# CS 210
# Lab Week 7

## 1. Objectives

- Practice `csv.reader` input and output.
- Practice with more complex dictionaries.
- Practice function parameter passing -- understand what passing by object reference means.
- Continue practicing plotting with `matplotlib`.
- Practice using Coding Rooms and VSCode; feel free to share CR workspaces and work in pairs.
- Include your partner's name in all files if you work in pairs. Include the type and amount of collaboration between the two students.

## 2. Lab Exercises

### 2.1. [25 pts] Outcome Prediction

Predict the results of entering the following expressions into the Python Shell and indicate the resulting value type. Where predicted results diverge from actual results, determine why.

Take a moment to write your predictions before trying the following Python expressions in the shell. Make a copy of the file `lab-7-notes.xlsx` provided on Canvas and Coding Rooms, and write your answers there. Upload it back to Coding Rooms.

| Ex.# | Expression | Evaluates to value | of type |
|---|---|---|---|
| 1 | `[list(range(3)), list(range(3))]` | [[0, 1, 2], [0, 1, 2]] | list |
| 2 | `[[1, 2, 3], [4, 5, 6]][1][2]` | | |
| 3 | `[x % 2 for x in [0, 1, 2, 3, 4]]` | | |
| 4 | `my_list = [10, 20, [300, 400, [5000, 6000], 500], 30, 40]`<br>`my_list[2][2].append(7000)`<br>`my_list` | | |
| 5 | `[i for i in [5, 20, 15, 20, 25, 50, 20] if i != 20]` | | |
| 6 | `3 * [1, 2, 3]` | | |
| 7 | `3 * [[1, 2, 3]]` | | |
| 8 | `[x + y for x, y in zip(['Si', 'goo'], ['lly', 'se!'])]` | | |
| 9 | `{('Name', str): ['Bob', 'Lisa', 'Jill'], ('Age', int): [21, 19, 23]}[('Name', str)]` | | |

| Ex.# | Expression | Evaluates to value | of type |
|---|---|---|---|
| 10 | `{'Names': ['Bob', 'Lisa', 'Jill'], 'Ages': [21, 19, 23]}['Ages']` | | |
| 11 | `{'Names': ['Bob', 'Lisa', 'Jill'], 'Ages': [21, 19, 23]}['Ages'][0]` | | |
| 12 | `{'A': {'a': [1, 2, 3], 'b': [4, 5, 6]}, 'B': 'Bananas'}['B']` | | |
| 13 | `{'A': {'a': [1, 2, 3], 'b': [4, 5, 6]}, 'B': 'Bananas'}['A']` | | |
| 14 | `{'A' : {'a': [1, 2, 3], 'b': [4, 5, 6]}, 'B': 'Bananas'}['A']['b']` | | |
| 15 | `{key: [] for key in range(3)}` | | |
| 16 | `{key: [] for key in zip(['a', 'b', 'c'], range(3))}` | | |
| 17 | `{key: [] for key in tuple(zip(['Name', 'Age'], [str, int]))}` | | |
| 18 | `{key: val for key, val in zip(['ubi', 'ibi', 'puer'], ['where', 'there', 'boy'])}` | | |
| 19 | `list(enumerate(['John', 'Jane', 'Adam', 'Eva', 'Ashley']))` | | |
| 20 | `{i: len(j) for i, j in enumerate(['cat', 'dog', 'okapi'])}` | | |

## 2.2. [25 pts] Extract columns

The file `city-scores.csv` (you may download it from Coding Rooms or Canvas) contains a dataset of cities and their associated quality-of-life scores. Each city has six quality-of-life metric scores: `Housing`, `Cost_Of_Living`, `Commute`, `Safety`, `Healthcare`, and `Education`.

Use the `csv.reader` module to read the file and store it in `row_list`.

Recall that the first line of the file contains the column names. Extract the columns and store them in a list named `fields`.

Write a function `read_data(file_name)-> dict:` that accepts a file name and returns a dictionary containing the following structure:

```
{(<city value>, <country value>):
 {"Housing": <housing value>,
  "Cost_Of_Living": <cost_of_living value>,
  "Commute": <commute value>,
  "Safety": <safety value>,
  "Healthcare": <healthcare value>,
  "Education": <education value>
 }
}
```

Note that this structure is a dictionary whose keys are tuples (city, country) (yes, a dictionary key can be any object). Their associate values are a dictionary with all the city indices. The following is an example of the city of Yerevan:

```
>>> housing_dict[('Yerevan', 'Armenia')]
{'Housing': '9.6945',
 'Cost_of_Living': '9.431',
 'Commute': '5.64925',
 'Safety': '8.873',
 'Healthcare': '5.102',
 'Education': '0'}
```

You may use `for` loops, but this is your chance to practice dictionary comprehension!

Save your solution to the file `housing.py`.

### 2.3. [25 pts] Extract columns

Define the function `create_total_score(data_dict:dict)->None`. This function creates a new field named `total_score` for each city in `data_dict`. The values of `total_score` will be the sum of all the metric scores.

**Note**: We are **not** returning a new dictionary but modifying the existing one by *passing its reference* as a function parameter! We can only do this for *mutable* objects such as lists or dictionaries, where we change them "in place".

*Challenge Problem*: Print the top ten cities with the highest `total_score`.

### 2.4. [25 pts] Scatter Plot

Define a function `plot_data(data_dict: dict): -> None` that plots a scatterplot of the fields `'Housing'` vs. `'Cost_of_Living'` for all cities. Save an image of the plot in the file `housing.png` and upload it to Coding Rooms.

Hint: You need to get all the housing and cost_of_living scores of all cities as two lists.

## 3. What to submit

- Upload the file `housing.py` to Coding Rooms. This file must contain the final version of your implementation.
- Upload the file `lab-7-notes.xlsx` to Coding Rooms. This file must contain your answers to exercise 1.
- Upload the file `housing.png` to Coding Rooms. This file must contain an image of the plot you produced in exercise 2.c.