CS2020
Introduction to Programming
Practice Exercise Set 6

List Comprehensions and map/filter/reduce

In this set, you will rewrite many of the previous exercises using list comprehensions and the map/filter/reduce functions. The functions map and filter are built-in while the function reduce is from the functools module.

You may use list methods such as find, index, and count and built-in functions such as max, min, and sum, unless indicated otherwise. Also, unless explicitly requested, you are not required to define a function. For each question, it should take less than 15 minutes to derive a solution using one technique.

Practice Topic

- Processing lists using map/filter/reduce
- Processing lists using list comprehensions

Prerequisite

You need to possess knowledge on

- Basic operations such as assignments and expressions
- Input and output (print)
- Data types int, float, and str.
- Simple selection control using if-else
- Basic looping (Practice Exercise Set 1)
- Defining and using functions (Practice Exercise Set 2)
- String processing (Practice Exercise Set 4)
- List processing (Practice Exercises Sets 3 and 5)

Example 1

Given a list of floats numbers, find out how many are less than or equal to the average of all elements.

a. map / filter / reduce

```
avg = sum(numbers) / len(numbers)
cnt = len(list(filter(lambda x: x <= avg, numbers)))
```

b. list comprehension

```
avg = sum(numbers) / len(numbers)
cnt = len([x for x in numbers if x <= avg])
```

Example 2

Write a function partition that accepts a list of floats numbers and returns two lists. The first list contain those elements that are less than or equal to the average and the second list those that are greater than the average.

a. map / filter / reduce

```
def partition(numbers):
    avg = sum(numbers) / len(numbers)
    lower = list(filter(lambda x: x <= avg, numbers))
    upper = list(filter(lambda x: x > avg, numbers))
    return lower, upper
```

b. list comprehension

```
def partition(numbers):
    avg = sum(numbers) / len(numbers)
    lower = [x for x in numbers if x <= avg]
    upper = [x for x in numbers if x > avg]
    return lower, upper
```

Example 3

Find the maximum number in a list of integers L using the reduce function.

```
from functools import reduce
mx = reduce(lambda m, e: e if e > m else m, L)
```

Exercises

For all exercises, provide two versions of a solution using map/filter/reduce and list comprehension.

1. Write a function get_words_of_length that accepts a list of strings words and an integer N. Return a list of words with length less than or equal to N.
2. Write a function get_average_length that accepts a list of strings words and returns the average length of the strings in the list.
3. Write a function generate_ints that accepts two integers low and high and returns a list of all integers between low and high, inclusive. Assume low is less than or equal to high.

4. Given a list of strings words, determine how many strings in the list have length 5 or less.
5. Given a list of strings words, determine the average length of the strings in the list.
6. Write a function concat that accepts a list of strings words and returns a single string by concatenating strings in the words list separated by one blank space.
7. Write a function has_upper that accepts a list of strings words and returns True if any string in the list contains an uppercase Return False otherwise.
8. Write a function replace_lower that accepts a string S and returns a new string with all lowercase alphabets in S replaced by 'X'. Do not use the replace method here.
9. Write a function extract_negatives that accepts an n-by-m matrix M and returns a list of negative numbers found in M.
10. Given a roster list, find the names of the youngest students. There could be more than one student. Store their names in a list. The roster is organized as a list of student records. A student record is a list of his/her id (string), name (string), class (int), age (int) and a list of test scores (float). The class is a single digit number to designate the student's class level: 0 - Freshman, 1 - Sophomore, 2 - Junior, and 3 - Senior. The id value for a student is unique; there will be no two students with the same id value.
11. Write a function find_average_scores that accepts a roster list (Exercise 10) and returns the average test scores of each student. Return the result as a list of pairs (tuples) containing the student name and his/her average.
12. Write a function find_student that accepts a roster list (Exercise 10) and a student id and returns the student record if found and None otherwise.
13. Write a function group that accepts a roster list (Exercise 10) and the class id and returns a list of students belonging to the class. You return a complete record for the students.