[CS 11 25.1] Mock HOPE 2c – Philip and Pila

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

Submit solution [CS 11 25.1] Mock HOPE 2

My submissions

Points: 130 (partial)
Time limit: 4.0s
Memory limit: 1G

Problem type

Allowed languages
py3

Problem Statement

Philip is a bank teller. He needs to process a queue of people on a first-come, first-served basis.

There are 26 types of transactions that Philip can handle, so we can conveniently represent each transaction by a distinct letter. Each transaction is a two-step process—the first time a person talks to Philip, they are handed a form they must fill out. After filling out the form, the person will enter the queue again from the back to wait for their turn again. Once their turn comes, they submit the form and their transaction is finished.

The current state of the queue is represented by a string of letters, each of which represents a person with a particular transaction (corresponding to the letter). An uppercase letter means that they haven't done the first step yet, and a lowercase letter means that they have just finished their first step (and have filled out their form).

Assume that there are no more new people going into the queue, and that each person fills out their form in negligible time.

Task Details

Your task is to implement a function called `process_order`. It has a single argument, an *iterable* of characters representing the queue from front to back. (A character is a `str` of length `1`)

The function must return a *generator* that generates `str`s. The `str`s must be the transactions processed by Philip in order, with uppercase meaning the first step, and lowercase meaning the second step.

Note that your generator must be **as lazy as possible**. It should yield each resulting next element as soon as it has enough information, and it should produce these results while advancing the input generators for as little as possible.

Restrictions

Note that some names are banned.

For this problem:

- Loops and lists are allowed.
- Sets and dictionaries are allowed.
- Generators and comprehensions are allowed.
- Recursion is **disallowed**. (The recursion limit has been greatly reduced.)
- Up to 3 function definitions are allowed.
- The source code limit is 800.

Example Calls

Example 1 Function Call

```
[*process_order("PBb")]
```

Copy

Example 1 Return Value

```
['P', 'B', 'b', 'p', 'b']
```

Copy

Example 2 Function Call

```
''.join(process_order(iter("HOrsES")))
```

Copy

Example 2 Return Value

```
"HOrsEShoes"
```

Copy

Example 3 Function Call

```
''.join(process_order(iter("AuThEntic")))
```

Copy

Example 3 Return Value

```
"AuThEnticate"
```

Copy

Example 3 Function Call

```
[*process_order("")]
```

Copy

Example 3 Return Value

```
[]
```

Copy

Constraints

- The function `process_order` will be called at most 200 times.
- At most 500 elements will be consumed from the returned generator.

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 130 🔴 points if you solve all test cases.

Clarifications

Report an issue

No clarifications have been made at this time.