

## Lists 2.0: looping through lists.

**Week 7** | Lecture 2 (7.2)

**While waiting, open the Jupyter Notebook for today's lecture**

### Upcoming

- Lab 4 due this Friday 11:59 pm.
  - Lab 5 Released Thursday 6:00 pm.
  - Reflection 7 Released Friday 6:00 pm.
  - Tutorial (in-person AND online) running all week.
  - Practical sessions (in-person AND online) running ONLY Friday this week.
- if nothing else, write `#cleancode`

# This Week's Content

- **Lecture 7.1**
  - Lists: indexing and slicing
- **Lecture 7.2**
  - Lists: nested lists and looping
- **Lecture 7.3**
  - Design Problem! Cryptography...

# Recap! Adding to a list...

| Method                                | Description   | Example  |
|---------------------------------------|---|--|
| <code>list.append(object)</code>      | Append object to end of list                                | <pre>&gt;&gt;&gt; colours = ['blue', 'yellow'] &gt;&gt;&gt; colours.append('brown') &gt;&gt;&gt; colours ['blue', 'yellow', 'brown']</pre>                   |
| <code>list.extend(list)</code>        | Append the items in the list parameter to the list          | <pre>&gt;&gt;&gt; colours = ['blue', 'yellow'] &gt;&gt;&gt; colours.extend(['pink', 'green']) &gt;&gt;&gt; colours ['blue', 'yellow', 'pink', 'green']</pre> |
| <code>list.insert(int, object)</code> | Insert object at the given index, moving items to make room | <pre>&gt;&gt;&gt; grades = [95, 65, 75, 85] &gt;&gt;&gt; grades.insert(3, 80) &gt;&gt;&gt; grades [95, 65, 75, 80, 85]</pre>                                 |

# Recap! Removing from a list...

| Method                           | Description  | Example   |
|----------------------------------|--|---|
| <code>list.remove(object)</code> | Remove the first occurrence of the object; <b>error if not there</b>           | <pre>&gt;&gt;&gt; colours = ['blue', 'yellow'] &gt;&gt;&gt; colours.remove('blue') &gt;&gt;&gt; colours ['yellow']</pre>  |
| <code>list.pop([index])</code>   | Remove the item at the end of the list; optional index to remove from anywhere | <pre>&gt;&gt;&gt; colours = ['blue', 'yellow', 'pink'] &gt;&gt;&gt; colours.pop() 'pink' &gt;&gt;&gt; colours ['blue', 'yellow'] &gt;&gt;&gt; colours.pop(0) 'blue' &gt;&gt;&gt; colours ['yellow']</pre> |

# Recap! The fun stuff...

| Method                          | Description  | Example   |
|---------------------------------|--|---|
| <code>list.reverse()</code>     | Reverse the list   | <pre>&gt;&gt;&gt; colours = ['blue', 'yellow', 'pink'] &gt;&gt;&gt; colours.reverse() &gt;&gt;&gt; colours ['pink', 'yellow', 'blue']</pre> |
| <code>list.sort()</code>        | Sort the list from smallest to largest (also sorts list of strings alphabetically) | <pre>&gt;&gt;&gt; grades = [95, 65, 75, 85] &gt;&gt;&gt; grades.sort() &gt;&gt;&gt; grades [65, 75, 85, 95]</pre>                           |
| <code>list.count(object)</code> | Return the number of times object occurs in list                                   | <pre>&gt;&gt;&gt; letters = ['a', 'a', 'b', 'c'] &gt;&gt;&gt; letters.count('a') 2</pre>  |
| <code>list.index(object)</code> | Return the index of the first occurrence of object;<br><b>error if not there</b>   | <pre>&gt;&gt;&gt; letters = ['a', 'a', 'b', 'c'] &gt;&gt;&gt; letters.index('a') 0</pre>  |

# List and String Similarities

- Lists share many similarities with strings
  - Indexing (the [ ] operator)
  - Slicing ([start : end] and [start : end : step])
  - Membership (the in operator)
  - Length (built-in function len)
  - Concatenate (the + operator combining lists with other lists)
  - Repeat (the \* operator between lists and an integer)
  - Comparison operators (>, <, ==, !=, etc.)

# List and String Differences

- Lists can contain a mixture of any Python objects
  - Strings only hold characters
- Lists are mutable (i.e. their elements can be changed)
  - Strings are immutable
- Lists are designated with `[ ]`, with elements separated by commas
  - Strings are designated with `" "` or `' '`



# Motivating Example: The Speeder

- We have a list of numbers that represent velocity of a car taken at regular intervals

```
speed_list = [70, 97, 101, 120, 116, 110, 98, 99, 100, 102]
```

- Assuming the speed limit is 100 km/h, we want to examine many times the car is speeding. How do we achieve this?



# Motivating Example: The Speeder

- Using what we've learned so far, we would need to write ten if statements to check if velocity is greater than 100 km/h

```
if speed_list[0] > 100:  
    print("speeding")
```

```
if speed_list[1] > 100:  
    print("speeding")
```

```
if speed_list[2] > 100:  
    print("speeding")
```

```
. . .
```

```
if speed_list[9] > 100:  
    print("speeding")
```

Repeating code -> think loops!

# for loops

- A **for** loop starts with the keyword **for**.
- Next, we provide the name of one of more variables
- Our variable **character** will be bound to each of the items in the sequence in turn.
- What is the iterable?
- An iterable is an object that can be iterated over.

## GENERAL FORM:

```
for item in iterable:  
    do something
```

## EXAMPLE:

```
name = 'Sebastian'
```

```
for character in name:  
    print(character)
```

# Example: for Loop through List

- Iterate over a list of strings


```
fruits = ['apples', 'oranges', 'pears', 'apricot']
```

```
for fruit in fruits:  
    print(fruit)
```

# Visualize for Loop through a List

```
fruits = ['apples', 'oranges', 'pears', 'apricot']
```

OUTPUT:




```
for fruit in fruits:  
    print(fruit)
```

# Visualize for Loop through a List

```
fruits = ['apples', 'oranges', 'pears', 'apricot']
```

OUTPUT:



```
for fruit in fruits:  
    print(fruit)
```

# Visualize for Loop through a List

```
fruits = ['apples', 'oranges', 'pears', 'apricot']
```



```
for fruit in fruits:  
    print fruit
```

OUTPUT:


apples

# Visualize for Loop through a List

```
fruits = ['apples', 'oranges', 'pears', 'apricot']
```

OUTPUT:

apples



```
for fruit in fruits:  
    print(fruit)
```

# Visualize for Loop through a List



```
fruits = ['apples', 'oranges', 'pears', 'apricot']
```

```
for fruit in fruits:  
    print fruit
```

OUTPUT:


apples

oranges



# Visualize for Loop through a List

```
fruits = ['apples', 'oranges', 'pears', 'apricot']
```



```
for fruit in fruits:  
    print(fruit)
```

OUTPUT:

apples

oranges

# Visualize for Loop through a List

```
fruits = ['apples', 'oranges', 'pears', 'apricot']
```



```
for fruit in fruits:  
    print fruit
```

OUTPUT:


apples

oranges

pears

# Visualize for Loop through a List

```
fruits = ['apples', 'oranges', 'pears', 'apricot']
```



```
for fruit in fruits:  
    print(fruit)
```

OUTPUT:

apples

oranges

pears

# Visualize for Loop through a List

```
fruits = ['apples', 'oranges', 'pears', 'apricot']
```




```
for fruit in fruits:  
    print fruit
```

OUTPUT:

```
apples  
oranges  
pears  
apricot
```

# Visualize for Loop through a List

```
fruits = ['apples', 'oranges', 'pears', 'apricot']
```



```
for fruit in fruits:  
    print(fruit)
```

OUTPUT:

```
apples  
oranges  
pears  
apricot
```

# Visualize for Loop through a List

```
fruits = ['apples', 'oranges', 'pears', 'apricot']
```

```
for fruit in fruits:  
    print(fruit)
```



Next line of code...

OUTPUT:

apples

oranges

pears

apricot

# Python Visualizer

Watch how a loop works through a simple list:

- <https://tinyurl.com/aps106listloop>

# Let's Code!

- Let's take a look at how this works in Python!
  - Looping through a list
  - BREAKOUT SESSION 1

**Open your  
notebook**

**Click Link:**

**1. For Loops Over  
Lists**



# Back to Our Speedster

- We have a list of numbers that represent velocity of a car taken at regular intervals

```
speed_list = [70, 97, 101, 120, 116, 110, 98, 99, 100, 102]
```

- Assuming the speed limit is 100 km/h, we want to examine many times the car is speeding. How do we achieve this?

# Let's Code!

- Let's take a look at how this works in Python!
  - BREAKOUT SESSION 2

**Open your  
notebook**

**Click Link:**  
**2. Speedster 1.0**

# Nested Lists Require Nested Loops!

➔ `aps106_grades = [['Midterm 1', 60],  
                      ['Midterm 2', 90],  
                      ['Exam', 100]]`

OUTPUT:


```
for row in aps106_grades:  
    for column in row:  
        print(column)
```



# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```

OUTPUT:




```
for row in aps106_grades:  
    for column in row:  
        print(element)
```

# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```

OUTPUT:




```
for row in aps106_grades:  
    for column in row:  
        print(element)
```

# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```

OUTPUT:



```
for row in aps106_grades:  
    for column in row:  
        print(column)
```


# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```

OUTPUT:

Midterm 1

```
for row in aps106_grades:  
    for column in row:  
        print(column)
```




# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```

OUTPUT:

Midterm 1



```
for row in aps106_grades:  
    for column in row:  
        print(column)
```



# Nested Lists Require Nested Loops!

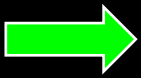
```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```

OUTPUT:

Midterm 1

60

```
for row in aps106_grades:  
    for column in row:  
        print(column)
```




# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```

OUTPUT:

Midterm 1

60




```
for row in aps106_grades:  
    for column in row:  
        print(column)
```

# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```

OUTPUT:

Midterm 1  
60




```
for row in aps106_grades:  
    for column in row:  
        print(column)
```

# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```

OUTPUT:

Midterm 1  
60



```
for row in aps106_grades:  
    for column in row:  
        print(column)
```

# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```


OUTPUT:

Midterm 1

60

Midterm 2

```
for row in aps106_grades:  
    for column in row:  
        print(column)
```



# Nested Lists Require Nested Loops!


```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```

OUTPUT:

Midterm 1

60

Midterm 2




```
for row in aps106_grades:  
    for column in row:  
        print(column)
```

# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```

```
for row in aps106_grades:  
    for column in row:  
        print(column)
```



OUTPUT:

Midterm 1

60

Midterm 2

90

# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```


OUTPUT:

Midterm 1

60

Midterm 2

90



```
for row in aps106_grades:  
    for column in row:  
        print(column)
```



# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```


OUTPUT:

Midterm 1

60

Midterm 2

90



```
for row in aps106_grades:  
    for column in row:  
        print(column)
```

# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```


OUTPUT:

Midterm 1

60

Midterm 2


90



```
for row in aps106_grades:  
    for column in row:  
        print(column)
```

# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```



```
for row in aps106_grades:  
    for column in row:  
        print(column)
```

OUTPUT:

Midterm 1

60

Midterm 2

90

Exam

# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```

OUTPUT:


Midterm 1

60

Midterm 2

90


Exam



```
for row in aps106_grades:  
    for column in row:  
        print(column)
```

# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```



```
for row in aps106_grades:  
    for column in row:  
        print(column)
```

OUTPUT:

Midterm 1

60

Midterm 2

90

Exam

100

# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```

OUTPUT:

Midterm 1


60

Midterm 2

90

Exam

100



```
for row in aps106_grades:  
    for column in row:  
        print(column)
```

# Nested Lists Require Nested Loops!

```
aps106_grades = [['Midterm 1', 60],  
                  ['Midterm 2', 90],  
                  ['Exam', 100]]
```

OUTPUT:

Midterm 1


60

Midterm 2

90

Exam

100



```
for row in aps106_grades:  
    for column in row:  
        print(column)
```

# Python Visualizer

Watch how a loop works through a nested list:

- <https://tinyurl.com/aps106listloop2>



# Let's Code!

- Let's take a look at how this works in Python!
  - Looping through nested lists with nested loops
  - Adding matrices



Matrix addition reminder:

$$\begin{bmatrix} 4 & 8 \\ 3 & 7 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 5 & 2 \end{bmatrix} = \begin{bmatrix} 4+1 & 8+0 \\ 3+5 & 7+2 \end{bmatrix}$$

**Open your  
notebook**

**Click Link:**

**3. Looping Over  
Nested Lists**

## Lists 2.0: looping through lists.

Week 7 | Lecture 2 (7.2)

if nothing else, write `#cleancode`