Problem Solving Template (change the title)

To learn how to use this template, check out the course "Data Structures and Algorithms in Python".

How to run the code and save your work

The recommended way to run this notebook is to click the "Run" button at the top of this page, and select "Run on Binder". This will run the notebook on mybinder.org, a free online service for running Jupyter notebooks.

This tutorial is an executable Jupyter notebook. You can *run* this tutorial and experiment with the code examples in a couple of ways: *using free online resources* (recommended) or *on your computer*.

Option 1: Running using free online resources (1-click, recommended)

The easiest way to start executing the code is to click the **Run** button at the top of this page and select **Run on Binder**. You can also select "Run on Colab" or "Run on Kaggle", but you'll need to create an account on Google Colab or Kaggle to use these platforms.

Option 2: Running on your computer locally

To run the code on your computer locally, you'll need to set up Python, download the notebook and install the required libraries. We recommend using the Conda distribution of Python. Click the **Run** button at the top of this page, select the **Run Locally** option, and follow the instructions.

Saving your work

Before staring the assignment, let's save a snapshot of the assignment to your Jovian profile, so that you can access it later, and continue your work.

```
project_name = 'finding-difference' # give it an appropriate name
```

```
!pip install jovian --upgrade --quiet
```

```
import jovian
```

```
jovian.commit(project=project_name)
```

```
[jovian] Attempting to save notebook..
[jovian] Updating notebook "aakashns/python-problem-solving-template" on https://jovian.ai/
[jovian] Uploading notebook..
[jovian] Capturing environment..
[jovian] Committed successfully! https://jovian.ai/aakashns/python-problem-solving-template
```

```
'https://jovian.ai/aakashns/python-problem-solving-template'
```

Problem Statement

> find the difference between the strings assigned to the variables s and t.here,only one letter will be extra in the t variable and find it. (past the problem statement here)(past the problem statement here)

Source: leetcode (question:389)(link to the source)

The Method

Here's the systematic strategy we'll apply for solving problems:

1. State the problem clearly. Identify the input & output formats.
2. Come up with some example inputs & outputs. Try to cover all edge cases.
3. Come up with a correct solution for the problem. State it in plain English.
4. Implement the solution and test it using example inputs. Fix bugs, if any.
5. Analyze the algorithm's complexity and identify inefficiencies, if any.
6. Apply the right technique to overcome the inefficiency. Repeat steps 3 to 6.

This approach is explained in detail in Lesson 1 of the course. Let's apply this approach step-by-step.

Solution

1. State the problem clearly. Identify the input & output formats.

While this problem is stated clearly enough, it's always useful to try and express in your own words, in a way that makes it most clear for you.

Problem

> ***1.as per the problem statement,we have to identify the extra element or letter in the string 't'(assigned variable).**
> **2.only one element will be extra and the variable 's' should be compare with the variable 't'.**
> **3.by using the methods such as ord function ,hash table and iteration the extra element should be find. * (express the problem clearly in our own words, and in absract terms** (express the problem clearly in our own words, and in absract terms

Input

1. **s = "sdfg"**
2. **t = "sdfgh"**

(add more if required)

Output

1. **'h'**

(add more if required)

Based on the above, we can now create a signature of our function:

```
def findTheDifference(s: str, t: str):
    pass   # Create a function signature here. The body of the function can contain a s
```

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Attempting to save notebook..
```

2. Come up with some example inputs & outputs. Try to cover all edge cases.

Our function should be able to handle any set of valid inputs we pass into it. Here's a list of some possible variations we might encounter:

1. **in both the variables by comparing s and t ,having and extra element in t at the corner.**
2. **in variable 's' contains no elements ,where as in variable 't' contains one element.**
3. **in both the variables by comparing s and t ,having and extra element in t in the middle**
4. **???**
5. **???**

(add more if required)

We'll express our test cases as dictionaries, to test them easily. Each dictionary will contain 2 keys: `input` (a dictionary itself containing one key for each argument to the function and `output` (the expected result from the function).

```
test = {
    'input': { "s":"sdfg","t":"sdfgh"

    },
    'output': 'h'
}
```

Create one test case for each of the scenarios listed above. We'll store our test cases in an array called `tests` .

```
tests = []
```

```
tests.append(test)
```

```
tests.append({
    'input': {
        "s":" ","t":"r"
    },
```

```
    'output': "r"
})
```

```
tests.append({
    'input': {
        "s":" santhu","t":"santyhu"
    },
    'output': "y"
})
```

```
tests
```

```
[{'input': {'s': ' ', 't': 'r'}, 'output': 'r'},
 {'input': {'s': ' santhu', 't': 'santyhu'}, 'output': 'y'},
 {'input': {'s': ' santhu', 't': 'santyhu'}, 'output': 'y'}]
```

```
# add more test cases
```

```

```

3. Come up with a correct solution for the problem. State it in plain English.

Our first goal should always be to come up with a *correct* solution to the problem, which may not necessarily be the most *efficient* solution. Come with a correct solution and explain it in simple words below:

1. **here,using hashtable the strings are stored in the dictionaries**
2. **in next step,enumerate the string of assigned variables s by iterating one by one**
3. **now,enumerate the string of assigned variables t by iterating one by one**
4. ** the elements and keys s and t which are assigned to the strings should found equal**
5. **otherwise the extra element will be returned**

(add more steps if renow,enumerate the string of assigned variables t by iterating one by onequired)

Let's save and upload our work before continuing.

```
jovian.commit()
```

```
[jovian] Attempting to save notebook..
[jovian] Updating notebook "aakashns/python-problem-solving-template" on
https://jovian.ai/
[jovian] Uploading notebook..
[jovian] Capturing environment..
[jovian] Committed successfully! https://jovian.ai/aakashns/python-problem-solving-
template
```

```
'https://jovian.ai/aakashns/python-problem-solving-template'
```

4. Implement the solution and test it using example inputs. Fix bugs, if any.

```
def findTheDifference(s: str, t: str):
        hashMap1 = dict()
        hashMap2 = dict()
        for ndx, val in enumerate(s):
            hashMap1[val] = hashMap1.get(val, 0) + 1

        for ndx, val in enumerate(t):
            hashMap2[val] = hashMap2.get(val, 0) + 1

        for key, val in hashMap1.items() if len(hashMap1) > len(hashMap2) else hashMap2
            if hashMap1.get(key) != hashMap2.get(key):
                return key
```

We can test the function by passing the input to it directly or by using the `evaluate_test_case` function from `jovian` .

```
from jovian.pythondsa import evaluate_test_case
```

```
evaluate_test_case(findTheDifference,test)
```

```
Input:
{'s': 'sdfg', 't': 'sdfgh'}

Expected Output:
h


Actual Output:
h

Execution Time:
0.012 ms

Test Result:
```

```
PASSED
```

```
('h', True, 0.012)
```

Evaluate your function against all the test cases together using the `evaluate_test_cases` (plural) function from `jovian` .

```
from jovian.pythondsa import evaluate_test_cases
```

```
evaluate_test_cases(findTheDifference,tests)
```

```
TEST CASE #0

Input:
{'s': ' ', 't': 'r'}

Expected Output:
r


Actual Output:
r

Execution Time:
0.007 ms

Test Result:
PASSED


TEST CASE #1

Input:
{'s': ' santhu', 't': 'santyhu'}

Expected Output:
y


Actual Output:
y

Execution Time:
```

```
0.01 ms

Test Result:
PASSED


TEST CASE #2

Input:
{'s': ' santhu', 't': 'santyhu'}

Expected Output:
y


Actual Output:
y

Execution Time:
0.009 ms

Test Result:
PASSED


SUMMARY

TOTAL: 3, PASSED: 3, FAILED: 0
[('r', True, 0.007), ('y', True, 0.01), ('y', True, 0.009)]
```

Verify that all the test cases were evaluated. We expect them all to fail, since we haven't implemented the function yet.

Let's save our work before continuing.

```
jovian.commit()
```

```
[jovian] Attempting to save notebook..
[jovian] Updating notebook "aakashns/python-problem-solving-template" on https://jovian.ai/
[jovian] Uploading notebook..
[jovian] Capturing environment..
[jovian] Committed successfully! https://jovian.ai/aakashns/python-problem-solving-template
```

```
'https://jovian.ai/aakashns/python-problem-solving-template'
```

5. Analyze the algorithm's complexity and identify inefficiencies, if any.

Time complexity: O(n) explaination: here,n is the length of the longer string between s and t. because, the code iterates through both strings once to count the frequency of characters and store them in dictionaries

Space compexity: O(n) explaination: here,n is the length of the longer string between s and t.because, the code creates two dictionaries, hashMap1 and hashMap2, to store the frequency of characters in s and t. The space required by these dictionaries is proportional to the number of unique characters in s and t, which can be at most equal to n.

```
jovian.commit()
```

```
[jovian] Attempting to save notebook..
[jovian] Updating notebook "aakashns/python-problem-solving-template" on https://jovian.ai/
[jovian] Uploading notebook..
[jovian] Capturing environment..
[jovian] Committed successfully! https://jovian.ai/aakashns/python-problem-solving-template
```

```
'https://jovian.ai/aakashns/python-problem-solving-template'
```

6. Apply the right technique to overcome the inefficiency. Repeat steps 3 to 6.

the code may have some inefficiencies due to enumerates and dictionaries comparision
1.enumerate() is used to get the index and value of each character in the strings s and t. However, the index is not used in the code, so using enumerate() is unnecessary and adds some overhead.
2.len(hashMap1) > len(hashMap2) uses ,as a condition to determine which dictionary to iterate over in the final loop. even though, this comparison is redundant because both dictionaries contain the same characters as keys, so the length of the dictionaries will always be the same.

```
jovian.commit()
```

```
[jovian] Attempting to save notebook..
[jovian] Updating notebook "aakashns/python-problem-solving-template" on
```

```
https://jovian.ai/
[jovian] Uploading notebook..
[jovian] Capturing environment..
[jovian] Committed successfully! https://jovian.ai/aakashns/python-problem-solving-template
```

'https://jovian.ai/aakashns/python-problem-solving-template'

7. Come up with a correct solution for the problem. State it in plain English.

Come with the optimized correct solution and explain it in simple words below:

1. **in this a common dictionary is provided to both the variables s and t.**
2. **initially,using for loop and if and else statements ,the charaters will be added **
3. **the characters will be added one by one into the dictionary**
4. **same as previous loop the characters should be their in variable t by not equal to 0 and no extra element**
5. **if an extra element presents,then it returns.**

(add more steps if required)

Let's save and upload our work before continuing.

```
jovian.commit()
```

```
[jovian] Attempting to save notebook..
[jovian] Updating notebook "aakashns/python-problem-solving-template" on https://jovian.ai/
[jovian] Uploading notebook..
[jovian] Capturing environment..
[jovian] Committed successfully! https://jovian.ai/aakashns/python-problem-solving-template
```

'https://jovian.ai/aakashns/python-problem-solving-template'

8. Implement the solution and test it using example inputs. Fix bugs, if any.

```
def find_Difference(s: str, t: str):
    freq = {}
    for char in s:
        if char not in freq:
            freq[char] = 1
```

```
        else:
            freq[char] += 1

    for char in t:
        if char not in freq or freq[char] == 0:
            return char
        else:
            freq[char] -= 1
```

```
evaluate_test_case(find_Difference,test)
```

```
Input:
{'s': 'sdfg', 't': 'sdfgh'}

Expected Output:
h



Actual Output:
h

Execution Time:
0.006 ms

Test Result:
PASSED


('h', True, 0.006)
```

```
evaluate_test_cases(find_Difference,tests)
```

```
TEST CASE #0

Input:
{'s': ' ', 't': 'r'}

Expected Output:
r



Actual Output:
r
```

Execution Time:
0.005 ms

Test Result:
PASSED

TEST CASE #1

Input:
{'s': ' santhu', 't': 'santyhu'}

Expected Output:
y

Actual Output:
y

Execution Time:
0.007 ms

Test Result:
PASSED

TEST CASE #2

Input:
{'s': ' santhu', 't': 'santyhu'}

Expected Output:
y

Actual Output:
y

Execution Time:
0.005 ms

Test Result:
PASSED

```
SUMMARY

TOTAL: 3, PASSED: 3, FAILED: 0

[('r', True, 0.005), ('y', True, 0.007), ('y', True, 0.005)]
```

9. Analyze the algorithm's complexity and identify inefficiencies, if any.\

Time complexity: The time complexity of this code is O(n). Here, n is the length of the longer string between s and t. The code iterates through both strings once to count the frequency of characters and store them in the freq dictionary.

Space complexity: The space complexity of this code is O(n). Here n is the length of the longer string between s and t. The code creates a dictionary freq to store the frequency of characters in s. The space required by this dictionary is proportional to the number of unique characters in s, which can be at most equal to n.

Inefficiencies: the code is much more efficient compared to the previous implementation. It removes the unnecessary use of enumerate() and the redundant length comparison. The code also simplifies the logic by directly checking if a character exists in the freq dictionary and if its count is zero. If it meets these conditions, it returns the character as the added character.

If you found the problem on an external platform, you can make a submission to test your solution.

Share your approach and start a discussion on the Jovian forum: https://jovian.ai/forum/c/data-structures-and-algorithms-in-python/78

```
jovian.commit()
```

```
[jovian] Attempting to save notebook..
```