Nested Structures

CSE 160 Summer 2026

Questions?

Sli.do #cse160

Announcements

- Programming Practice 4 **due Sunday, July 26 at 11:59 PM**
- Homework Assignment 4 **due Monday, August 3 at 11:59 PM**
  - Longer and historically more challenging than past homework assignments
  - Two weeks to complete homework!
- **Homework 2 Feedback** now available on Gradescope
- Resubmission Cycle 2 now available, **due Wednesday July 29th at 11:59pm**
  - Homework 1 and Homework 2 are eligible
  - Fill out the Google Form!

Today's Roadmap

1. Nested (Data) Structures
2. Problem Solving Strategies
3. Examples :)

Jupyter Hub

Think Pair Share

Given the `olympics.txt` file containing each 2026 Winter Olympics discipline and its corresponding location:

Create a dictionary where **each *location* is a key**. The value for a location should be a **list** of all Olympic disciplines taking place at the location.

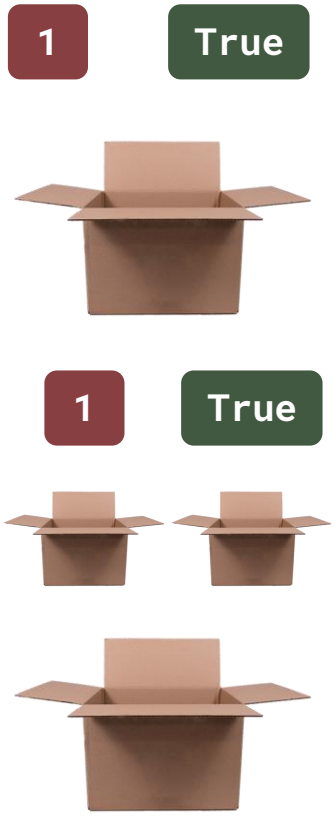
```
olympics_dict["Milan"] →

["Ice Hockey", "Speed Skating",
 "Figure Skating"]
```

```
Ice Hockey: Milan
Cross-Country Skiing: Tesero
Speed Skating: Milan
Alpine Skiing: Cortina
Freestyle Skiing: Livigno
Figure Skating: Milan
Ski Jumping: Predazzo
Ski Mountaineering: Bormi
Snowboarding: Livigno
Biathlon: Anterselva
```

`olympics.txt`

Nested Structures



Data structures like **lists** and **dictionaries** store data in a standardized format.

- We can take advantage of the **standardization** when working with this data
  - e.g. **list** elements can be accessed using indices, **dictionary** entries have keys
- We can **traverse** (iterate) through these structures, and **update** individual values

***Nested* data structures** involve storing data structures within other data structures.

- We can take advantage of the **hierarchy** *and* **standardization** of nested structures
- We need to understand and navigate this **hierarchy** to **traverse**, and **update** values

Previously on *CSE 160*…

Lists

`food_lst`

```
[["Agua Verde", "Montlake", 4.3],
 ["Bangrak", "Belltown", 4.8],
 ["Crackle Mi", "Ballard", 4.4]]
```

```
food_lst[1][2]
```

Dictionaries

`food_dict`

```
{"Agua Verde": {"Neighborhood": "Montlake",
                "Rating": 4.3},
 "Bangrak": {"Neighborhood": "Belltown",
             "Rating": 4.8},
 "Crackle Mi": {"Neighborhood": "Ballard",
                "Rating": 4.4}}
```

```
food_dict["Bangrak"]["Rating"]
```

Data Structure Decisions

Lists

`food_lst`

```
[["Agua Verde", "Montlake", 4.3],
 ["Bangrak", "Belltown", 4.8],
 ["Crackle Mi", "Ballard", 4.4]]
```

Dictionaries

`food_dict`

```
{"Agua Verde": {"Neighborhood": "Montlake",
                "Rating": 4.3},
 "Bangrak": {"Neighborhood": "Belltown",
             "Rating": 4.8},
 "Crackle Mi": {"Neighborhood": "Ballard",
                "Rating": 4.4}}
```

Which to use? **It depends.**

Problem Solving Strategies

You reach a problem and you are stumped. What now?

- **Solve a Subproblem**
  - Is there a smaller problem I can solve? (with loops, work inside out!)
- **Iterative Development**
  - Can I start by solving a different problem that is easier?
- **Inputs/Outputs + Debugging**
  - How do the inputs relate to the outputs?

Example 1: Nest-maxing

You are given a **triply-nested list** (`pixel_grid`) representing a grid of color pixels (each with **Red**, **Green**, and **Blue** values).

```
pixel_grid = [[[128, 50, 20], [50, 50, 60]],
              [[75, 67, 90], [42, 240, 100]]]
```

Your Task

For each pixel, set all three color values to the highest of the pixel's original RGB values.

```
modified_pixel_grid = [[[128, 128, 128], [60, 60, 60]],
                       [[90, 90, 90], [240, 240, 240]]]
```

Example 1: Nest-maxing

You are giv
grid of colo

Your Task

For each pi RGB values.

mod

```
[[90, 90, 90], [240, 240, 240]]]
```

PAUSE

Let's make a plan!

1. **Understand Inputs → Outputs**
2. **Solve a Simpler Problem**
3. **Increase Complexity**

Example 1: Nest-maxing

1. Understand Inputs → Outputs

You are given a **triply-nested list** (`pixel_grid`) representing a grid of color pixels (each with **Red**, **Green**, and **Blue** values).

```
pixel_grid = [[[128, 50, 20], [50, 50, 60]],
              [[75, 67, 90], [42, 240, 100]]]
```

TAKEAWAY: Each innermost list is a single pixel (RGB values)

Your Task

For each pixel, set all three color values to the highest of the pixel's original RGB values.

```
modified_pixel_grid = [[[128, 128, 128], [60, 60, 60]],
                       [[90, 90, 90], [240, 240, 240]]]
```

Example 1: Nest-maxing (Simpler Problem)

You are given a **triply-nested list** (`pixel_grid`) representing a grid of color pixels (each with **Red**, **Green**, and **Blue** values).

2. Solve a Simpler Problem

```
pixel_grid = [[[128, 50, 20], [50, 50, 60]],
              [[75, 67, 90], [42, 240, 100]]]
```

Your Task

100

For each pixel, set all three color values to ~~the highest of the pixel's original RGB values~~.

```
modified_pixel_grid = [[[100, 100, 100], [100, 100, 100]],
                       [[100, 100, 100], [100, 100, 100]]]
```

Example 1: Nest-maxing (Complexity)

You are given a **triply-nested list** (`pixel_grid`) representing a grid of color pixels (each with **Red**, **Green**, and **Blue** values).

2. Solve a Simpler Problem

```
pixel_grid = [[[128, 50, 20], [50, 50, 60]],
              [[75, 67, 90], [42, 240, 100]]]
```

3. Increase Complexity

Your Task

for each pixel, find highest value

For each pixel, set all three color values to the highest of the pixel's original RGB values.

```
modified_pixel_grid = [[[128, 128, 128], [60, 60, 60]],
                       [[90, 90, 90], [240, 240, 240]]]
```

Example 2: Structured Strings

Instead of analyzing DNA (A, T, C, G) sequences like in HW2, now you are given an **RNA** (A, U, C, G) sequence as an input string. **Each sequence of three nucleotides could translate to an amino acid.**

"AUG" - "Methionine"
"UGC" - "Crysteine"
"UCU" - "Serine"

Your Task

Use a **for loop**, **range**, and **string slicing** to "translate" an RNA sequence to amino acids.

```
"AUGCUCAUG"  ⟶  ["Methionine", "Methionine"]

"ACCUUUAUGAUUUGCUCUUUUUGCUCU"  ⟶  ["Methionine", "Crysteine",
                                   "Serine", "Crysteine", "Serine"]
```

Example 2: Structured Strings

1. Understand Inputs → Outputs

Instead of analyzing DNA (A, T, C, G) sequences like in HW2, now you are given an **RNA** (A, U, C, G) sequence as an input string. **Each sequence of three nucleotides could translate to an amino acid.**

"AUG" - "Methionine"
"UGC" - "Crysteine"
"UCU" - "Serine"

Your Task

Use a **for loop**, **range**, and **string slicing** to "translate" an RNA sequence to amino acids.

"AUGCUCAUG" → `["Methionine", "Methionine"]`

"ACCUUUAUGAUUUGCUCUUUUGCUCU" → `["Methionine", "Crysteine", "Serine", "Crysteine", "Serine"]`

TAKEAWAY: Move in steps of 3

Example 2: Structured Strings (Subproblem 1)

Instead of analyzing DNA (A, T, C, G) sequences like in HW2, now you are given an **RNA** (A, U, C, G) sequence as an input string. **Each sequence of three nucleotides could translate to an amino acid.**

2. Subproblem 1:
RNA sequence to amino acid

"**AUG**" - "**Methionine**"
"**UGC**" - "**Crysteine**"
"**UCU**" - "**Serine**"

Your Task

Use a **for loop**, **range**, and **string slicing** to "translate" an RNA sequence to amino acids.

Example 2: Structured Strings (Subproblem 2)

Instead of analyzing DNA (A, T, C, G) sequences like in HW2, now you are given an **RNA** (A, U, C, G) sequence as an input string. **Each sequence of three nucleotides could translate to an amino acid.**

"AUG" - "Methionine"
"UGC" - "Crysteine"
"UCU" - "Serine"

Your Task

Use a **for loop**, **range**, and **string slicing** to "translate" an RNA sequence to amino acids.

2. Subproblem 1:
RNA sequence to amino acid

3. Subproblem 2:
Iterate over RNA in steps of 3

TAKEAWAY: Move in steps of 3