Koç University COMP125
Programming with Python: Midterm 2
Instructor: Beren Semiz
Date: April 15th – April 18th, 2021

The exam will be available on April 15th at 9:00 AM
Deadline to submit on Blackboard: April 18th at 11:59 PM (23:59)

Instructions
Make sure you read and understand every part of this document

Download & Extract the zip to your Desktop:
Blackboard -> Assessments -> MT2-> MT2.zip

Zip file includes:
vectorExercise.py castingAgency.py gradeDictionary.py matrixOperations.py
Midterm2_Spring2021.pdf

Open the .py files using Spyder.

Do not change the names of the files. Do not change the names of the functions that are given to you. Do not use Turkish characters. You may add new functions if you want to (helper functions).

When you are finished:
Submit a single compressed archive (.zip or .rar file) containing:
vectorExercise.py castingAgency.py gradeDictionary.py matrixOperations.py

Multiple attempts are allowed. We will grade only the LAST version submitted to Blackboard. The deadline is a strict deadline. There won't be a late-submission option as you have in homework submissions. **After 11.59 PM (23.59), the system will be closed.**

By submitting this file, you automatically agree to the honor pledge: "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:
Only clarification questions regarding the midterm are allowed.
Installation related or my_code_is_not_working questions will not be answered.

Q0: Honor Pledge

Between April 15th 9.00 AM and April 18th 11.59 PM (23.59) Honor Pledge will be available under **Blackboard -> Assessments -> MT2 Honor Pledge.** You will enter the text you see to the textbox and SIGN IT WITH YOUR NAME to indicate that you accept the Honor Pledge for this midterm. If the honor pledge is not completed and signed, we reserve the right to dismiss your midterm.

Q1: Vector Exercise - 20 pts

In this question, you will be given a list including N vectors in the format [vec1, vec2, …] where each element of the list is a tuple corresponding to a vector's coordinates, i.e., vec1 is a tuple corresponding to the first vector, vec2 is a tuple corresponding to the second vector, and so forth. You can assume that the given list will always consist of 2D vectors having (x, y) coordinates.

Open *vectorExercise.py* and perform the following task.

Cosine of the angle, $cos\theta$, between any given vector pair is calculated based on the following formula:

$$cos\theta = \frac{\underset{u}{\rightarrow} \cdot \underset{v}{\rightarrow}}{\left|\underset{u}{\rightarrow}\right| \left|\underset{v}{\rightarrow}\right|}$$

where:

$\underset{u}{\rightarrow} \cdot \underset{v}{\rightarrow}$ is the dot product of the vectors $u(x_1, y_1)$ and $v(x_2, y_2)$ calculated as $x_1 * x_2 + y_1 * y_2$

$\left|\underset{u}{\rightarrow}\right|$ is the magnitude of the given vector, which is calculated as $\sqrt{x^2 + y^2}$

For example, for the following pair: (-4,0) and (4,0)

$\underset{u}{\rightarrow} \cdot \underset{v}{\rightarrow} = -4 * 4 + 0 * 0 = -16$

$\left|\underset{u}{\rightarrow}\right| = \sqrt{(-4)^2 + 0^2} = 4$

$\left|\underset{v}{\rightarrow}\right| = \sqrt{4^2 + 0^2} = 4$

So, $cos\theta = \frac{\underset{u}{\rightarrow} \cdot \underset{v}{\rightarrow}}{\left|\underset{u}{\rightarrow}\right| \left|\underset{v}{\rightarrow}\right|}$ will result in $\frac{-16}{4 * 4} = -1.0$

Implement a function calculate_cos(vectors) which takes a list of vectors and returns the two vectors that have the minimum $cos\theta$ value (return order is not important). Note: Since you are returning two vectors, your function should have two return values.

Examples:
[(7,1), (5,5), (-4,0)] should return (7, 1), (-4, 0)
[(-4,0), (4,0), (3,4)] should return (-4, 0), (4, 0)
[(3,2), (-2,7), (6,1), (1,3)] should return (-2,7), (6,1)

HINT: *Lab 5 questions will be really helpful!*

RULES:

1. You can use the math module only for the square-root calculation. You cannot use any other module or library (such as numpy).
2. Make sure that you are not returning any self-pairs.

Q2: Casting Agency - 20 pts

You are working at a casting agency and you are given a list of un-organized data including the names of the movie characters and their ID numbers. The data you need to process will always contain firstname, lastname and ID number separated by dashes. Firstnames will always consist of lowercase letters and lastnames will always consist of uppercase letters. However, the order of firstname, lastname and ID number may be different. That is, one data entry could have “firstname-lastname-ID” but another data entry could have “lastname-ID-firstname”. Few examples: 'severus-SNAPE-2343', 'POTTER-5674-harry', etc.

Your task is to organize the given data such that each entry will form a tuple and have the format (‘Firstname’, ‘LASTNAME’, ID). While constructing these tuples, you need to **capitalize the first letter of the firstname** and **convert the ID number to an integer**.

Open *castingAgengy.py* and implement the code casting_record(rec) which takes a list including the un-organized data and returns a list including the organized tuples.

For example:

```
rec = ['severus-SNAPE-2343', 'POTTER-5674-harry', 'WEASLEY-4389-ron', '2365-draco-MALFOY', 'hermione-1478-GRANGER']
```

should return

```
[('Severus', 'SNAPE', 2343), ('Harry', 'POTTER', 5674), ('Ron', 'WEASLEY', 4389), ('Draco', 'MALFOY', 2365), ('Hermione', 'GRANGER', 1478)]
```

HINT: *Read the first paragraph carefully: firstname, lastname and ID will always have the same format– so string operations in Lecture 9 will be really helpful!*

Q3: Grade Dictionary - 24 pts

You are given a dictionary of dictionaries including the math and science exam scores for students in a class. Test cases may include varying number of students, but they will always include only math and science exam scores.

Here is a sample dictionary of dictionaries for you:

```
test_d = {'Maria': {'math': 67, 'science': 45},
          'John': {'math': 88, 'science': 90},
          'Sarah': {'math': 65, 'science': 90},
          'Albert': {'math': 74, 'science': 60},
          'Bob': {'math': 100, 'science': 65}}
```

Open *gradeDictionary.py* and perform the following tasks:

Part A: (12 pts)

Write a function calculate_avg(d) which takes a dictionary of dictionaries as an input and calculates the average score for the math and science exams given in this dictionary. There should be two return values: the average for math exams and the average for science exams (order is important: first math average, then science average). Return values should be floating-point numbers.

Examples:

```
test_d = {'Maria': {'math': 67, 'science': 45},
          'John': {'math': 88, 'science': 90},
          'Sarah': {'math': 65, 'science': 90},
          'Albert': {'math': 74, 'science': 60},
          'Bob': {'math': 100, 'science': 65}}
```

should return 78.8, 70.0

```
test_d = {'Nate': {'math': 47, 'science': 65},
          'Claudia': {'math': 90, 'science': 90},
          'Meghan': {'math': 66, 'science': 87},
          'Harry': {'math': 77, 'science': 56}}
```

should return 70.0, 74.5

Part B: (12 pts)

Write a function higher_grade(d) which takes a dictionary of dictionaries as an input and determines the course that each student has received a relatively higher grade. Your return value should be a list of tuples. Each tuple should consist of the name of the student and the course.

> **RULE:** If the science and math exam scores for any student are the same, then instead of the course name, you should have '-1' (string) inside the tuple.

Examples:

```
test_d = {'Maria': {'math': 67, 'science': 45},
          'John': {'math': 88, 'science': 90},
          'Sarah': {'math': 65, 'science': 90},
          'Albert': {'math': 74, 'science': 60},
          'Bob': {'math': 100, 'science': 65}}
```

should return [('Maria', 'math'), ('John', 'science'), ('Sarah', 'science'), ('Albert', 'math'), ('Bob', 'math')]

```
test_d = {'Nate': {'math': 47, 'science': 65},
          'Claudia': {'math': 90, 'science': 90},
          'Meghan': {'math': 66, 'science': 87},
          'Harry': {'math': 77, 'science': 56}}
```

should return [('Nate','science'), ('Claudia', '-1'), ('Meghan', 'science'), ('Harry', 'math')]

Q4: Matrix Operations - 36 pts

You are given an N*N matrix A. You can assume that the input matrix will always be a valid square matrix (number of rows is same as number of columns) - you do not need to check for erroneous inputs. You cannot use any module or library (such as numpy) while solving the questions. Open *matrixOperations.py* and perform the following tasks:

Part A: (12 pts)
Implement a function matrix_transpose(A) which takes the matrix A as an input, computes its transpose (switching the row and column indices the elements, $A_{ij} \leftarrow A_{ji}$), and returns this computed transpose.

Example: A = [[1, 4, 2],
[2, 1, 7],
[1, 1, 1]] should return

[[1, 2, 1],
[4, 1, 1],
[2, 7, 1]]

Part B: (12 pts)
A symmetric matrix is a square matrix that is equal to its transpose. Implement a function symmetric_matrix(A) which takes the matrix A as input and checks whether the matrix A is symmetric or not. If it is symmetric, your function should return 1 (integer), if not your function should return -1 (integer). *You can directly call the matrix_transpose(A) function you wrote in Part A to compute the transpose.*

Example: A = [[1, 4, 2],
[2, 1, 7],
[1, 1, 1]] should return -1

A = [[1, 2, 3],
[2, 2, 8],
[3, 8, 6]] should return 1

Part C: (12 pts)
A diagonal matrix is a matrix such that all elements other than the main diagonal are 0. Implement a function diagonal_matrix(A) which takes the matrix A as input and updates its elements such that all elements are replaced with zeros except the elements in the first diagonal. Return value should be the new matrix you computed.

Example: A = [[1, 4, 2],
[2, 1, 7],
[1, 1, 1]] should return [[1, 0, 0],
[0, 1, 0],
[0, 0, 1]]

A = [[1, 2, 3],
[2, 2, 8],
[3, 8, 6]] should return [[1, 0, 0],
[0, 2, 0],
[0, 0, 6]]

HINT: *Lab 6 questions will be really helpful!*

Submission + Honor Pledge

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). DO NOT USE TURKISH CHARACTERS.**

When you are finished, compress your MT 2 containing all of your answers:

vectorExercise.py   castingAgency.py   gradeDictionary.py   matrixOperations.py

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