I590 – Python Summer 2017

Assignment No:	9	Issued:	June 25, 2017
Title:	Data Processing with NumPy	Due:	July 2, 2017

Notes:

- Please don't use any imported package, other than *NumPy*.

Data

Final grades in Computer Science CS333 course are based on the weights of the assignment groups:

Labs	30 %
Homework	20 %
Quiz	15 %
Midterm 1	10 %
Midterm 2	15 %
Final project	10 %
Total	100 %

There are assignment groups that contain a single assignment, for example Midterm 1, Midterm 2 and Final project. Other assignment groups contain several assignments that together constitute the grade for the group. For example, there are 6 homework assignments that together contribute 20% towards the final grade.

Excel file "Grades_all.xlsx" contains all grades at the end of the semester. Column A contains the student ID numbers and the other columns contain the grades:

B to G:	Homework
H to R:	Labs
S:	Final project
T:	Midterm 1
U to AC:	Quizzes (Note that Quiz 4 is missing)
AD:	Midterm 2

Rows contain the following information:

1:	Textual description of columns
2:	Weights of individual assignment within its group
3 to 46:	Grades of students in corresponding assignments

Homework and lab assignments, final project and midterms are graded with maximum of 100 points. The quizzes are graded with maximum of 10 points.

For your convenience we converted the data from the Excel file "Grades_all.xlsx" into a *NumPy* file "Grades.npy" which you are able to load by using a statement:

```
np.load("Grades.npy")
```

Note that the first row of the Excel file "Grades_all.xlss" that contains the textual description of the columns is dropped in the file "Grades.npy". Accordingly, the first row (row 0) of the file "Grades.npy" contains the weights of the individual assignments.

If you are not able to load the file from your working directory, try to give a full path. For example, if your file is in the folder:

```
C:\folder1\folder2\folder3
```

use the following statement to load the file:

```
np.load("C:/folder1/folder2/folder3/Grades.npy")
```

Note that the backslashes "\" have been replaced by slashes "/"!

Any missing value in the file should be filled out with 0. For example, the Lab 11 score for student ID 60921 is missing and should be inserted as 0.

Question (100 points)

Write a Python program that uses *NumPy* to calculate, print and save in separate files:

- The final scores (as numbers) and corresponding letter grades of students. The letter grades are explained in Assignment 6, question 1. Save results to file "FinalGrades.npy".
- Average final score and corresponding letter grade of the whole class. Save results to file "AverageGrades.npy".
- Average scores per each assignment. Save results to file "AverageAssig.npy".
- Average scores of the assignment groups. Save results to file "AverageGroup.npy".

Additional explanations

- The final scores (as numbers) and corresponding letter grades of students

  Calculate total score for each student and the corresponding letter grade. For each student, save ID number, total score and letter grade to the file.

- Average final score and corresponding letter grade of the whole class

  An (arithmetic) average is the sum of a series of numbers divided by the count of that series of numbers. To find the class average, add up all the total scores of the students, and then divide that sum by the number of students. Find the corresponding letter grade.

- Average scores per each assignment

  Add up all the scores for a specific assignment and divide by the number of students. You have to find average score for each lab, each homework, each quiz, Midterm 1, Midterm 2 and Final project.

- Average scores of the assignment groups

For the lab assignment group, first find the average scores of each lab, and then find the average of those (the average of average scores of each lab). Repeat this for homework and quizzes. Average scores of the assignment groups Midterm 1, Midterm 2 and Final Project are the same as the average scores of Midterm 1, Midterm 2 and Final Project, respectively.

Example

After running the program, the final scores are in file "FinalGrades.npy". The printout bellow shows the first few lines from the file.

```
g = np.load("FinalGrades.npy")


g
Out[117]:
[['49647.0' '82.3469696969697' 'B-']
 ['49865.0' '87.06818181818181' 'B+']
 ['88260.0' '80.4121212121212' 'B-']
 ...,
```

Hints

One way to solve this problem is to use a "for" loop and in iteration *i* process the scores of a student in the row *i*. An algorithm would look like this:

```
load the file with data
for i in range(1, row)
    for a student in row i calculate group grades and final score
convert final scores into letter grade
print and store results into npy files
```

Use *shape* method to get the dimension of an array. For example, if you read the data into a file "grades" than you can get number of rows and columns by:

```
row, col = grade.shape
```

Slice *ndarray* to select scores of an assignment group. For example, if the grades are in *ndarray* "grade", the following statement selects the homework grades of the first student:

```
grade[1][1:7]
Out[12]: array([ 90, 100,  92,  90,  40, 100])
```

The sum of the homework scores:

```
grade[1][1:7].sum(axis = 0)
Out[10]: 512
```

You may want to use a list to store letter grades, because *NumPy* complains about mixing numbers and strings in a same array. If you convert the list with letter grades to an array, you will be able to use "np.hstack" function to merge two arrays into the final array with id numbers, final scores and letter grades. *NumPy* will automatically convert the numbers into the strings and the resulting array will contain the strings.

For example, suppose that you have an array *a*:

```
a
array([[ 1.,  0.],
       [ 0.,  0.]])
```

Create an array *c*:

```
c = np.array([11, 12])

c
Out[97]: array([11, 12])
```

To merge the two arrays together, we first have to reshape *c* to make its shape compatible to *a*:

```
c = c.reshape((2,1))

c
Out[105]:
array([[11],
       [12]])
```

Merge two arrays together:

```
np.hstack((a, c))
Out[106]:
array([[  1.,   0.,  11.],
       [  0.,   0.,  12.]])
```

If you create an array of letter grades (the element of the array are strings) and merge it with an array of numbers, it will produce an array of strings:

```
d = c.astype(np.str)

d
Out[108]:
array([['11'],
       ['12']],
      dtype='<U11')

np.hstack((a, d))
Out[109]:
array([['1.0', '0.0', '11'],
       ['0.0', '0.0', '12']],
      dtype='<U32')
```

Slide 11, lecture 9.5, explains file input and output.

NumPy may produce the values in scientific notation. To suppress scientific notation you may want to use statement:

```
np.set_printoptions(suppress=True)
```

Use option `precision` to reduce the number of fractional places. For example, use the following statement to print 2 fractional places:

```
np.set_printoptions(precision = 2)
```

Deliverables

- Write your programs in Python 3.
- Make sure your code is properly formatted, structured and commented.
- Name program file 'Question.py'.
- A program should run the entire code by directly executing it.
- Submit a single zip file that includes the question program files together. Name the zip file '***IU-user-name_assignmentNumber.zip***'. For example, if your IU username is 'rakshah', your zip file will have name '***rakshah_assignment5.zip***.'
- If you want to provide more instructions for executing your code, put a 'readme.txt' file inside of this zip file, where you may provide additional information.