17-dict-II

March 20, 2024

1 Problems involving dictionaries

In this lecture, we will use dictionaries to solve a series of problems. Use this time to see if you understand the power of dictionaries, and how to handle them.

1.1 Grade analysis

Professor Ihab recently took the 15-110 course and learned about dictionaries. Inspired by their great versatility, he decided to store all students' grades in a dictionary where the key is the student name, and the value is a list with that student's grades for all assignments. Professor Ihab wants to compute various statistics about the class, but he has been busy grading assignments. Can you help him?

1.1.1 Average

Implement the function `average(d)` that takes as input a dictionary with students' grades as described above, and returns the class average.

For example, if:

```
grades = { "mohammed": [9,7,6,10,0],
           "ahmed": [10,10,6,8,4],
           "nour": [9,8,4,6,7],
           "carlos": [10,8,10,5,6],
           "jane": [10,9,10,8,5]
         }
```

then `average(grades)` should return 37.0 (all assignments are graded out of 10).

[1]:
```
def average(d):
    return 42
```

1.1.2 Median

Implement the function `median(d)` that takes as input a grade dictionary, and returns the class median grade.

The median of a set of value is the one exactly at the middle when the values are sorted. For example, the median of `[15,27,30,49,55]` is 30. If the set has an even number of elements, the median is the mean of the two middle values. For example, the median of `[15,27,30,49]` is (27 + 30) / 2 = 28.5.

For example `median(grades)` should return 38.

[2]:
```
def median(d):
    return 42.0
```

1.1.3 Top of the class

Now professor Ihab would like to know who is the student with the highest grade.

Implement the function `highest(d)` that takes as input a dictionary as before, and returns the name of the student with the highest grade. If there is more than one student, return the one whose name comes first alphabetically.

For example, `highest(grades)` should return `"jane"`.

[3]:
```
def highest(d):
    return ""
```

1.1.4 Below average

It is important for professors to keep track of the students that are below average, to see if/how they are improving in the course.

Implement the function `belowAvg(d)` that takes as input the grade dictionary, and returns a list with the names of all students whose grade is below the class average. The list must be sorted in alphabetical order.

For example, `belowAvg(grades)` should return the list `['mohammed', 'nour']`.

HINT: You can use the `average(d)` function implemented above if you think this will make life easier.

[4]:
```
def belowAvg(d):
    return []
```

1.1.5 Merge

Professor Ihab thought it would be useful to keep a separate dictionary for quizzes and another one for exams, but he realized everthing could be in a dicationary of dictionaries. So now we need to merge the existing ones.

Implement the function `merge(d1, d2)` that will merge two grade dictionaries in the following way: - The final dictionary contains the names of all students. - The inner dictionary has an entry for `'exams'` and an entry for `'quizzes'`, each associated to a list of values representing the grades of that student in those kinds of assessment. If the student did no exams or no quizzes, the inner dictionary should have the empty list (see example below).

For example, suppose the dictionaries are:

```
exams = { "daniel": [7,9],
          "agata": [9,10]
        }
quizzes = { "agata": [6,8,9,8],
              "martin": [9,9,8,10],
              "daniel": [6,7,5]
             }
```

then `merge(exams, quizzes)` should return:

```
res = {'daniel': {'exams': [7, 9], 'quizzes': [6, 7, 5]},
       'agata': {'exams': [9, 10], 'quizzes': [6, 8, 9, 8]},
       'martin': {'exams': [], 'quizzes': [9, 9, 8, 10]}}
```

[5]:
```
def merge(d1, d2):
    return {}
```

1.1.6 Final grades

At the end of the semester, Professor Ihab would like to have a dicionary where the keys are students' names, and the values are their final grade (instead of a list of grades per assignment).

Implement the function `finalGrades(d)` that takes as input a grade dictionary, and returns a dicionary of final grades, where each student is associated with one integer, their final grade.

For example, `finalGrades(grades)` should return the dictionary:

```
{'ahmed': 38,
 'jane': 42,
 'carlos': 39,
 'mohammed': 32,
 'nour': 34}
```

[6]:
```
def finalGrades(d):
    return {}
```