LEC 09

# CSE 122

# Nested Collections

Questions during Class?

Raise hand or send here

sli.do #cse122

## BEFORE WE START

***Talk to your neighbors:***
*What was the example you found for Creative Project 1 about how data set bias has caused harm to people?*

*Music: [Hunter/Miya's Playlist]*


Instructor: Hunter Schafer / Miya Natsuhara

TAs: Ajay, Andrew, Anson, Anthony, Audrey, Chloe, Colton, Connor, Elizabeth, Evelyn, Gaurav, Hilal, Hitesh, Jake, Jin, Joe, Joe, Karen, Kyler, Leon, Melissa, Noa, Parker, Poojitha, Samuel, Sara, Simon, Sravani, Tan, Vivek

# Agenda

- **Announcements**
- Review/Finish: mostFrequentStart
- Recap: Nested Collections
- Practice: Search Engine
- Images Debrief

# Announcements

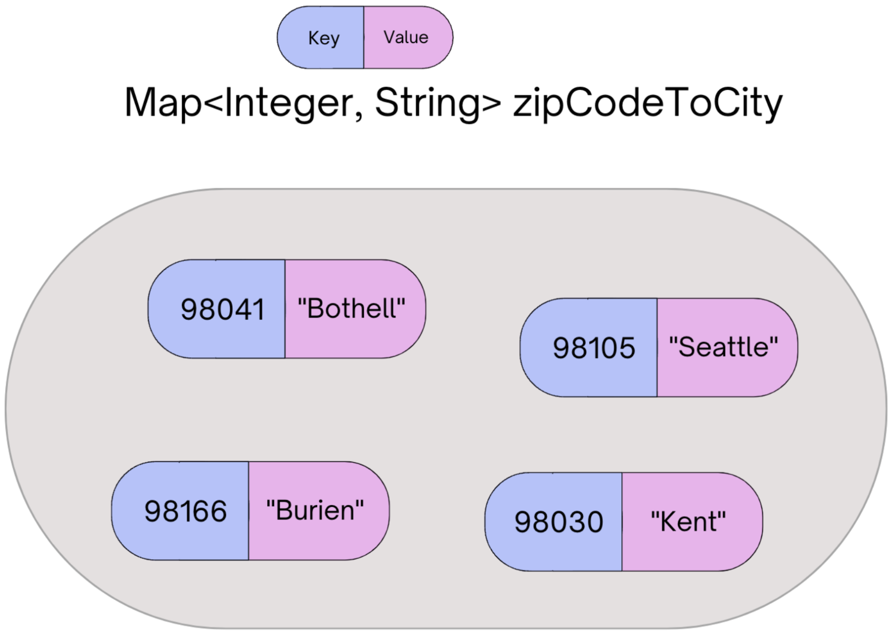
- Programming Assignment 2 will be out today and due next Thursday
  - Start early! Fairly complex assignment
  - Due Thursday 11/3 @ 11:59 pm
- Assignment Resubmission Form posted.
  - Due Tuesday 11/1 @ 11:59 pm
- Quiz Retake Form posted for the 11/1 Retakes
  - Due Sunday 10/30 @ 11:59 pm

# Agenda

- Announcements
- **Review/Finish: mostFrequentStart**
- Recap: Nested Collections
- Practice: Search Engine
- Images Debrief

# (Review) Map ADT

- Data structure to map keys to values
  - Keys can be any* type; Keys are unique
  - Values can be any type
- Example: Mapping nucleotides to counts!
- Operations
  - `put(key, value)`: Associate key to value
    - Overwrites duplicate keys
  - `get(key)`: Get value for key
  - `remove(key)`: Remove key/value pair



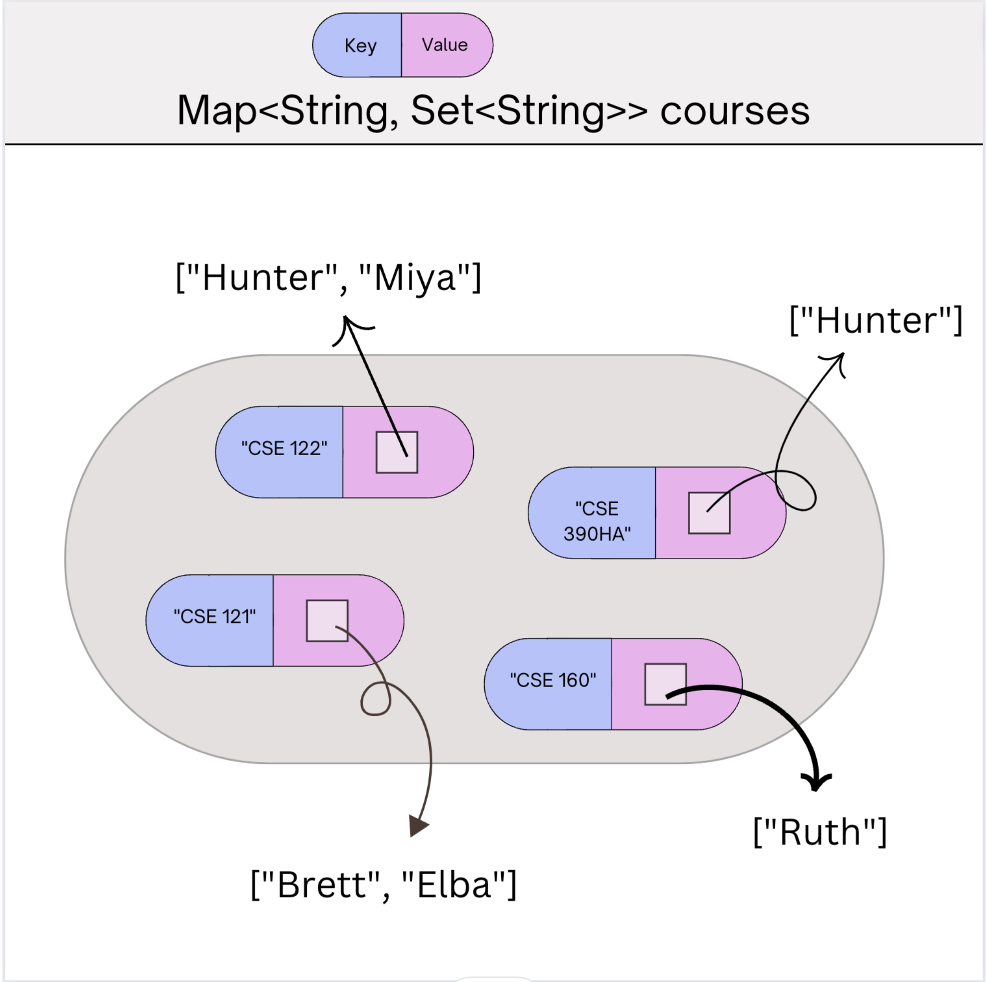

Same as Python's dict

# Agenda

- Announcements
- Review/Finish: mostFrequentStart
- **Recap: Nested Collections**
- Practice: Search Engine
- Images Debrief

# (PCM) Nested Collections

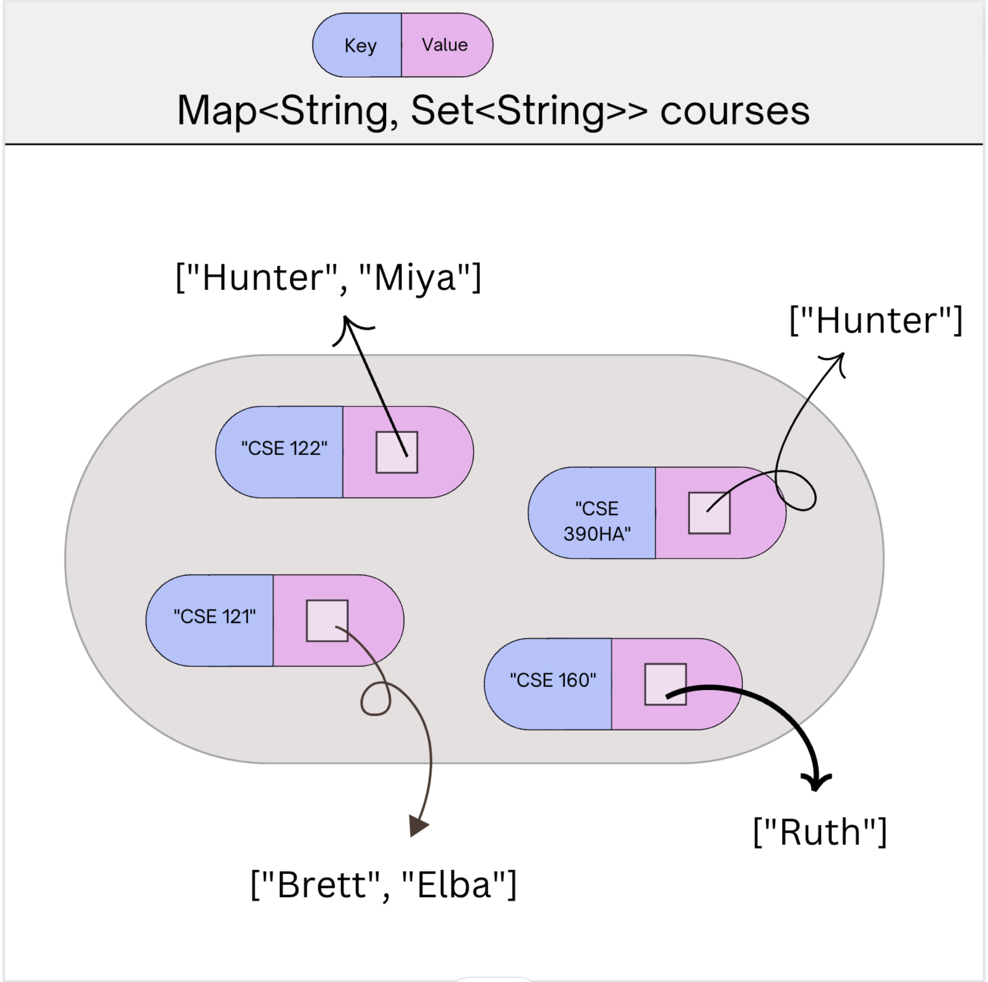
- The values inside a Map can be any type, including *data structures*
- Common examples:
  - Mapping Section -> Set of students in that section
  - Mapping Recipe -> Set of ingredients in that recipe
    - Or even Map<String, Map<String, Double>> for units!



# (PCM) Updating Nested Collections

The value inside the Map is a reference to the data structure!

- Think carefully about number of references to a particular object




```
courses.put("CSE 123", new HashSet<String>());
courses.get("CSE 123").add("Kasey");

Set<String> cse123 = courses.get("CSE 123");
cse123.add("Brett");
```

# Practice : Think

sli.do #cse122

## Suppose map had the following state. What would its state be after running this code?

```
map: {"KeyA"=[1, 2], "KeyB"=[3], "KeyC"=[4, 5, 6]}
```

```
Set<Integer> nums = map.get("KeyA");
nums.add(7);
map.put("KeyB", nums);
map.get("KeyA").add(8);
map.get("KeyB").add(9);
```

```
A. {"KeyA"=[1, 2],          "KeyB"=[1, 2, 7],       "KeyC"=[4, 5, 6]}
B. {"KeyA"=[1, 2, 8],       "KeyB"=[1, 2, 7, 9],    "KeyC"=[4, 5, 6]}
C. {"KeyA"=[1, 2, 7, 8],    "KeyB"=[1, 2, 7, 9],    "KeyC"=[4, 5, 6]}
D. {"KeyA"=[1, 2, 7, 8, 9], "KeyB"=[1, 2, 7, 8, 9], "KeyC"=[4, 5, 6]}
```

# Practice : Pair

sli.do #cse122

## Suppose map had the following state. What would its state be after running this code?

```
map: {”KeyA"=[1, 2], ”KeyB"=[3], ”KeyC"=[4, 5, 6]}
```

```
Set<Integer> nums = map.get("KeyA");
nums.add(7);
map.put("KeyB", nums);
map.get("KeyA").add(8);
map.get("KeyB").add(9);
```

```
A. {"KeyA"=[1, 2],          "KeyB"=[1, 2, 7],       "KeyC"=[4, 5, 6]}
B. {"KeyA"=[1, 2, 8],       "KeyB"=[1, 2, 7, 9],    "KeyC"=[4, 5, 6]}
C. {"KeyA"=[1, 2, 7, 8],    "KeyB"=[1, 2, 7, 9],    "KeyC"=[4, 5, 6]}
D. {"KeyA"=[1, 2, 7, 8, 9], "KeyB"=[1, 2, 7, 8, 9], "KeyC"=[4, 5, 6]}s
```

# Agenda

- Announcements
- Review/Finish: mostFrequentStart
- Recap: Nested Collections
- **Practice: Search Engine**
- Images Debrief

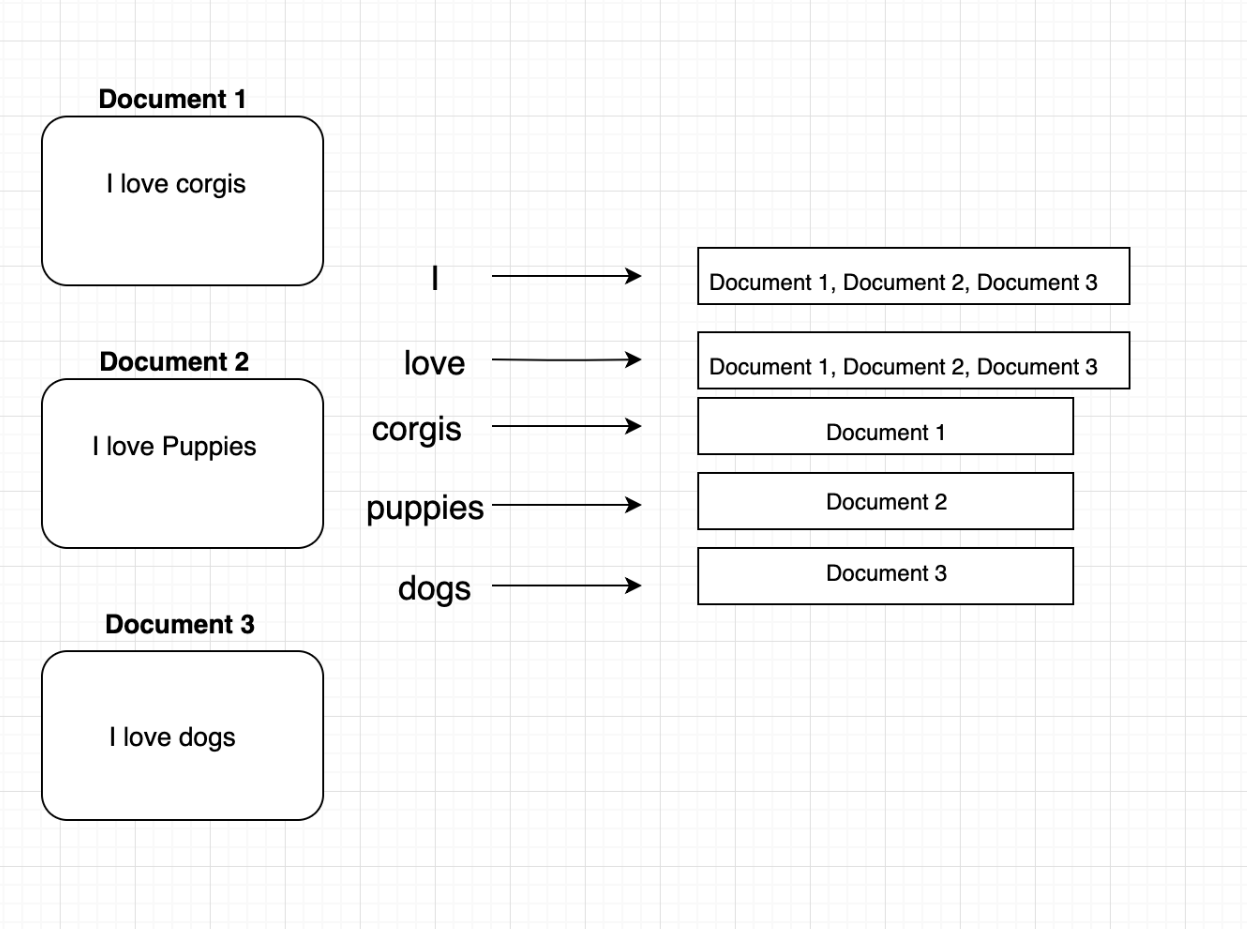
# Background: Search Engines

- A **search engine** receives a **query** and returns a set of relevant **documents.** Examples: Google.com, Mac Finder, more.
  - Queries often can have more
- A search engine involves two main components
  - An **index** to efficiently find the set of documents for a query
    - Will focus on “single word queries” for today’s example
  - A **ranking algorithm** to order the documents from most to least relevant
    - Not the focus of this example

- Goal: Precompute a data structure that helps find the relevant documents for a given query

# Inverted Index

- An **inverted index** is a Mapping from possible query words to the set of documents that contain that word
  - Answers the question: “What documents contain the word ‘corgis’?”



# (Optional) Ranking Results

- There is no one right way to define which documents are “most relevant” There are approximations, but make decisions about what relevance means
- Idea 1: Documents that have more hits of the query should come first
  - Pro: Simple
  - Con: Favors longer documents (query: “the dogs” will favor long documents with lots of “the”s)
- Idea 2: Weight query terms based on their “uniqueness”. Often use some sort of score for “Term Frequency – Inverse Document Frequency (TF-IDF)
  - Pro: Doesn’t put much weight on common words like “the”
  - Cons: Complex, many choices in how to compute that yield pretty different rankings
- Idea 3: Much more! Most companies keep their ranking algorithms very very secret ☺

# Agenda

- Announcements
- Review/Finish: mostFrequentStart
- Recap: Nested Collections
- Practice: Search Engine
- **Images Debrief**

# Data Bias

- Common Misconception: Models or Artificial Intelligence (AI) are somehow “less biased” or “more objective” than humans. **Not true.**
- The programs we use operate on real-world data, and will often reflect the biases that data contains
- Have to carefully consider the context and limitations of the data we gather. If the data an algorithm is built on is vastly different than the context in which it’s used, some pretty awful outcomes can happen

# Data Bias

In modern artificial intelligence, data rules. A.I. software is only as smart as the data used to train it. If there are many more white men than black women in the system, it will be worse at identifying the black women.

One widely used facial-recognition data set was estimated to be more than 75 percent male and more than 80 percent white, according to another research study.

**Color Matters in Computer Vision**

Facial recognition algorithms made by Microsoft, IBM and Face++ were more likely to misidentify the gender of black women than white men.

Gender was misidentified in **up to 1 percent of lighter-skinned males** in a set of 385 photos.

Gender was misidentified in **up to 7 percent of lighter-skinned females** in a set of

Gender was misidentified in **up to 12 percent of darker-skinned males** in a set of 318 photos.

Gender was misidentified in **35 percent of darker-skinned females** in a set of 271 photos.

# What to do?

- Obviously, ideal to have datasets that aren't biased in the first place.
  - But might not always be possible if we can't fix the sources of the bias in the real world...
- AI/Models aren't "neutral" or "more objective", they just quickly and automatically codify the status quo (and perpetuate biases)
  - Garbage in → Garbage out
- Lots of work going into how to de-bias models *even if* they are trained on biased data. Active area of research!
  - Key take-away: None of this comes "for free", requires hard work to fight bias
- Ask ourselves:
  - What biases might be present in my data?
  - What assumptions might I be making about who is using my program?
  - How can I write code to be more inclusive?
  - What happens *when (**not if**)* mistakes happen? Who potentially benefits and who is potentially harmed?