Koç University COMP125-01/02/03
Programming with Python: Midterm 6
Instructors: Mehmet Emre Gürsoy, Mehmet Sayar, Barış Akgün
Date: January 3, 2021

Start: 12:00 noon
Deadline to submit on Blackboard: 1:15pm

Instructions

Make sure you read and understand every part of this document

Download & Extract the zip to your Desktop:

Blackboard -> Assessments -> Midterm 6 -> MT6.zip

Open in Spyder:

LinearSystemSolver.py StudentGrades.py MiniFunctions.py

- Do not change the names of the files. Do not change the names and parameters of the functions that are given to you.
- You are allowed to import only the following: math, random, numpy, matplotlib. Do not use external libraries or modules other than these.

When you are finished:

Submit a single compressed archive (zip file) containing:

LinearSystemSolver.py StudentGrades.py MiniFunctions.py

Multiple attempts are allowed, so upload ASAP, clean up later.

We will grade only the LAST version submitted to Blackboard.

By submitting this midterm, you certify that you have already completed the Honor Pledge essay given on Blackboard. If you have not done so, we reserve the right to dismiss your midterm.

"You certify that you have completed this exam on your own without any help from anyone else. You understand that the only sources of authorized information in this exam are (i) the course textbook, (ii) the material that is posted at Blackboard for this class, and (iii) any study notes handwritten by yourself. You have not used, accessed or received any information from any other unauthorized source in taking this exam. The effort in the exam belongs completely to you."

Communication before & during exam:

Zoom session (Meeting ID: 922 5796 1536 Passcode: 129757)

Starts at 11:45 am

Students are not allowed to speak or chat via this channel once the exam starts.

Violations may be punished via disciplinary action !!!

For clarification questions:

Raise your hand in Zoom.

We will transfer you to a breakout room.

Only clarification questions regarding the midterm are allowed.

Installation related or my_code_is_not_working questions will not be answered.

Q1: Linear System Solver - 30 pts (15+15 pts)

You are given a file containing a system of linear equations, such as: eq1.txt, eq2.txt, eq3.txt, … The following screenshot is from eq1.txt:

```
x,y,z
2,1,-2,3
1,-1,-1,0
1,1,3,12
```

This is equivalent to the following system of linear equations:

$$2x + 1y - 2z = 3$$
$$1x - 1y - 1z = 0$$
$$1x + 1y + 3z = 12$$

The first line of the file contains the names of the unknowns, e.g., x,y,z. You can assume each unknown is represented by a single letter. You must parse this line to understand how many unknowns exist in the system and what are their names.

In each of the remaining lines, there is one equation. You must parse these lines to construct the system of linear equations in Python. You can assume that inputs will be valid and you do not need to perform any error/exception checking. Furthermore, you can assume that if there are N unknowns, there are N equations.

Open LinearSystemSolver.py and implement the following parts.

Part A: Write a function called read_system(filename) such that:

- filename is the name of the file that should be read, e.g., “eq1.txt”
- read_system has 3 return values:
    - First return value is a Python list (not ndarray) containing the names of the unknowns. Example: [x, y, z] (Hint: your solution should be general, i.e. should work for any number of unknowns)
    - Second return value is the ndarray A that will be used in solving the system via Numpy, as in: Ax = b (do not solve the system yet!)
    - Third return value is the ndarray b that will be used in solving the system via Numpy, as in: Ax = b

Example return values, when printed to console:

```
unknown names:['x', 'y', 'z']
A:
[[ 2.  1. -2.]
 [ 1. -1. -1.]
 [ 1.  1.  3.]]
b:
[[ 3.]
 [ 0.]
 [12.]]
```

Part B: Write a function called solve_system(A, b) such that:

- A is the ndarray constructed in the previous part (Ax = b)
- b is the ndarray constructed in the previous part (Ax = b)
- solve_system must use Numpy to solve the system of linear equations
- Return value is an ndarray that contains the solution (i.e., the values of the unknowns)

Example return value of solve_system, when printed to console:

```
[[3.5]
 [1. ]
 [2.5]]
```

Q2: Student Grades - 45 pts (10+10+10+15 pts)

You are given a file containing student grades: student_grades.txt.

```
100,8
mid1,hw1,mid2,hw2,mid3,hw3,final,lab
94.98,94.38,63.49,62.53,40.63,22.62,41.31,87.78
87.14,75.94,32.68,32.76,54.55,81.76,71.27,92.14
54.36,56.08,64.01,66.00,41.36,59.10,40.38,83.50
75.95,68.90,31.11,32.35,0.00,0.00,60.96,91.86
29.16,30.09,53.35,56.11,72.95,43.28,63.14,86.46
```

The first line contains two integers separated by a comma. The first integer corresponds to how many students exist (in the above example, 100). The second integer corresponds to how many columns exist (in the above example, 8).

The second line contains the headers of the grades table: mid1 is Midterm 1, hw1 is Homework 1, mid2 is Midterm 2, and so forth. final is the final exam grade, lab is the lab grade.

Then, each remaining line in the text file corresponds to the grades of one student separated by commas. You may assume that there are no errors in the file, e.g., no erroneous or empty grades, all grades are legitimate (between 0 and 100).

Open StudentGrades.py and implement the following functions.

Part A: Write a function called read_data(filename) such that:

- filename is the name of the file that should be read, e.g., “student_grades.txt”
- read_data returns a NumPy ndarray that contains the grades of all students.

For example, for the above file, the return value is a 100x8 array that produces the following output when printed to console (this is a partial output, the full array is quite large):

```
[[ 94.98  94.38  63.49  62.53  40.63  22.62  41.31  87.78]
 [ 87.14  75.94  32.68  32.76  54.55  81.76  71.27  92.14]
 [ 54.36  56.08  64.01  66.    41.36  59.1   40.38  83.5 ]
 [ 75.95  68.9   31.11  32.35   0.     0.    60.96  91.86]
 [ 29.16  30.09  53.35  56.11  72.95  43.28  63.14  86.46]
```

Part B: Write a function called calc_variance(data, col_name) that calculates the variance in the corresponding grade column.

- data is the ndarray constructed in Part A.
- col_name is a string containing the column name. Example: col_name = “hw2” or col_name = “mid3” or col_name = “final”

- calc_variance must retrieve the corresponding column and calculate the variance of that column.
- Return value of calc_variance is a single float corresponding to the variance.

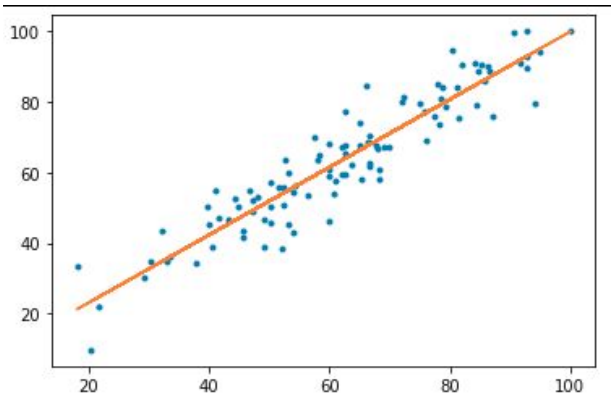
Part C: When calculating each student's course grade, each midterm weighs 15%, each homework weighs 5%, final exam is 30% and lab grade is 10%. Write a function called course_grade_average(data) such that:

- data is the ndarray constructed in Part A.
- The function computes each student's course grade, according to the weights.
- The function returns the average (mean) course grade of all students.

Part D: Write a function called plot_stats_and_fit(data, item) such that:

- data is the ndarray constructed in Part A.
- item is a single integer: item = 1 or item = 2 or item = 3.
- Say that item = N. Then:
    - plot_stats_and_fit retrieves the midN and hwN scores of the students.
    - It treats midN scores as the x coordinates and hwN scores as the y coordinates.
    - Using least squares optimization, it fits a line to capture the relationship between midN and hwN.
    - Finally, it plots the data points and the linear line together on the same graph.
- It is sufficient for the plot to appear in the "Plots" tab of Spyder.
- plot_stats_and_fit has no return value.

Here is the sample plot for N = 1, which captures the relationship between hw1-mid1:



Q3: Mini Functions - 25 pts (10+15 pts)

Open MiniFunctions.py and implement the following functions.

func1(n):

Implement a function called func1 such that:

- func1 has 1 input: integer n (you can assume n ≥ 3, no need for error checking)
- func1 returns an n by n Numpy matrix that has an outer frame of 1s and all inner values are 0s

Return value when n=4:

```
[[1. 1. 1. 1.]
 [1. 0. 0. 1.]
 [1. 0. 0. 1.]
 [1. 1. 1. 1.]]
```

Return value when n=5:

```
[[1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1.]]
```

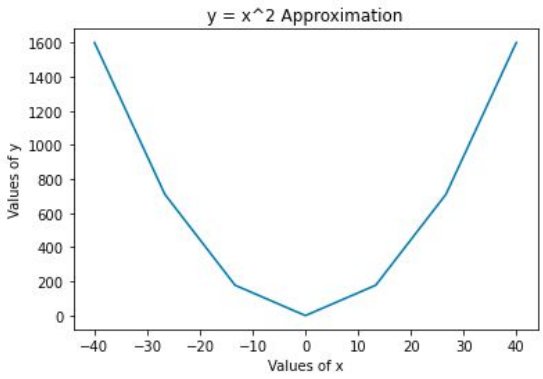
func2(n):

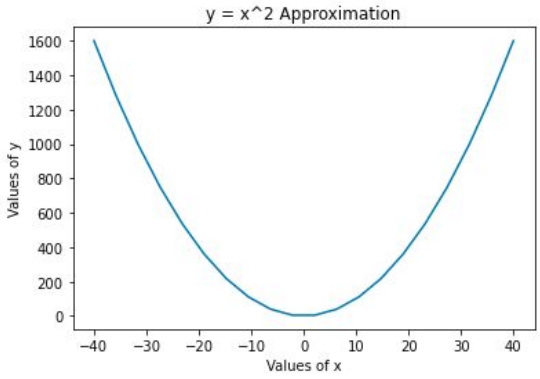
We would like to approximate the function $y = x^2$ by evaluating it at multiple x values and plotting the corresponding values of the function.

Implement a function called func2 such that:

- func2 has 1 input: integer n
- func2 creates n equally spaced x values between [-40, +40] and computes $y = x^2$ at each of these x values. x values should start with -40 and end with +40.
- func2 creates a line graph of the x values and y values found in the previous step
    - Title of plot should be: “y = x^2 Approximation”
    - X axis label should be: “Values of x”
    - Y axis label should be: “Values of y”
    - You do not need to modify any other details of the plot

Here are the plots generated by func2 when n=7 and n=20 respectively. As n becomes larger, you should be able to visually verify that the line graph is a better (smoother) approximation of the function $y = x^2$.





****

Submission

Solve each question in its own file. **Do not change the names of the files. Do not change the names and parameters of the functions that are given to you.**

Only the following imports are allowed: math, random, numpy, matplotlib. Do not use external libraries or modules other than these.

When you are finished, compress your MT6 folder containing your answers:
LinearSystemSolver.py StudentGrades.py MiniFunctions.py

The result should be a SINGLE compressed file (file extension: .zip, .rar, .tar, .tar.gz, or .7z). Upload this compressed file to Blackboard.

<u>You may receive 0 if one or more of the following is true:</u>

- You do not follow the submission instructions, i.e., you submit something other than a single compressed file.
- Your compressed file does not contain your Python files or the file names are wrong.
- Your compressed file is corrupted.
    - **After you submit, you should download your submission from Blackboard to make sure it is not corrupted and it has the latest version of your code.**
- Your code contains syntax errors.
- You use external libraries or modules other than the ones allowed.
- Your code does not run without errors the way that it is submitted.
    - **We should not have to open your file and comment/uncomment parts of it in order to run your code.**
- Your code does not terminate, e.g.: it contains infinite loops.