20-dict-II

November 8, 2020

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 Ravi recently took the 15-110 course and learnt 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 that holds this student's grades for all assignments. Professor Ravi wants to compute various statistics about the class, but he has been busy with some other academic work recently. Can you help him?

1.1.1 Average

Implement the function `average(d, n)` that returns the class average for a particular assignment. The function takes as input a dictionary with students' grades as described above and the assignment number, n, which is between 1 and 5, assuming there are only 5 assignments in Professor Ravi's class.

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, 1)` should return 9.6.

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

1.1.2 Median

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

For example `median(grades, 1)` should return 10.

[2]:
```
def median(d, n):
    return 42
```

1.1.3 Top of the class

Now, professor Ravi would like to know who is the student who took the highest score in any assignnment.

Implement the function `highest(d, n)` that takes as input a dictionary of grades as before, and returns the name of the student with the highest grade in assignment, n. If there is more than one student, return them all.

For example, `highest(grades, 1)` should return `"ahmed"`, `"carlos"`, and `"jane"`.

[3]:
```
def highest(d, n):
    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, n)` that takes as input the grade dictionary and assignment number, and returns a list with the names of all students whose grades are below the class average.

For example, `belowAvg(grades, 1)` should return the list `['mohammed', 'nour']`.

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

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

1.1.5 Merge

Professor Ravi thought it would be useful to keep a separate dictionary for quizzes and another one for exams, but he has changed his mind. He would like to have one dictionary with all grades, 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: - If a student is in one dictionary but not in the other, keep the student and their grades as is. - If a student is in both dictionaries, the new entry should contain that student's grades for both the exams and quizzes (in one list).

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:

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

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

1.1.6 Final grades

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

Implement the function `finalGrades(d)` that takes as input a grade dictionary, and returns a dictionary of final grades, where each student is associated with one integer, their final grade. You can assume that all assignments have an equal weight.

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

```
{'ahmed': 6.4,
 'jane': 7.6,
 'carlos': 6.8,
 'mohammed': 7.8,
 'nour': 8.4}
```

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