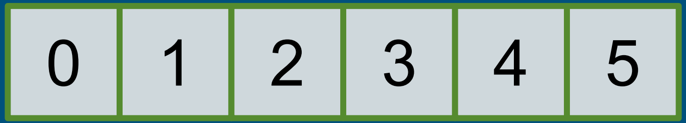
# List and More Iteration

CS 101
Profs. Briggs and Grant

# What are **Lists?**

A **List** is a mutable, ordered sequence of elements



They are one of many Data Structures that you will learn about - ways of storing and organizing data.

## How do we **make** a list?

Method 1:
Using the built-in list constructor

```
L = list()
```

OR

```
L = list([element1, element2, …])
```

# How do we **make** a list?

Method 2:
Using the shorthand display notation (hard brackets)

L = [ ]

OR

L = [element1, element2, …]

## How do we **make** a list?

Method 3:
Using a list comprehension

L = list(expression for variable in sequence)

OR

L = [ expression for variable in sequence ]

(Yes, a “for” loop in a list!)

## What can we put **inside** a list?

ANYTHING, as long as it is a valid Python expression!

Examples:

L = [ 5, 10, ‘Hello’, 5 < 3, 5+3, ‘c’*3 ]

Is valid syntax and evaluates to

[ 5, 10, ‘Hello’, False, 8, ‘ccc’ ]

## What can we put **inside** a list?

You can even put a list inside of a list!

Examples:

```
L = [ 5, 10, ['the', 'inner', 'list'], 8, 'ccc' ]
```

## How do we **access** items in the list?

Indexing!

Given the list L = [“cat”, “dog”, 10, 20, True]

len(L) is 5

Indices go from 0 to 4

L[0] is “cat”

L[4] is True

L[1:4] is [“dog”, 10, 20]

## How do we **access** items in the list?

What if we use a nested list?

L = [ 5, 10, [‘the’, ‘inner’, ‘list’], 8, ‘ccc’ ]

len(L) is 5

Indices go from 0 to 4

L[2] is [‘the’, ‘inner’, ‘list’]

L[2][1] is ‘inner’

## How do we **manipulate** items in the list?

Method 1:

Assign a value to a (valid) index

```
L = [10, 20, 30, 40, 50]
L[2] =  100
L = [10, 20, 100, 40, 50]
```

## How do we **manipulate** items in the list?

Method 2:

Assign a value to a (valid) index

```
L = [10, 20, 30, 40, 50]
L.append(100)
L = [10, 20, 30, 40, 50, 100]
```

# EXAMPLES