Lists

CSE 160 Summer 2026

Questions?

Sli.do #cse160

Announcements

- **Programming Practice 2** due Sunday, July 12th at 11:59PM
- **Homework Assignment 2** due Monday, July 13th at 11:59 PM
  - Processing DNA Files!
  - You should have everything you need to complete HW 2!
- Section tomorrow!
  - Written Check-in 2 due Friday, July 10th at 11:59PM
- If you are not seeing new files in JupyterHub, **restart your server.**
  - File → Hub Control Panel → Stop My Server
  - Start My Server

Python Building Blocks so Far

- Print Statements
- Expressions
- Assignment Statements
- Loops
- Conditionals (aka "if statements")
- Functions

Ways to Store Data so Far

- **Individual values** of type: int, float, string, bool can be assigned to variables:

```
x = 5        y = 2.5          z = "hi"    w = True
```

- What if I have **multiple values**?
  - HW2: examine the individual characters in a string of DNA
    - `"GTGGGGGTGATGTCCACGAT"`
  - We have also seen **Lists** that contain multiple values

Loop Example 3: Lists

this is a list of strings!

```
for x in ["Hi", "Ciao", "Hola"]:
    print(x)
```

Output

```
Hi
Ciao
Hola
```

Example: Courses

this is a list of strings!

```
for dept in ["CSE", "BIO", "INFO"]:
    for num in [180, 331, 480]:
        print(dept, num)
```

this is a list of integers!

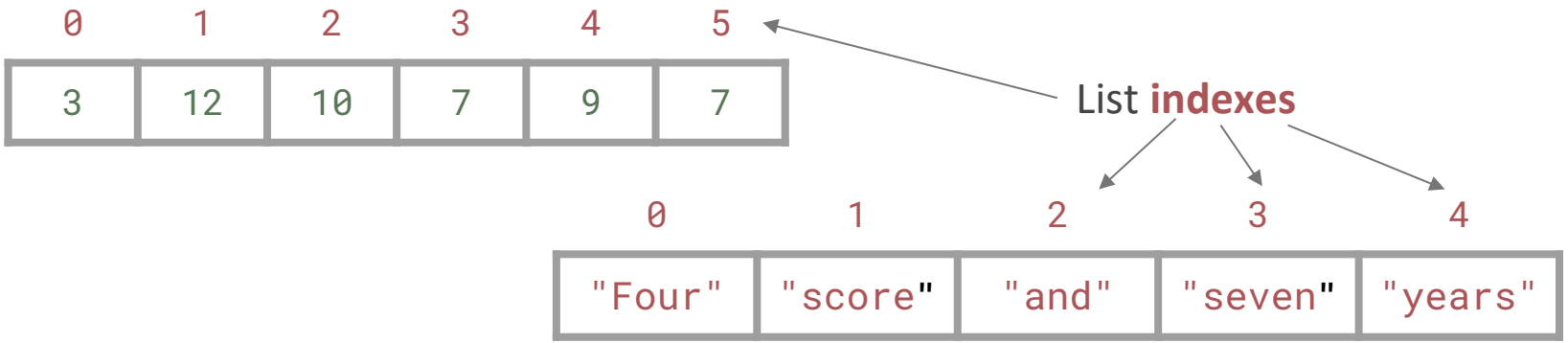
Today's Roadmap

1. What is a List?
2. Creating Lists
3. Querying Lists
4. Modifying Lists

Jupyter Hub

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)



Creating Lists in Python

Python Tutor

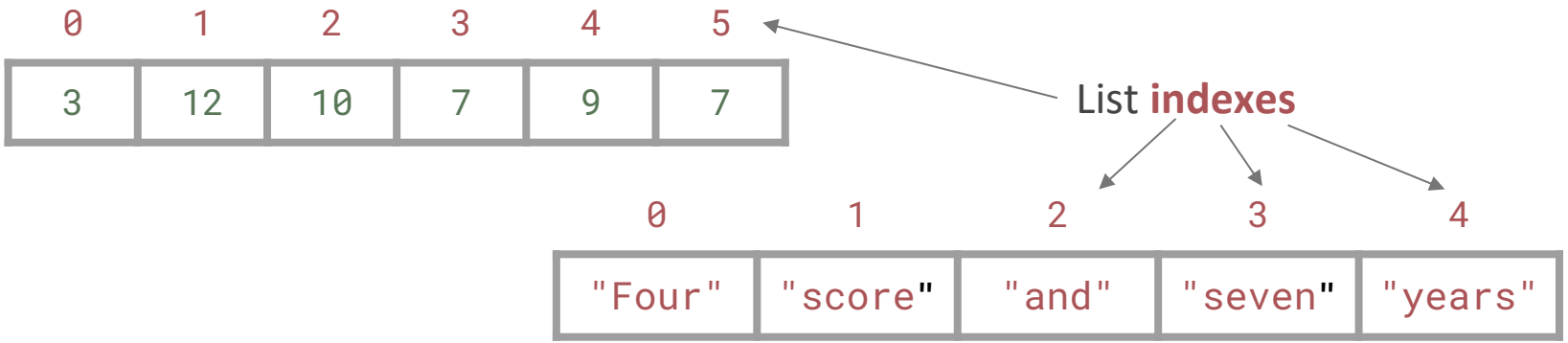
```
a = [3, 1, 2 * 2, 1, 10 / 2, 10 - 1]
b = [5, 3, "hi"]
c = [4, "a", a]
d = [[1, 2], [3, 4], [5, 6]]
e = [ ]   # An empty list
```

Note: Elements in a list do not have to be of the same type!

What is the length of a list?

The `len()` function, returns the length of a list.

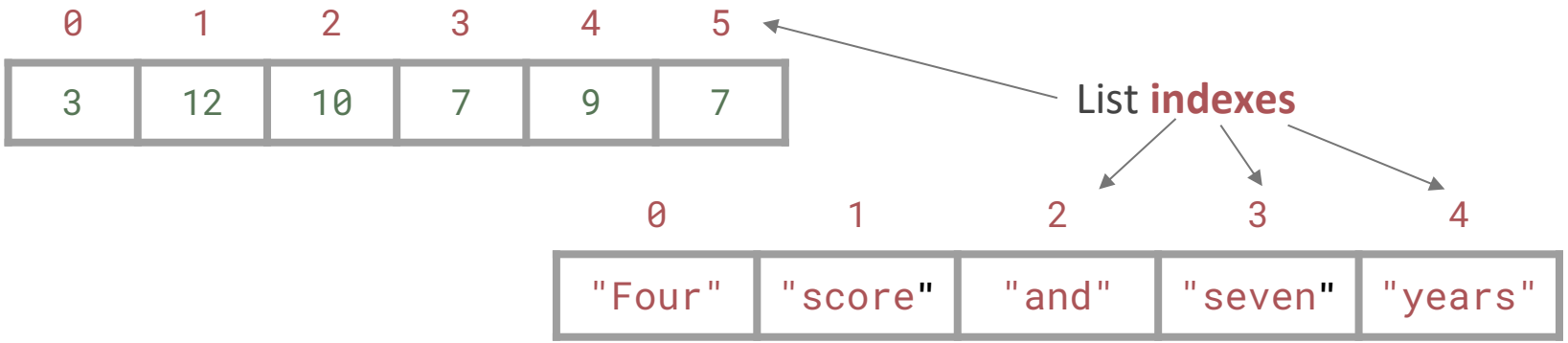
```
ticket_buyers = [3, 12, 10, 7, 9, 7]
important_document = ["Four", "score", "and", "seven", "years"]
print(len(ticket_buyers))                # Prints 6
print(len(important_document))  # Prints 5
```



Referring to a single element of a list

`list_name[index]` is an expression that evaluates to a single element of a list, stored at that index.

```
ticket_buyers = [3, 12, 10, 7, 9, 7]
important_document = ["Four", "score", "and", "seven", "years"]
print(ticket_buyers[1])                    # Prints 12
print(important_document[4])    # Prints "years"
```

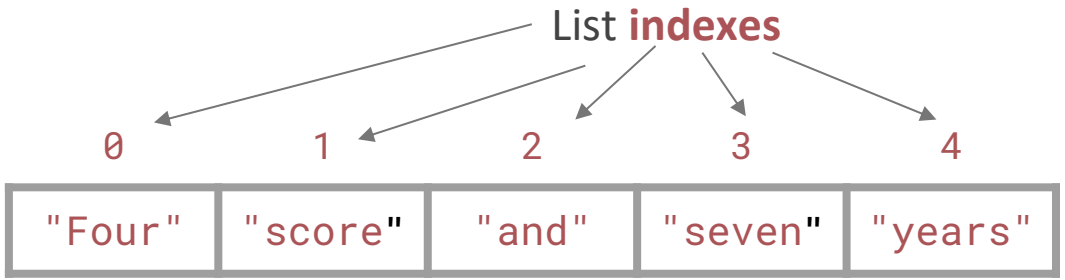


Referring to a "slice" of a list

`list_name[start:end]` is an expression that evaluates to a "slice" or a sublist of a list.

- The sublist begins with the value stored at index **`start`** and ends with the value stored at index **`end - 1`**

```
important_document = ["Four", "score", "and", "seven", "years"]
print(important_document[2:4])  # Prints ["and", "seven"]
print(important_document[0:3])  # Prints ["Four", "score", "and"]
```

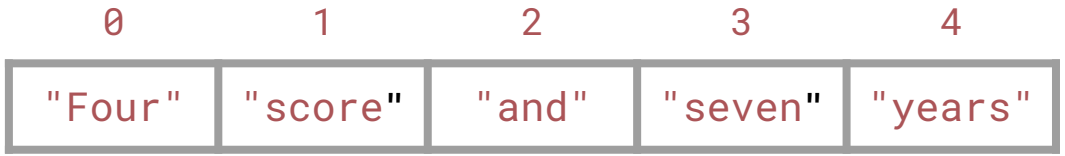


List "slicing" details

`list_name[start:end]` is an expression that evaluates to a "slice" or a sublist of a list.

- The sublist begins with the value stored at index **`start`** and ends with the value stored at index **`end - 1`**
- If **`start`** is omitted, it defaults to **0**
- If **`end`** is omitted, it defaults to **`len()`** of the list
- If both are omitted, it evaluates to the whole list

```
important_document = ["Four", "score", "and", "seven", "years"]
print(important_document[:3])   # Prints ["Four", "score", "and"]
print(important_document[3:])   # Prints ["seven", "years"]
```

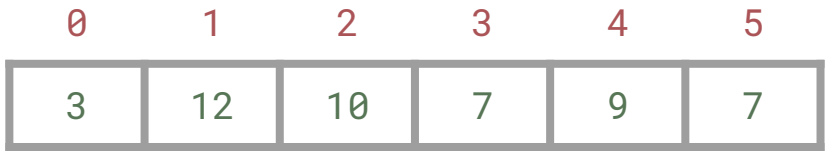


List Indexing and Slicing Examples

Python Tutor

```
a = [3, 12, 10, 7, 9, 7]
print(a[0])
print(a[5])
print(a[6])  # IndexError: list index out of range
print(a[-1]) # last element in list
print(a[-2]) # next to last element

print(a[1:3])
print(a[2:5])
print(a[:])
print(a[0:len(a)])
```



Think Pair Share:

What Python code will print: 9 4 7

```
a = [2, 7, 3, 9, 4]
```

A. `print(a[4], a[5], a[2])`

B. `print(a[3], a[-1], a[1])`

C. `print(a[4:6], a[2])`

D. `print(a[9], a[4], a[7])`

E. `print(a[3], a[5], a[1])`

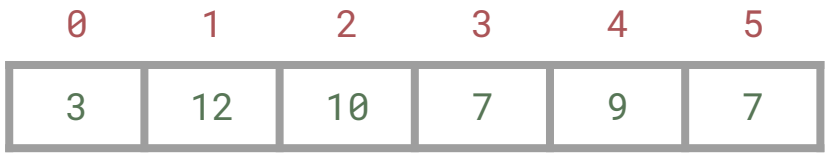
Sli.do #cse160

Is a value in a list?

`x in list_name` is a boolean expression that evaluates to **True** if **`x`** is found in **`list_name`**.

`x not in list_name` is a boolean expression that evaluates to **True** if **`x`** is ***not*** found in **`list_name`**.

```
ticket_buyers = [3, 12, 10, 7, 9, 7]
print(10 in ticket_buyers)              # Prints True
print(2 in ticket_buyers)               # Prints False
print(4 not in ticket_buyers)   # Prints True
```

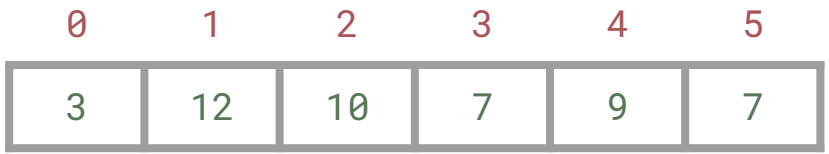


Where is a value in a list?

`list_name.index(x)` returns the integer index of the first occurrence of **`x`** in **`list_name`**

Note: It is an Error if **`x`** is ***not*** found in **`list_name`**.

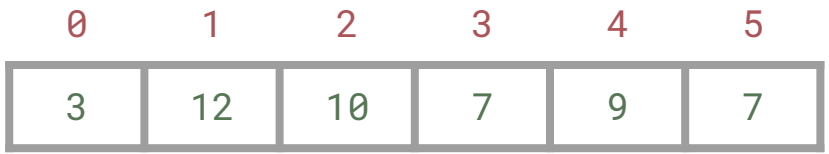
```
ticket_buyers = [3, 12, 10, 7, 9, 7]
print(ticket_buyers.index(10))  # Prints 2
print(ticket_buyers.index(7))   # Prints 3
print(ticket_buyers.index(4))   # ValueError: 4 is not in list
```



How many of a value are in a list?

`list_name.count(x)` returns the number of times **x** occurs in **list_name**

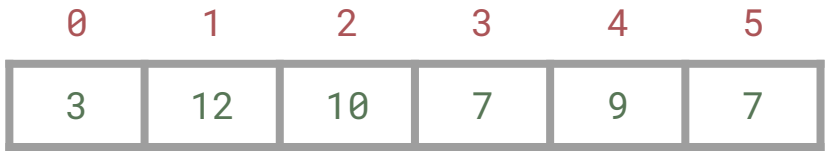
```
ticket_buyers = [3, 12, 10, 7, 9, 7]
print(ticket_buyers.count(10))  # Prints 1
print(ticket_buyers.count(7))   # Prints 2
print(ticket_buyers.count(4))   # Prints 0
```



List Querying Examples

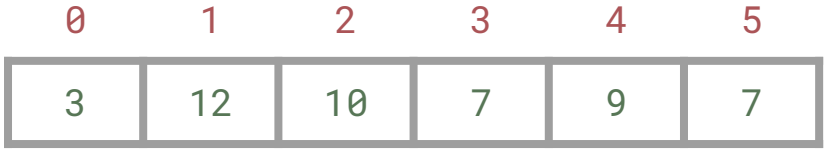
Python Tutor

```
a = [3, 12, 10, 7, 9, 7]
print(9 in a)
print(16 in a)
print(a.index(7))
print(a.index(16))
print(a.count(7))
print(a.count(16))
```



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)`

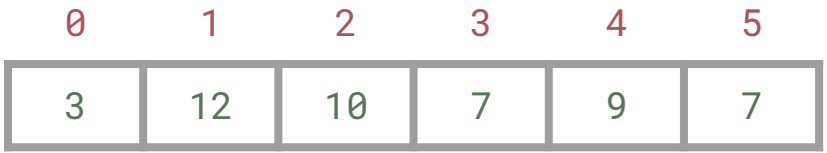


Today's Roadmap

1. ~~What is a List?~~
2. ~~Creating Lists~~
3. ~~Querying Lists~~
4. Modifying Lists

Ways to modify a list

- Add elements
- Remove elements
- Replace elements
- Rearrange elements

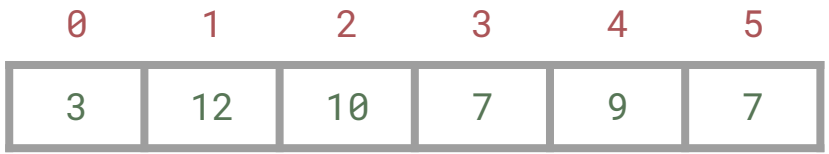


Adding elements to the end of a list

`list_name.append(x)` Adds item **`x`** at the end of **`list_name`**.

`list_name.extend(L)` Adds each of the elements in the list **`L`** to the end of **`list_name`**.

```
ticket_buyers = [3, 12, 10, 7, 9, 7]
ticket_buyers.append(34)
ticket_buyers.extend([14, 8])
```



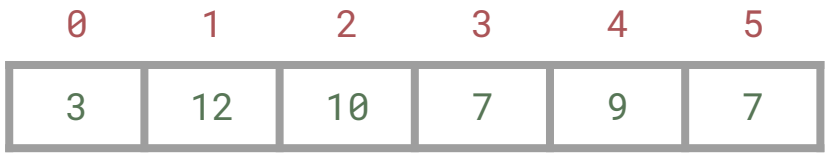
Note: **append** and **extend** return **None**

Inserting elements into a list

`list_name.insert(i, x)` Inserts item **`x`** at index **`i`** of **`list_name`**.

Note: **`x`** does not replace the value at index **`i`**, all values are shifted to the right (and the length of the list grows by 1).

```
ticket_buyers = [3, 12, 10, 7, 9, 7]
ticket_buyers.insert(0, 34)     # Insert at the beginning of list
ticket_buyers.insert(2, 8)
```



Note: **`insert`** returns **`None`**

Adding Elements Examples

Python Tutor

```
lst = [1, 2, 3, 4]
lst.append(5)
lst.extend([6, 7, 8])
lst.insert(3, 3.5)
```

Think Pair Share

What is **printed** when this program runs?

A 4

B 5

C 3

D `[4, 6]`

E `IndexError: list index out of range`

```
lst = [1, 3, 5]
lst.insert(2, [4, 6])
print(lst[2])
```

Sli.do #cse160

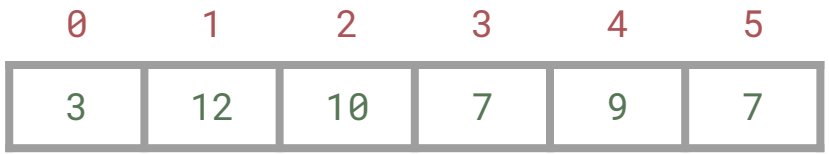
Removing elements from a list

`list_name.remove(x)`

Removes the first occurrence of **`x`** from **`list_name`**.

Note: It is an Error if **`x`** is not in **`list_name`**.

```
ticket_buyers = [3, 12, 10, 7, 9, 7]
ticket_buyers.remove(7)
ticket_buyers.remove(8)  # Error
```



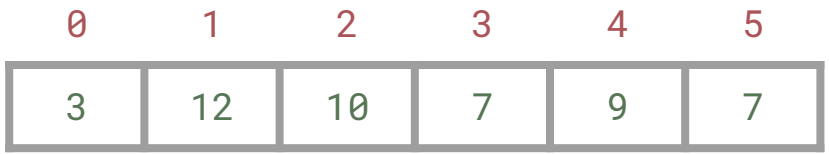
Note: **`remove`** returns **`None`**

Another way to remove elements from a list

`list_name.pop(i)` Removes and returns the element at index **`i`** from **`list_name`**.

`list_name.pop()` Removes and returns the last element from **`list_name`**.

```
ticket_buyers = [3, 12, 10, 7, 9, 7]
x = ticket_buyers.pop(3)
y = ticket_buyers.pop()
```



Replace elements in a list

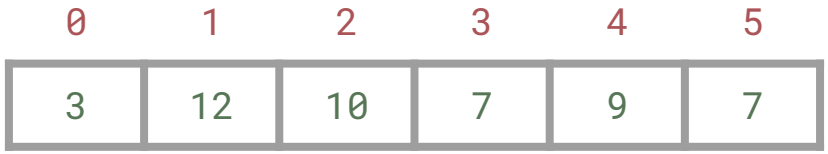
`list_name[i] = new_value`

- Replaces the value currently stored at index **`i`** with **`new_value`**.

`list_name[start:end] = new_sublist`

- Replaces **`list_name[start]`**, . . .**`list_name[end - 1]`** with **`new_sublist`**.
- Note: This can change the length of **`list_name`**!

```
ticket_buyers = [3, 12, 10, 7, 9, 7]
ticket_buyers[3] = 25
ticket_buyers[2:4] = [20, 21, 30]
```



Removing and Replacing Elements Examples

```
lst = [1, 2, 3, 4, 5, 6, 7]
print(lst.pop())
print(lst.pop(1))
lst.remove(3)
lst[3] = 'blue'
lst[1:3] = [10, 11, 12]
```

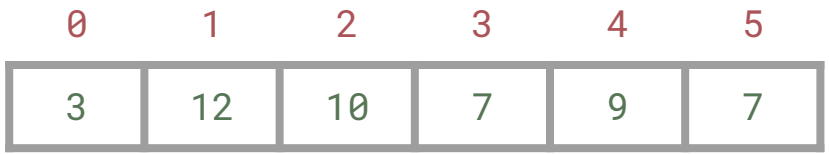
Python Tutor

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**

Modifying Lists Summary

- Add an element to the end
- Add each element from L to the end
- Insert element x at index i
- Remove first occurrence of x
- Remove and return element at index i
- Replace value at index I
- Sort the list
- Reverse the list

```
my_list.append(x)
my_list.extend(L)
my_list.insert(i, x)
my_list.remove(x)
my_list.pop(i)
my_list[i] = x
my_list.sort()
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
```