More Lists

CSE 160 Summer 2026

Questions?

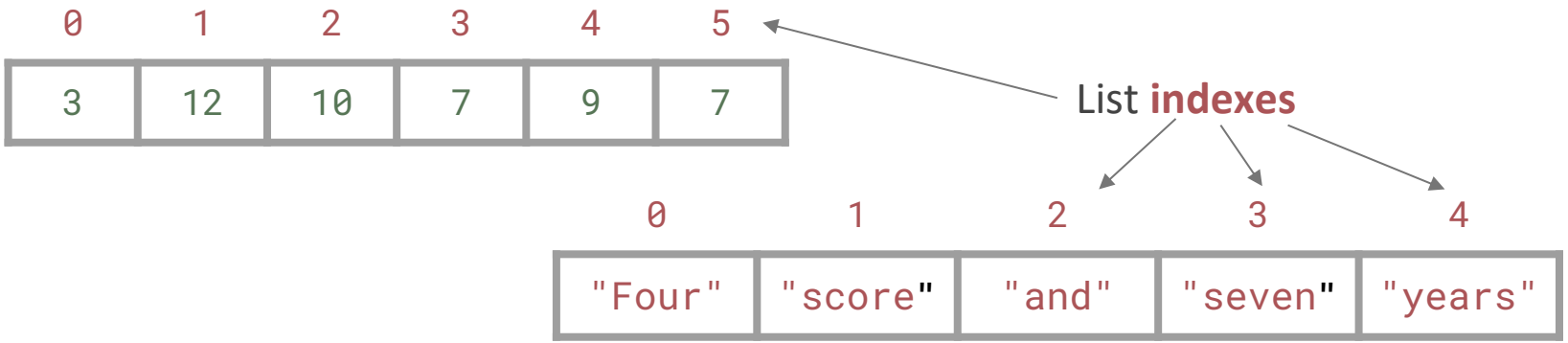
Sli.do #cse160

Announcements

- **Written Check-in 3** due Friday, July 10th at 11:59PM
- **Programming Practice 2** due Sunday, July 12th at 11:59PM
- **Homework Assignment 2** due Monday, July 13th at 11:59 PM
- **Practice Midterms** now available on the course website
    - We have not gone through all the material, so don't worry if you can't do some of the problems!
- Grades for Homework 0, Programming Practice 1, Written Check-in 1 and 2, and should now be available on Gradescope (read your feedback on Gradescope)

What is a List?

- A list is an ordered sequence of values
  - A list of integers: `[3, 12, 10, 7, 9, 7]`
  - A list of strings: `["Four", "score", "and", "seven", "years"]`
- Each value has an **index**
  - Indexing is zero-based (counting starts with zero)

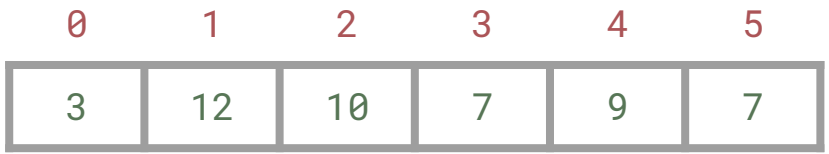


Rearrange elements in a list

`list_name.sort()` Sorts the elements in **`list_name`** "in place". Modifies the original list.

`list_name.reverse()` Reverses the order of elements in **`list_name`** "in place". Modifies the original list.

```
ticket_buyers = [3, 12, 10, 7, 9, 7]
ticket_buyers.sort()
ticket_buyers.reverse()
```



Note: **`sort`** and **`reverse`** return **None**

Querying Lists Summary

- What is the length of a list? `len(my_list)`
- Refer to a single element of a list `my_list[3]`
- Refer to a "slice" of a list `my_list[2:5]`
- Is a value in a list? `x in my_list`
- Where is a value in a list? `my_list.index(x)`
- How many of a value are in a list? `my_list.count(x)`

Modifying Lists Summary

- Add an element to the end `my_list.append(x)`
- Add each element from L to the end `my_list.extend(L)`
- Insert element x at index i `my_list.insert(i, x)`
- Remove first occurrence of x `my_list.remove(x)`
- Remove and return element at index I `my_list.pop(i)`
- Replace value at index i `my_list[i] = x`
- Sort the list `my_list.sort()`
- Reverse the list `my_list.reverse()`

List Modification Examples

Python Tutor

```
lst = [10, 12, 23, 54, 15]
lst.append(7)
lst.extend([8, 9, 3])
lst.insert(2, 2.75)
lst.remove(3)
print(lst.pop())
print(lst.pop(4))
lst[1:5] = [20, 21, 22]
lst2 = [4, 6, 8, 2, 0]
lst2.sort()
lst2.reverse()
lst3 = lst2
lst4 = lst2[:]
lst2[-1]= 17
```

Today's Roadmap

1. Modifying Lists
2. Loops and Lists
3. Nested Lists

Jupyter Hub

List Modification Examples

Python Tutor

```
lst = [10, 12, 23, 54, 15]
lst.append(7)
lst.extend([8, 9, 3])
lst.insert(2, 2.75)
lst.remove(3)
print(lst.pop())
print(lst.pop(4))
lst[1:5] = [20, 21, 22]
lst2 = [4, 6, 8, 2, 0]
lst2.sort()
lst2.reverse()
lst3 = lst2              # Creates a reference to lst2
lst4 = lst2[:]           # Creates a copy of lst2
lst2[-1]= 17
```

Think Pair Share:

What will convert a into `[1, 2, 3, 4, 5]`?

```
a = [1, 3, 5]
```

A. `a.insert(1, 2)`
   `a.insert(2, 4)`

A. `a[1:2] = [2, 3, 4]`

A. `a.extend([2, 4])`

A. `a[1] = 2`
   `a[3] = 4`

Sli.do #cse160

Exercise: list lookup

Jupyter Hub

```
def my_index(lst, value):
    """Return the position of the first occurrence
    of value in the list lst. Return None if value
    does not appear in lst."""


gettysburg = ["four", "score", "and", "seven", "years", "ago"]
assert my_index(gettysburg, "and") == 2
print(my_index(gettysburg, "years"))  # Should return 4
```

Exercise: list lookup (for-each loop)

```
def my_index(lst, value):
    """Return the position of the first occurrence
    of value in the list lst. Return None if value
    does not appear in lst."""
    pos = 0
    # Loop over elements in lst
    for element in lst:
        # If find an element equal to value, return its position
        if element == value:
            return pos
        pos = pos + 1
    return None
```

Exercise: list lookup (using indexing)

```
def my_index(lst, value):
    """Return the position of the first occurrence
    of value in the list lst. Return None if value
    does not appear in lst."""
    # Loop over elements in lst
    for pos in range(0, len(lst)):
        # If find an element equal to value, return its position
        if lst[pos] == value:
            return pos
    return None
```

Nested Lists

- Lists so far have contained elements such as integers, strings, etc.
- A **nested list** is simply a list whose elements are other lists

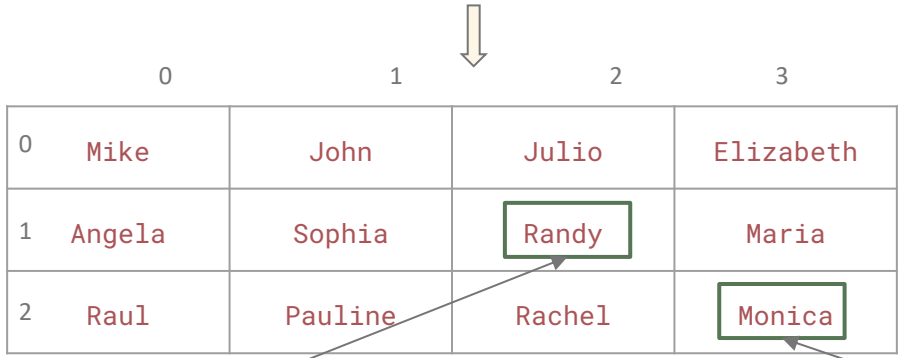
```
seating_chart = [["Mike", "John", "Julio", "Elizabeth"],
                 ["Angela", "Sophia", "Randy", "Maria"],
                 ["Raul", "Pauline", "Rachel", "Monica"]]
```

```
seating_chart[1]
```

```
seating_chart[2][3]
```

Visualizing Nested Lists using Grids

```
seating_chart = [["Mike", "John", "Julio", "Elizabeth"],
                 ["Angela", "Sophia", "Randy", "Maria"],
                 ["Raul", "Pauline", "Rachel", "Monica"]]
```



	0	1	2	3
0	Mike	John	Julio	Elizabeth
1	Angela	Sophia	Randy	Maria
2	Raul	Pauline	Rachel	Monica

Row: **1**
Column: **2**

→ `seating_chart[1][2]`

Row: **2**
Column: **3**

→ `seating_chart[2][3]`

Think Pair Share

Which of these statements will result in an error?

```
chart = [["Mike", "John", "Julio", "Elizabeth"],
         ["Angela", "Sophia", "Randy", "Maria"],
         ["Raul", "Pauline", "Rachel", "Monica"]]
```

A. `print(chart[1])`

B. `print(chart[1][1])`

C. `print(chart[2][4])`

A. `print(chart[2][3][1])`

B. None of the above

Sli.do #cse160

Traversing through a Nested List

```
seating_chart = [["Mike", "John", "Julio", "Elizabeth"],
                 ["Angela", "Sophia", "Randy", "Maria"],
                 ["Raul", "Pauline", "Rachel", "Monica"]]
```

How do you traverse this nested list to print out each name one at a time?

Using a for-each loop

```
for row in seating_chart:
    for student in row:
        print(student)
```

Using indexing

```
for ri in range(len(seating_chart)):
    for ci in range(len(seating_chart[ri])):
        print(seating_chart[ri][ci])
```

A list of lists of lists?

```
letters = [[["a","b"], ["c", "d"], ["e", "f"]],
           [["g","h"], ["i", "j"], ["k", "l"]]]
print(letters[0])
print(letters[0][2])
print(letters[1][2][0])
```