

6. Return the updated array, but there is a trick which helps us to select the starting index

```
'''
```

```
# Change the arr in-place
```

```
def add_one(arr):
```

```
    borrow = 1; # initial value
```

```
    """
```

The three arguments of range() function are:

starting index, ending index(non-inclusive), and the increment/decrement value

```
    """
```

```
    # Traverse in reverse direction starting from the end of the list
```

```
    # The argument of range() functions are:
```

```
    # starting index, ending index (non exclusive), and the increment/decrement size
```

```
    for i in range(len(arr), 0, -1):
```

```
        # The "digit" denotes the updated Unit, Tens, and then Hundred position iteration
```

```
        digit = borrow + arr[i - 1]
```

```
        '''
```

The "borrow" will be carried to the next left digit

If the digit is between 0-9, borrow will be 0.

If digit is 10, then the borrow will be 1.

```
        '''
```

```
    # The "//" is a floor division operator
```

```
    borrow = digit//10
```

```
    if borrow == 0:
```

```
        # Update the arr[i - 1] with the updated digit, and quit the FOR loop.
```

```
        arr[i - 1] = digit
```

```
        break
```

```
    else:
```

```
        # Update the arr[i - 1] with the remainder of (digit % 10)
```

```
        arr[i - 1] = digit % 10
```

```
    # Prepend the final "borrow" to the original array.
```

```
    arr = [borrow] + arr
```

```

# In this final updated arr, find a position (starting index) from where to return t
# For [0, 1, 2, 4] , the position (starting index) will be 1
# For [1, 0, 0, 0] , the position (starting index) will be 0
position = 0
while arr[position]==0:
    position += 1

return arr[position:]

#-----#
# Descriptive Example 1 - Original array is [1, 2, 3]
#-----#

'''
FOR LOOP BEGINS
    For i=3 , arr[2]=3 , digit=4 , borrow=0
    BORROW COMPARISON START
        Since borrow is 0, update arr[2] = digit = 4 and quit the FOR loop.
        NO need to check other digits on the left of current digit
FOR LOOP ENDS

Append [0] to the beginning of the original arr. Now arr = [0, 1, 2, 4]
In this final updated arr, find a position from where to return the list. This position
Return [1, 2, 4]

'''
#-----#
# Descriptive Example 2 - Original array is [9, 9, 9]
#-----#
'''
FOR LOOP BEGINS
    For i= 3 , arr[ 2 ] = 9 , digit = 10 , borrow = 1
    BORROW COMPARISON START
        Since borrow is non-zero, update arr[ 2 ] = digit % 10 = 0
        Update output = borrow = 1
    BORROW COMPARISON ENDS

    For i= 2 , arr[ 1 ] = 9 , digit = 10 , borrow = 1
    BORROW COMPARISON START
        Since borrow is non-zero, update arr[ 1 ] = digit % 10 = 0
        Update output = borrow = 1
    BORROW COMPARISON ENDS

    For i= 1 , arr[ 0 ] = 9 , digit = 10 , borrow = 1
    BORROW COMPARISON START
        Since borrow is non-zero, update arr[ 0 ] = digit % 10 = 0

```

```
        Update output = borrow = 1
    BORROW COMPARISON ENDS
```

```
FOR LOOP ENDS
```

```
Append [1] to the beginning of the original arr. Now arr = [1, 0, 0, 0]
```

```
In this final updated arr, find a position from where to return the list. This position
Return [1, 0, 0, 0]
'''
```

```
In [ ]: # A helper function for Test Cases
def test_function(test_case):
    arr = test_case[0]
    solution = test_case[1]

    output = add_one(arr)
    for index, element in enumerate(output):
        if element != solution[index]:
            print("Fail")
            return
    print("Pass")
```

```
In [ ]: # Test Case 1
arr = [0]
solution = [1]
test_case = [arr, solution]
test_function(test_case)
```

```
In [ ]: # Test Case 2
arr = [1, 2, 3]
solution = [1, 2, 4]
test_case = [arr, solution]
test_function(test_case)
```

```
In [ ]: # Test Case 3
arr = [9, 9, 9]
solution = [1, 0, 0, 0]
test_case = [arr, solution]
test_function(test_case)
```