

Koç University

COMP 125: Programming with Python

Homework #3

Deadline: May 2, Sunday at 23:59pm

Submission through: Blackboard

Make sure you read and understand every part of this document

This homework assignment contains 3 programming questions.

Download [Hw3.zip](#) from Blackboard and unzip the contents to a convenient location on your computer. Each file in [Hw3.zip](#) contains the starter code for one programming question.

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.** *If you want, you can add helper functions.*

When you are finished, compress your Hw3 folder containing all of your answers. The result should be a SINGLE compressed file (file extension must be .zip, or .rar). Upload this compressed file to Blackboard.

Q1: Daily Temperatures - 45 pts

You are given two data files: [ISTANBUL.txt](#) and [ANKARA.txt](#), which contain daily temperature readings from these two cities. The data contains 4 columns: first column is month, second column is day, third column is year, fourth column is average daily temperature (in Fahrenheit). Note that there are some entries with value “-99” in the temperature column; this is because the temperature reading for that day is missing.

1	1	1995	46.2
1	2	1995	50.9
1	3	1995	46.7
1	4	1995	41.6
1	5	1995	41.8
1	6	1995	36.9
1	7	1995	35.6
1	8	1995	34.7
1	9	1995	32.4
1	10	1995	33.4

Open [DailyTemperatures.py](#) and solve the following parts. **While opening the files, please do not use any path specific to your computer. When you have the .txt and .py files in the same folder, you don't need to specify the path, you can directly open the file you want using its name.**

Part A: (15 pts)

Write a function called `read_temperatures(filename)` to read the data from the given text file into a matrix (stored as a list of lists).

- `filename` is the name of the file that should be read, e.g., "ANKARA.txt"
- The return value should be a matrix in list of lists representation such that:
 - Each row in the matrix is one row of the data file
 - First column is month, second column is day, third column is year, fourth column is average daily temperature (same column order as the file)
- Convert day, month, year to int. Convert temperature to float.

Partial output for ANKARA.txt when the matrix is printed to console (full matrix is quite large):

```
[[1, 1, 1995, 46.2], [1, 2, 1995, 50.9], [1, 3, 1995, 46.7], [1, 4, 1995, 41.6], [1, 5, 1995, 41.8], [1, 6, 1995, 36.9], [1, 7, 1995, 35.6], [1, 8, 1995, 34.7], [1, 9, 1995, 32.4], [1, 10, 1995, 33.4], [1, 11, 1995, 37.2], [1, 12, 1995, 37.0], [1, 13, 1995, 42.0], [1, 14, 1995, 41.2], [1, 15, 1995, 37.2], [1, 16, 1995, 32.1], [1, 17, 1995, 26.6], [1, 18, 1995, 24.2], [1, 19, 1995, 24.3], [1, 20, 1995, 24.4], [1, 21, 1995, 23.5], [1, 22, 1995, 20.6], [1, 23, 1995, 25.9], [1, 24, 1995, 32.3], [1, 25, 1995, 36.5], [1, 26,
```

HINT: In addition to trailing whitespace characters, pay attention to leading whitespace characters as well.

Part B: (15 pts)

Write a function called `delete_year(data, year)` to remove data from a given `year`.

- `data` is the matrix from Part A
- The return value should contain all rows from `data` minus those rows that have year equal to the given `year`

For example, if `year = 2020`, then the return value (list of lists) of `delete_year` contains all rows other than those rows with year equal to 2020.

Part C: (15 pts)

Write a function called `monthly_averages(data, year)` to calculate month-by-month average temperatures for the given `year`.

- `data` is the matrix from Part A
- `year` is an integer, e.g.: `year = 2004` or `year = 2019`
- The return value should be a list of length 12:

[`avg_of_January`, `avg_of_February`, `avg_of_March`, ..., `avg_of_December`]

- When calculating the monthly averages, only calculate the monthly averages for the given `year`. Do not calculate a multi-year average.
- Make sure that you account for missing data ("-99"). For example, if 2 days out of 30 are missing for a certain month, then you need to calculate the average across the remaining 28 days. In other words, do not take "-99"s into consideration.

Sample output for ANKARA data and 2017 when the return value is printed to console:

```
[25.22258064516129, 33.49285714285714, 43.719354838709684,
48.89999999999999, 57.222580645161294, 65.33, 74.31935483870967,
73.63548387096773, 69.21, 51.164516129032265, 41.016666666666665,
36.98064516129033]
```

Q2: Iris Dataset - 30 pts

Iris flower dataset is a multivariate dataset which includes sepal and petal characteristics of three different iris flower species (Iris Setosa, Iris Virginica and Iris Versicolor). It was first introduced by Ronald Fisher in 1936 and since then has been widely used in many statistical classification projects as a test dataset to validate the proposed algorithms. (If you are curious about the dataset, you can read the following paper: *The Use of Multiple Measurements in Taxonomic Problems*.)

Here are the images for these three flower species (*images were taken from Wikipedia*)



Iris Setosa



Iris Virginica



Iris Versicolor

Assume that you are working at a research lab and you need to perform some analyses on the Iris dataset. You are given a csv file: [iris_data.csv](#) and a starter code [AnalyzeIris.py](#)

You can check the contents of [iris_data.csv](#) either using Excel (if your settings are correct) or using another program (such as TextEdit or NotePad++)

	A	B	C	D	E
1	sepal_length	sepal_width	petal_length	petal_width	variety
2	5.1	3.5	1.4	0.2	Setosa
3	4.9	3	1.4	0.2	Setosa
4	4.7	3.2	1.3	0.2	Setosa
5	4.6	3.1	1.5	0.2	Setosa
6	5	3.6	1.4	0.2	Setosa
7	5.4	3.9	1.7	0.4	Setosa
8	4.6	3.4	1.4	0.3	Setosa
9	5	3.4	1.5	0.2	Setosa
10	4.4	2.9	1.4	0.2	Setosa
11	4.9	3.1	1.5	0.1	Setosa

```
iris_data.csv
sepal_length,sepal_width,petal_length,petal_width,variety
5.1,3.5,1.4,0.2,Setosa
4.9,3.1,1.4,0.2,Setosa
4.7,3.2,1.3,0.2,Setosa
4.6,3.1,1.5,0.2,Setosa
5.3,6.1,4.0,2.0,Setosa
5.4,3.9,1.7,0.4,Setosa
4.6,3.4,1.4,0.3,Setosa
5.3,4.1,5.0,2.0,Setosa
4.4,2.9,1.4,0.2,Setosa
4.9,3.1,1.5,0.1,Setosa
5.4,3.7,1.5,0.2,Setosa
4.8,3.4,1.6,0.2,Setosa
4.8,3.1,1.4,0.1,Setosa
```

Open [AnalyzeIris.py](#) and solve the following parts. **While opening the files, please do not use any path specific to your computer. When you have the .csv and .py files in the same folder, you don't need to specify the path, you can directly open the file you want using its name.**

Part A: (15 pts)

Write a function called `read_iris(filename)` to read the data from the given csv file into a list of lists.

- The return value should be a list of lists such that:
 - Each list should be in the form:
[sepal_length, sepal_width, petal_length, petal_width, variety]
- Convert sepal_length, sepal_width, petal_length, petal_width to float. Variety should be string.
- Your matrix should not include the header you have in the csv file.

Partial output when the matrix is printed to console (full matrix is quite large):

```
[[5.1, 3.5, 1.4, 0.2, 'Setosa'], [4.9, 3.0, 1.4, 0.2, 'Setosa'], [4.7, 3.2, 1.3, 0.2, 'Setosa'], [4.6, 3.1, 1.5, 0.2, 'Setosa'], [5.0, 3.6, 1.4, 0.2, 'Setosa'], [5.4, 3.9, 1.7, 0.4, 'Setosa'], [4.6, 3.4, 1.4, 0.3, 'Setosa'], [5.0, 3.4, 1.5, 0.2, 'Setosa'], [4.4, 2.9, 1.4, 0.2, 'Setosa'], [4.9, 3.1, 1.5, 0.1, 'Setosa'], [5.4, 3.7, 1.5, 0.2, 'Setosa'], [4.8, 3.4, 1.6, 0.2, 'Setosa'], [4.8, 3.0, 1.4, 0.1, 'Setosa'], [4.3, 3.0, 1.1, 0.1, 'Setosa'], [5.8, 4.0, 1.2, 0.2, 'Setosa'], [5.7, 4.4, 1.5, 0.4, 'Setosa'], [5.4, 3.9, 1.3, 0.4, 'Setosa'], [5.1, 3.5, 1.4, 0.3, 'Setosa'], [5.7, 3.8, 1.7, 0.3, 'Setosa'], [5.1, 3.8, 1.5, 0.3, 'Setosa'], [5.4, 3.4, 1.7, 0.2, 'Setosa'], [5.1, 3.7, 1.5, 0.4, 'Setosa'], [4.6, 3.6, 1.0, 0.2, 'Setosa'], [5.1, 3.3, 1.4, 0.3, 'Setosa'], [5.2, 3.7, 1.4, 0.4, 'Setosa'], [5.2, 3.4, 1.4, 0.3, 'Setosa'], [5.2, 3.6, 1.3, 0.3, 'Setosa'], [5.4, 3.6, 1.5, 0.5, 'Setosa'], [5.5, 4.2, 1.4, 0.4, 'Setosa'], [5.7, 4.3, 1.5, 0.5, 'Setosa'], [5.8, 4.3, 1.5, 0.5, 'Setosa'], [5.9, 4.4, 1.4, 0.4, 'Setosa'], [6.0, 4.7, 1.5, 0.5, 'Setosa'], [6.1, 4.9, 1.4, 0.4, 'Setosa'], [6.4, 4.8, 1.5, 0.4, 'Setosa'], [6.5, 5.0, 1.4, 0.4, 'Setosa'], [6.7, 5.2, 1.4, 0.3, 'Setosa'], [6.9, 5.3, 1.5, 0.3, 'Setosa'], [7.0, 5.8, 1.6, 0.3, 'Setosa'], [7.1, 5.9, 1.6, 0.3, 'Setosa'], [7.4, 6.4, 1.6, 0.3, 'Setosa'], [7.7, 6.6, 1.6, 0.3, 'Setosa'], [7.9, 6.7, 1.7, 0.3, 'Setosa'], [8.0, 6.9, 1.6, 0.3, 'Setosa'], [8.1, 7.0, 1.7, 0.3, 'Setosa'], [8.3, 7.2, 1.7, 0.3, 'Setosa'], [8.4, 7.4, 1.7, 0.3, 'Setosa'], [8.5, 7.7, 1.8, 0.3, 'Setosa'], [8.6, 7.9, 1.8, 0.3, 'Setosa'], [8.7, 8.1, 1.9, 0.4, 'Setosa'], [8.9, 8.3, 1.8, 0.3, 'Setosa'], [9.0, 8.4, 1.9, 0.4, 'Setosa'], [9.4, 8.8, 1.8, 0.3, 'Setosa'], [9.5, 9.0, 1.9, 0.4, 'Setosa'], [9.7, 9.1, 1.8, 0.3, 'Setosa'], [9.8, 9.3, 1.9, 0.4, 'Setosa'], [10.0, 9.4, 1.8, 0.3, 'Setosa'], [10.3, 9.5, 1.9, 0.4, 'Setosa'], [10.5, 9.7, 1.9, 0.4, 'Setosa'], [10.6, 9.9, 2.0, 0.5, 'Setosa'], [10.8, 10.1, 2.0, 0.5, 'Setosa'], [10.9, 10.3, 2.0, 0.5, 'Setosa'], [11.0, 10.5, 2.0, 0.5, 'Setosa'], [11.3, 10.6, 2.0, 0.5, 'Setosa'], [11.5, 10.8, 2.0, 0.5, 'Setosa'], [11.6, 11.0, 2.1, 0.6, 'Setosa'], [11.8, 11.2, 2.1, 0.6, 'Setosa'], [11.9, 11.4, 2.1, 0.6, 'Setosa'], [12.0, 11.6, 2.1, 0.6, 'Setosa'], [12.1, 11.8, 2.2, 0.7, 'Setosa'], [12.3, 12.0, 2.2, 0.7, 'Setosa'], [12.4, 12.2, 2.2, 0.7, 'Setosa'], [12.5, 12.4, 2.2, 0.7, 'Setosa'], [12.7, 12.6, 2.3, 0.8, 'Setosa'], [12.9, 12.8, 2.3, 0.8, 'Setosa'], [13.0, 13.0, 2.3, 0.8, 'Setosa'], [13.3, 13.1, 2.3, 0.8, 'Setosa'], [13.5, 13.3, 2.3, 0.8, 'Setosa'], [13.6, 13.5, 2.4, 0.9, 'Setosa'], [13.8, 13.7, 2.4, 0.9, 'Setosa'], [13.9, 13.9, 2.4, 0.9, 'Setosa'], [14.0, 14.1, 2.4, 0.9, 'Setosa'], [14.3, 14.3, 2.5, 1.0, 'Setosa'], [14.5, 14.5, 2.5, 1.0, 'Setosa'], [14.6, 14.7, 2.5, 1.0, 'Setosa'], [14.8, 14.9, 2.5, 1.0, 'Setosa'], [15.0, 15.1, 2.6, 1.1, 'Setosa'], [15.3, 15.2, 2.6, 1.1, 'Setosa'], [15.5, 15.4, 2.6, 1.1, 'Setosa'], [15.6, 15.6, 2.7, 1.2, 'Setosa'], [15.8, 15.8, 2.7, 1.2, 'Setosa'], [15.9, 16.0, 2.7, 1.2, 'Setosa'], [16.0, 16.2, 2.7, 1.2, 'Setosa'], [16.3, 16.3, 2.8, 1.3, 'Setosa'], [16.5, 16.5, 2.8, 1.3, 'Setosa'], [16.6, 16.7, 2.8, 1.3, 'Setosa'], [16.8, 16.9, 2.8, 1.3, 'Setosa'], [17.0, 17.1, 2.9, 1.4, 'Setosa'], [17.3, 17.2, 2.9, 1.4, 'Setosa'], [17.5, 17.4, 2.9, 1.4, 'Setosa'], [17.6, 17.6, 3.0, 1.5, 'Setosa'], [17.8, 17.8, 3.0, 1.5, 'Setosa'], [17.9, 18.0, 3.0, 1.5, 'Setosa'], [18.0, 18.2, 3.0, 1.5, 'Setosa'], [18.3, 18.3, 3.1, 1.6, 'Setosa'], [18.5, 18.5, 3.1, 1.6, 'Setosa'], [18.6, 18.7, 3.1, 1.6, 'Setosa'], [18.8, 18.9, 3.1, 1.6, 'Setosa'], [19.0, 19.1, 3.2, 1.7, 'Setosa'], [19.3, 19.2, 3.2, 1.7, 'Setosa'], [19.5, 19.4, 3.2, 1.7, 'Setosa'], [19.6, 19.6, 3.3, 1.8, 'Setosa'], [19.8, 19.8, 3.3, 1.8, 'Setosa'], [19.9, 20.0, 3.3, 1.8, 'Setosa'], [20.0, 20.2, 3.3, 1.8, 'Setosa'], [20.3, 20.3, 3.4, 1.9, 'Setosa'], [20.5, 20.5, 3.4, 1.9, 'Setosa'], [20.6, 20.7, 3.4, 1.9, 'Setosa'], [20.8, 20.9, 3.4, 1.9, 'Setosa'], [21.0, 21.1, 3.5, 2.0, 'Setosa'], [21.3, 21.2, 3.5, 2.0, 'Setosa'], [21.5, 21.4, 3.5, 2.0, 'Setosa'], [21.6, 21.6, 3.6, 2.1, 'Setosa'], [21.8, 21.8, 3.6, 2.1, 'Setosa'], [21.9, 22.0, 3.6, 2.1, 'Setosa'], [22.0, 22.2, 3.6, 2.1, 'Setosa'], [22.3, 22.3, 3.7, 2.2, 'Setosa'], [22.5, 22.5, 3.7, 2.2, 'Setosa'], [22.6, 22.7, 3.7, 2.2, 'Setosa'], [22.8, 22.9, 3.7, 2.2, 'Setosa'], [23.0, 23.1, 3.8, 2.3, 'Setosa'], [23.3, 23.2, 3.8, 2.3, 'Setosa'], [23.5, 23.4, 3.8, 2.3, 'Setosa'], [23.6, 23.6, 3.9, 2.4, 'Setosa'], [23.8, 23.8, 3.9, 2.4, 'Setosa'], [23.9, 24.0, 3.9, 2.4, 'Setosa'], [24.0, 24.2, 3.9, 2.4, 'Setosa'], [24.3, 24.3, 4.0, 2.5, 'Setosa'], [24.5, 24.5, 4.0, 2.5, 'Setosa'], [24.6, 24.7, 4.0, 2.5, 'Setosa'], [24.8, 24.9, 4.0, 2.5, 'Setosa'], [25.0, 25.1, 4.1, 2.6, 'Setosa'], [25.3, 25.2, 4.1, 2.6, 'Setosa'], [25.5, 25.4, 4.1, 2.6, 'Setosa'], [25.6, 25.6, 4.2, 2.7, 'Setosa'], [25.8, 25.8, 4.2, 2.7, 'Setosa'], [25.9, 26.0, 4.2, 2.7, 'Setosa'], [26.0, 26.2, 4.2, 2.7, 'Setosa'], [26.3, 26.3, 4.3, 2.8, 'Setosa'], [26.5, 26.5, 4.3, 2.8, 'Setosa'], [26.6, 26.7, 4.3, 2.8, 'Setosa'], [26.8, 26.9, 4.3, 2.8, 'Setosa'], [27.0, 27.1, 4.4, 2.9, 'Setosa'], [27.3, 27.2, 4.4, 2.9, 'Setosa'], [27.5, 27.4, 4.4, 2.9, 'Setosa'], [27.6, 27.6, 4.5, 3.0, 'Setosa'], [27.8, 27.8, 4.5, 3.0, 'Setosa'], [27.9, 28.0, 4.5, 3.0, 'Setosa'], [28.0, 28.2, 4.5, 3.0, 'Setosa'], [28.3, 28.3, 4.6, 3.1, 'Setosa'], [28.5, 28.5, 4.6, 3.1, 'Setosa'], [28.6, 28.7, 4.6, 3.1, 'Set
```

Part B: (15 pts)

Write a function called `avg_versicolor(lst)` to calculate and record the average values for sepal length and petal length for 'Versicolor' species.

- Input `lst` should be the result you obtained from Part A.
- Let the average sepal length for Versicolor species be `avg_sepal_length` and the average petal length for Versicolor species be `avg_petal_length`. Your return value should be a dictionary including `'Versicolor'` as the key, and the tuple `(avg_sepal_length, avg_petal_length)` as the value associated with this key. Your `avg_sepal_length` and `avg_petal_length` values should be floats – do not round them.

Your output should be as follows:

```
{'Versicolor': (5.936, 4.26)}
```

Q3: Exception Log - 25 pts

In software systems, it is common to maintain a “log file” that records events, errors or exceptions that occur while the software is running. The log file is periodically reviewed by system administrators to check for abnormal behavior, unusual events, cyberattacks, etc.

In this question, you will implement code to maintain an “exception log file” to record exceptions that occur in a simple Python function.

Open `ExceptionLog.py` and find the function `divide_elementwise(a, b)`. This function:

- Takes as input two matrices `a` and `b` (in list of lists representation)
- Performs element-wise division: divide `a[i][j]` by `b[i][j]`
- A caveat: `a` and `b` originally contain string elements, therefore the strings are converted to floats by `divide_elementwise` before division is performed.

You may assume `a` and `b` are square matrices (number of rows = number of columns).

There are multiple exceptions that may occur in `divide_elementwise`. You should modify the `divide_elementwise` function that is given to you so that these exceptions are handled. Furthermore, you should ensure that the appropriate exception log messages are appended to a file called `exceptionlog.txt`.

Here are the potential exceptions and how you should handle them:

- A string element cannot be converted to float since it contains text (not a number).
 - Determine what exception/error will be raised.
 - Catch the exception corresponding to this particular problem.
 - When the exception is caught, you should append a one-line message to `exceptionlog.txt`: "Exception! Cannot convert string to float."
- A division-by-zero occurs.
 - Determine what exception/error will be raised.
 - Catch the exception corresponding to this particular problem.
 - When the exception is caught, you should append a one-line message to `exceptionlog.txt`: "Exception! Cannot divide by zero."
- The sizes of `a` and `b` are different, e.g., `a` is $N \times N$ matrix but `b` is $M \times M$ where $M \neq N$:
 - Explicitly check for this condition at the beginning of the function (before division is performed) and raise an exception.
 - Catch the exception corresponding to this problem.
 - When the exception is caught, you should append a one-line message to `exceptionlog.txt`: "Exception! Matrix sizes are different."

If `divide_elementwise` runs without any exceptions, it should append a one-line message to `exceptionlog.txt`: "divide_elementwise() ran without any exceptions."

After every execution of `divide_elementwise`, regardless of whether an exception occurred or not, it should always append a one-line message to `exceptionlog.txt`: "Exiting divide_elementwise() now."

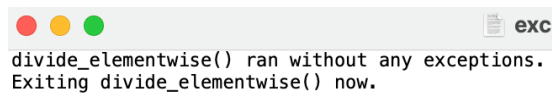
Rewrite `divide_elementwise` by keeping the original functionality but using a try-except structure to achieve the above exception handling behavior.

IMPORTANT: You must handle all exceptions and write your messages to `exceptionlog.txt` within `divide_elementwise`. Do not handle exceptions within the main() function.

IMPORTANT: In all cases, you must APPEND to `exceptionlog.txt`. Never overwrite existing information in `exceptionlog.txt`. For example, if you run the code without commenting out any

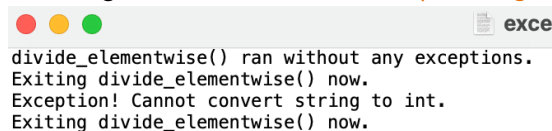
of the test cases or without changing the order of the test cases within main() function, your text file will be updated as follows after each case:

No errors



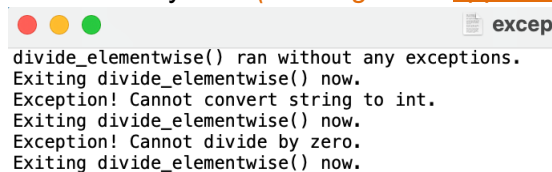
```
divide_elementwise() ran without any exceptions.  
Exiting divide_elementwise() now.
```

String to int conversion error (*message was appended to the previous .txt file*)



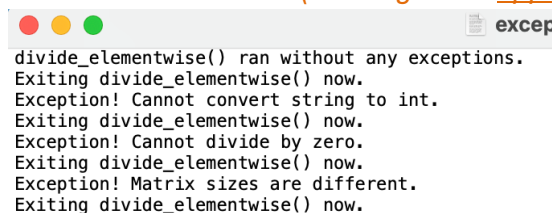
```
divide_elementwise() ran without any exceptions.  
Exiting divide_elementwise() now.  
Exception! Cannot convert string to int.  
Exiting divide_elementwise() now.
```

Division by zero (*message was appended to the previous .txt file*)



```
divide_elementwise() ran without any exceptions.  
Exiting divide_elementwise() now.  
Exception! Cannot convert string to int.  
Exiting divide_elementwise() now.  
Exception! Cannot divide by zero.  
Exiting divide_elementwise() now.
```

Different matrix size (*message was appended to the previous .txt file*)



```
divide_elementwise() ran without any exceptions.  
Exiting divide_elementwise() now.  
Exception! Cannot convert string to int.  
Exiting divide_elementwise() now.  
Exception! Cannot divide by zero.  
Exiting divide_elementwise() now.  
Exception! Matrix sizes are different.  
Exiting divide_elementwise() now.
```

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). DO NOT USE TURKISH CHARACTERS.** If you want, you can add helper functions.

When you are finished, compress your Hw3 folder containing all of your answers. The result should be a SINGLE compressed file (file extension must be .zip, or .rar). Upload this compressed file to Blackboard.

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

After you submit your Hw3, download it from Blackboard to make sure it is not corrupted and it has the latest version of your code. You are only going to be graded based on your Blackboard submission. **We will not accept homework via e-mail or other means.**

Happy coding! ☺