Koç University
COMP 125: Programming with Python
Homework #6

Deadline: January 22 , 2021 at 10:00 (Morning - No Late submissions!)
Submission through: Blackboard
Make sure you read and understand every part of this document

This homework assignment contains 3 programming questions. Each question may contain multiple parts.

Download Hw6.zip from Blackboard and unzip the contents to a convenient location on your computer. The .ipynb files contain starter codes for the programming questions. You should open them with Jupyter Notebook (not Spyder). Remaining files are sample text/data files.

Solve each question in its own file. **Do not change the names of the files. Do not change the headers of the given functions (function names, function parameters).** When you are finished, see the end of this document for submission instructions.

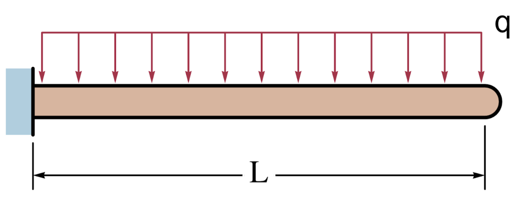
- Your submitted code should run as is. It should not yield syntax errors.
- We will not edit or comment/uncomment parts of your code in order to fix syntax errors.

Q1: Deformation of a Cantilever Beam - 35 pts

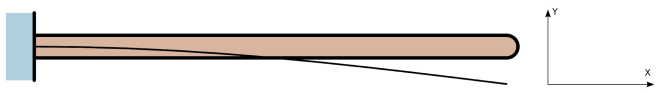
Cantilever beams are common structural elements. Given a cantilever beam of length **L** with a uniform distributed load **q** (Fig. a), the deflection of the cantilever beam along its length (Fig. b) is given by the following equation (x, is the distance from the fixed end):

$$y(x) = -\frac{qx^2}{24EI}\left(x^2 - 4Lx + 6L^2\right)$$

a) Cantilever beam



b) Deflection profile



Using Numpy and MatPlotLib solve the following questions

1) Given that the cantilever is L =10 m long, with modulus of elasticity E =200 10^9 Pa, area moment of inertia of cross section I =10^{-4} m^4, and the distributed load q =500 N/m find the deflection profile of the beam, y(x), by using Numpy. y(x) should be a numpy array corresponding to the deflection at distance x from the fixed end. Write the deflection profile y(x) as a function the distance x to a file in csv format, such that each line contains the distances and deflection values as follows:

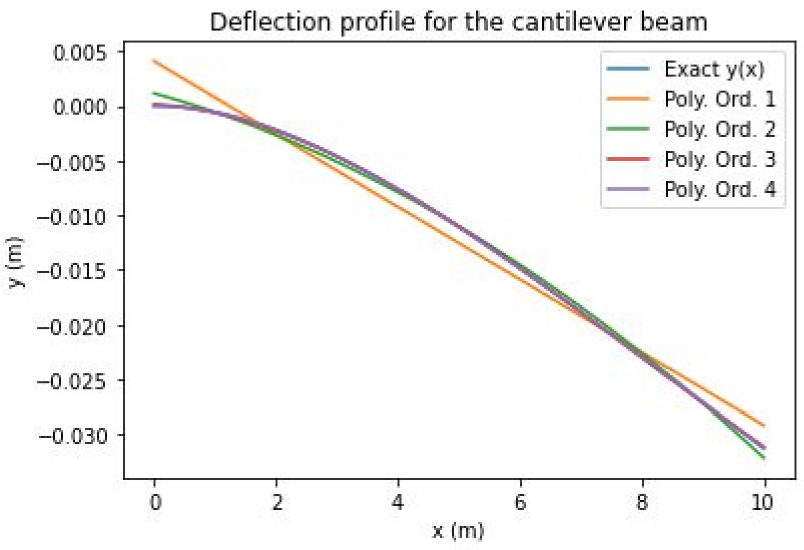
```
0.0, -0.0
0.10101010101010102, -6.334066666242626e-06
0.20202020202020204, -2.516579958172705e-05
0.30303030303030304, -5.624145003475499e-05
0.4040404040404041, -9.930987186451741e-05
0.5050505050505051, -0.00015412252146109444
```

The file should be named as “deflections.txt”

2) By using Numpy function polyfit, fit polynomials of degree 1 upto 4 (i.e. 4 separate curve fits). Write the coefficients of the polynomial fits to a file name “coefficients.txt”. The file should contain 4 lines, with each line displaying the coefficients separated by commas. No other information should be written to the file. Sample output

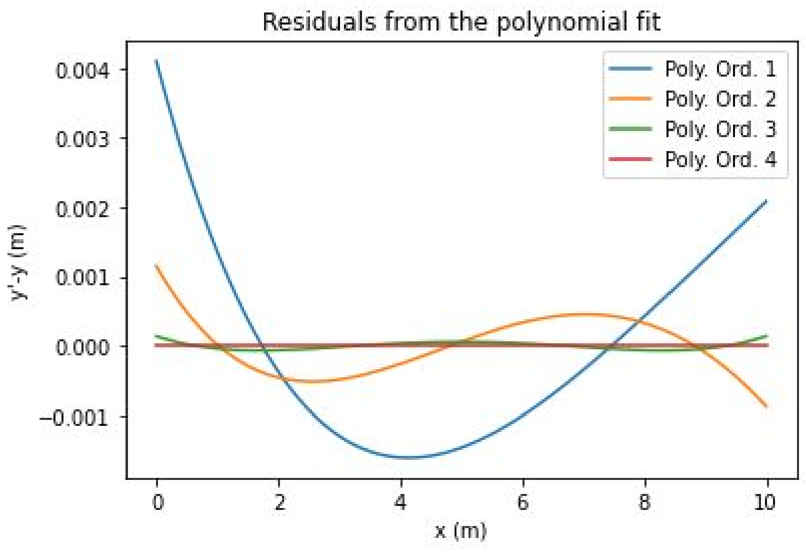
```
-0.0033270627146889786, 0.004103712847505999
-0.00017901477497437072, -0.001536914964945267, 0.0011502704321039176
2.0833333333333346e-05, -0.0004915147749743705, -0.00029318558358962585, 0.00013995685865902802
-1.0416666666666616e-06, 4.1666666666666415e-05, -0.0006249999999999978, -4.3157813326681086e-18, 7.632783294297952e-18
```

4) Plot the results from the analytical solution for the deformation profile (x, y(x)), and the polynomial fits of order 1 upto 4. The plot should be formatted as follows:



Note that the exact solution is not visible, since the 4th order polynomial exactly matches the analytical result.

5) Plot the residuals for each polynomial fit. i.e. y'(x)-y(x), where y'(x) is the result obtained from the polynomial fit and y(x) is the analytical solution. Confirm that the 4th order polynomial yields an exact fit.



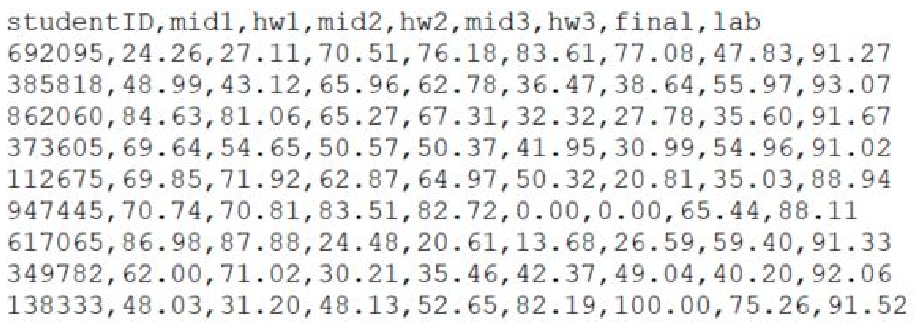
You should submit only the cantilever.ipynb file. When executed the file should produce the outputs:

1) deflections.txt file,
2) coefficients.txt file,
3) Plot for the deformation profile as a png file named as deformation.png,
4) Plot for the residuals as a png file named residuals.png.

Q2: New Student Grades - 30 pts

Recall the “Student Grades” question from Midterm 6. Here, you will solve a similar question but without numpy. Instead, you will use lists, dictionaries, tuples, etc. so that you can refresh your knowledge of these topics.

Consider the following file: new_student_grades.csv.



```
studentID,mid1,hw1,mid2,hw2,mid3,hw3,final,lab
692095,24.26,27.11,70.51,76.18,83.61,77.08,47.83,91.27
385818,48.99,43.12,65.96,62.78,36.47,38.64,55.97,93.07
862060,84.63,81.06,65.27,67.31,32.32,27.78,35.60,91.67
373605,69.64,54.65,50.57,50.37,41.95,30.99,54.96,91.02
112675,69.85,71.92,62.87,64.97,50.32,20.81,35.03,88.94
947445,70.74,70.81,83.51,82.72,0.00,0.00,65.44,88.11
617065,86.98,87.88,24.48,20.61,13.68,26.59,59.40,91.33
349782,62.00,71.02,30.21,35.46,42.37,49.04,40.20,92.06
138333,48.03,31.20,48.13,52.65,82.19,100.00,75.26,91.52
```

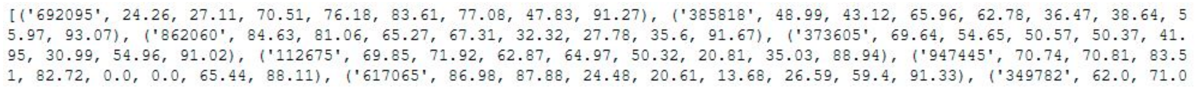
The first line contains the headers of the table: studentID is the 6-digit university ID of the student, mid1 is Midterm 1, hw1 is Homework 1, mid2 is Midterm 2, and so forth. final is the final exam grade, lab is the lab grade.

Then, each remaining line in the csv 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 NewStudentGrades.ipynb 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., “new_student_grades.csv”
- read_data returns a list of tuples where each tuple corresponds to one student. Tuple content should be in the same order as in the file, i.e.:
  (studentID, mid1, hw1, mid2, hw2, mid3, hw3, final, lab)
- studentID should be string, rest should be floats.



```
[('692095', 24.26, 27.11, 70.51, 76.18, 83.61, 77.08, 47.83, 91.27), ('385818', 48.99, 43.12, 65.96, 62.78, 36.47, 38.64, 5
5.97, 93.07), ('862060', 84.63, 81.06, 65.27, 67.31, 32.32, 27.78, 35.6, 91.67), ('373605', 69.64, 54.65, 50.57, 50.37, 41.
95, 30.99, 54.96, 91.02), ('112675', 69.85, 71.92, 62.87, 64.97, 50.32, 20.81, 35.03, 88.94), ('947445', 70.74, 70.81, 83.5
1, 82.72, 0.0, 0.0, 65.44, 88.11), ('617065', 86.98, 87.88, 24.48, 20.61, 13.68, 26.59, 59.4, 91.33), ('349782', 62.0, 71.0
```

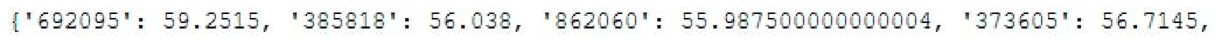
Part B: Write a function called midterm_averages(data) such that:

- data is the list of tuples constructed in Part A.
- midterm_averages has 3 return values in the following order:
  mid1_grade_avg, mid2_grade_avg, mid3_grade_avg

Part C: Write a function called cumulative_grades(data, mt_weight, hw_weight, final_weight, lab_weight) such that:

- data is the list of tuples constructed in Part A.

- You have to change the function header according to the following:
  - By default, each midterm weight is 0.15, each homework weight is 0.05, final exam weight is 0.3, lab weight is 0.1 (total is 1)
  - If the user does not specify any weights when calling the function, the default weights listed above should be used.
  - If the user specifies different weights when calling the function, the function should use the user-specified weights.
- cumulative_grades calculates each student's cumulative course grade according to the weights.
- Return value of cumulative_grades is a dictionary: key is studentID (string), value is the cumulative course grade of the student (float)



```
{'692095': 59.2515, '385818': 56.038, '862060': 55.987500000000004, '373605': 56.7145,
```

Part D: Write a function called passing_students(dic, threshold) such that:
- dic is the dictionary created in Part C.
- threshold is a float value corresponding to the passing grade of the course.
- passing_students returns a list of studentIDs (a list of strings) corresponding to the students who passed the course. Students who passed the course are those whose cumulative grade is higher than threshold.

Part E: Write a function called write_to_file(dic, threshold) such that:
- dic is the dictionary created in Part C.
- threshold is a float value corresponding to the passing grade of the course.
- write_to_file should write ONE LINE PER STUDENT to a file called "results.txt":
  - If the student passed the course, the line should be in the format:
    Student: **1234, course average: 73.921, PASSED
  - If the student failed the course, the line should be in the format:
    Student: **1234, course average: 26.308, FAILED

Notice that:
- The first two digits of the student ID are replaced by asterisks (**) to preserve anonymity, only the last 4 digits are printed.
- Course average contains two digits before the decimal point, three digits after the decimal point.
- The string "PASSED" or "FAILED" is written at the end according to whether the student's average is above or below the threshold.

Part F: Write a function called sort_students(dic) such that:
- dic is the dictionary created in Part C.
- sort_students returns a list of student IDs (only the student IDs) sorted in DESCENDING order of their cumulative grades.

Q3: Time Series and Matplotlib - 35 pts

The file TimeSeries.ipynb has all the information related to the question.

Submission and Grading

Solve each question in its own file. **Do not change the names of the files. Do not change the headers of the given functions (function names, function parameters).**

When you are finished, compress your Hw6 folder containing all of your answers. The result should be a SINGLE compressed file (file extension: .zip, .rar, .tar, .tar.gz, or .7z). Upload this compressed file to Blackboard.

Follow instructions, input-output formats, return values closely. Your code may be graded by an autograder, which means any inconsistency will be automatically penalized.

<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 code 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.
- 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.

You are only going to be graded based on your Blackboard submission. We will not accept homework via e-mail or other means.

Best of luck and happy coding!