[CS 11 25.1] Lab 4a – Topographical Error

Cheatsheet is available here: https://oj.dcs.upd.edu.ph/cs11cheatsheet/

Problem Statement

Your company is developping a kind of text editor, and you have been asigned to impliment a feature that auto-corrects simple typographical erors. As part off this feature, yu need to perform the fallowing function: Given a word, output the set of all the words that are "one mistake" away from it. We define one mistake as any one of the following operations on a word:

- Removing a letter.
- Inserting a letter (anywhere).
- Replacing any of its existing letters.

Clarification: You can "replace" a letter with the same letter.

Task Details

Your task is to implement a function called `word_mutations`. The function takes a single argument, a `str`.

The function must return a `set` of `str`s denoting the set of all words (consisting of lowercase letters) that are one mistake away from it.

Restrictions

Note that many names are banned. Here are a few of them: `zip`, `input`, `type`, `reversed`. This is not an exhaustive list. (If you accidentally use a variable name that turns out to be banned, please rename it.)

The following names are now allowed: `range`, `list`, `enumerate`, `print`, `append`, `pop`, `extend`, `remove`, `sort`, `insert`, `clear`, `reverse`, `reversed`.

For this problem:

- Loops and lists are allowed.
- Up to 6 function definitions are allowed.
- Recursion is **disallowed**. (The recursion limit has been greatly reduced.)
- Sets and dictionaries are allowed.
- Generators and comprehensions are allowed.
- The source code limit is $2{,}000$.

Example Calls

Example 1 Function Call

```
word_mutations('off')
```

Copy

Example 1 Return Value

```
{
    'aff', 'aoff', 'bff', 'boff', 'cff', 'coff', 'dff', 'doff',
'eff', 'eoff', 'ff', 'fff', 'foff',
    'gff', 'goff', 'hff', 'hoff', 'iff', 'ioff', 'jff', 'joff',
'kff', 'koff', 'lff', 'loff', 'mff',
    'moff', 'nff', 'noff', 'oaf', 'oaff', 'obf', 'obff', 'ocf',
'ocff', 'odf', 'odff', 'oef', 'oeff',
    'of', 'ofa', 'ofaf', 'ofb', 'ofbf', 'ofc', 'ofcf', 'ofd',
'ofdf', 'ofe', 'ofef', 'off', 'offa',
    'offb', 'offc', 'offd', 'offe', 'offf', 'offg', 'offh',
'offi', 'offj', 'offk', 'offl', 'offm',
    'offn', 'offo', 'offp', 'offq', 'offr', 'offs', 'offt',
'offu', 'offv', 'offw', 'offx', 'offy',
    'offz', 'ofg', 'ofgf', 'ofh', 'ofhf', 'ofi', 'ofif', 'ofj',
'ofjf', 'ofk', 'ofkf', 'ofl', 'oflf',
    'ofm', 'ofmf', 'ofn', 'ofnf', 'ofo', 'ofof', 'ofp', 'ofpf',
'ofq', 'ofqf', 'ofr', 'ofrf', 'ofs',
    'ofsf', 'oft', 'oftf', 'ofu', 'ofuf', 'ofv', 'ofvf', 'ofw',
'ofwf', 'ofx', 'ofxf', 'ofy', 'ofyf',
    'ofz', 'ofzf', 'ogf', 'ogff', 'ohf', 'ohff', 'oif', 'oiff',
'ojf', 'ojff', 'okf', 'okff', 'olf',
    'olff', 'omf', 'omff', 'onf', 'onff', 'oof', 'ooff', 'opf',
'opff', 'oqf', 'oqff', 'orf', 'orff',
    'osf', 'osff', 'otf', 'otff', 'ouf', 'ouff', 'ovf', 'ovff',
'owf', 'owff', 'oxf', 'oxff', 'oyf',
    'oyff', 'ozf', 'ozff', 'pff', 'poff', 'qff', 'qoff', 'rff',
'roff', 'sff', 'soff', 'tff', 'toff',
    'uff', 'uoff', 'vff', 'voff', 'wff', 'woff', 'xff', 'xoff',
'yff', 'yoff', 'zff', 'zoff',
}
```

Copy

Constraints

- The function `word_mutations` will be called at most 400 times.
- The total length of `word` across all inputs will be will be at most 400.
- `word` is a string of at most 400 lowercase letters.

Scoring

Note: New tests may be added and all submissions may be rejudged at a later time. (All future tests will satisfy the constraints.)

- You get 150 ● points if you solve all test cases where:
  - `word` is a string of at most 50 lowercase letters.
- You get 25 ● points if you solve all test cases.

Clarifications

Report an issue

No clarifications have been made at this time.

Submit solution

[CS 11 25.1] Lab Exercise 4

- ✓ **Points:** 175 (partial)
- **Time limit:** 4.0s
- **Memory limit:** 1G

Problem type

Allowed languages
py3