# ITP 115

## Lists

# Recall: Sequences Have Indices!

- Each individual item in a sequence is automatically given a position number

- This number is called an **index** and tells what position the item is in

- The **first index** is **zero (0)**

- The **last index** is the **number of items – 1**

# Recall: Two Categories of Sequences

- Mutable – changeable
  - Can modify A SINGLE item in the sequence


- Immutable – unchangeable
  - Can **NOT** modify A SINGLE item in the sequence

# Strings are Immutable

```
word = "game"
print (word)
word[0] = "l"
```

TypeError: 'str' object does not support item assignment

# Strings are Immutable

- Well that's frustrating…

- What kind of sequence is mutable then?

# Consider…

**Ask the user for three test scores.  Display the average along with the original scores.**

Create a **count** (*set to 0*) and create a **sum** (*set to 0*)

Ask user for 1$^{st}$ number (store in **testScore1**)
  Add number to sum and increment counter
Ask user for 2$^{nd}$ number (store in **testScore2**)
  Add number to sum and increment counter
Ask user for 3$^{rd}$ number (store in **testScore3**)
  Add number to sum and increment counter

Display **testScore1**, **testScore2**, **testScore3**, and average (**sum/count**)

# Consider…

## Now you have 6 test scores…

Create a **count** (*set to 0*) and create a **sum** (*set to 0*)

Ask user for 1$^{st}$ number (store in **testScore1**)
      Add number to sum and increment counter
Ask user for 2$^{nd}$ number (store in **testScore2**)
      Add number to sum and increment counter
Ask user for 3$^{rd}$ number (store in **testScore3**)
      Add number to sum and increment counter
Ask user for 4$^{th}$ number (store in **testScore4**)
      Add number to sum and increment counter
Ask user for 5$^{th}$ number (store in **testScore5**)
      Add number to sum and increment counter
Ask user for 6$^{th}$ number (store in **testScore6**)
      Add number to sum and increment counter

Display **testScore1**, **testScore2**, **testScore3**, **testScore4**, **testScore5**, **testScore6** and average (**sum/count**)

# Consider…

- Using a separate variable for each score…
  - Is impractical for more than a few scores
  - Makes it difficult to use a for loop for efficiency
  - Prone to errors

- All the scores are *related* so….
  - Instead we use a sequence (or group) of variables called a **list**

# Lists

- New type of variable!

- Are sequences like strings, but lists are **mutable**

- Contain all the same type of elements*
  - i.e. all strings or all ints

*Technically, Python allows lists to hold different types of elements. For our class, though, we will only store "like items"*

# Lists

- Syntax

  `itemsList = [item1, item2, ...]`

- `item1` could be any type of variable
  - string: `"hello"`
  - int: `7`
  - float: `8.5`
  - another list: `["this is", "another list"]`
  - Any other variable type we will cover

# Lists are Sequences

- Since lists are sequences, you can manipulate them just like strings!

# Lists are Sequences

```
things = ["emu", "pig"]
stuff = ["dog", "cat", "boa"]
```

**things**

| 0 | 1 |
|-----|-----|
| emu | pig |

**stuff**

| 0 | 1 | 2 |
|-----|-----|-----|
| dog | cat | boa |

# Lists are Sequences

**things**

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| emu | pig | dog | cat | boa |

```
things = ["emu", "pig"]
stuff = ["dog", "cat", "boa"]
```

**stuff**

| 0 | 1 | 2 |
|---|---|---|
| dog | cat | boa |

```
#concatenate
things += stuff
#alternatively
things = things + stuff
```

# Lists are Sequences

```
things = ["emu", "pig"]
stuff = ["dog", "cat", "boa"]

#index operator

something = stuff[0]
```

**things**

| 0 | 1 |
|-----|-----|
| emu | pig |

**stuff**

| 0 | 1 | 2 |
|-----|-----|-----|
| dog | cat | boa |

**animal**

| dog |
|-----|

- What type of variable is **stuff**?
- What type of variable is stored at **stuff[0]**?
- What type of variable is stored in **something**?

# Lists are Sequences

```
things = ["emu", "pig"]
stuff = ["dog", "cat", "boa"]

#slices

grabBag = stuff[0:2]
```

What type of variable is **grabBag**?

**things**

| 0 | 1 |
|-----|-----|
| emu | pig |

**stuff**

| 0 | 1 | 2 |
|-----|-----|-----|
| dog | cat | boa |

**grabBag**

| 0 | 1 |
|-----|-----|
| dog | cat |

# Lists are Sequences

```
things = ["emu", "pig"]
stuff = ["dog", "cat", "boa"]

#len operator

length = len(stuff)
```

**things**

| 0 | 1 |
|-----|-----|
| emu | pig |

**stuff**

| 0 | 1 | 2 |
|-----|-----|-----|
| dog | cat | boa |

**length**

| 3 |
|---|

# Lists are Sequences

**things**

| 0 | 1 |
|---|---|
| emu | pig |

**stuff**

| 0 | 1 | 2 |
|---|---|---|
| dog | cat | boa |

```
things = ["emu", "pig"]
stuff = ["dog", "cat", "boa"]

#in operator
if "dog" in stuff:
    print("Found dog")
else:
    print("No dog found")
```

**Found dog**

# Lists are Sequences

**things**

| 0 | 1 |
|---|---|
| emu | pig |

**stuff**

| 0 | 1 | 2 |
|---|---|---|
| dog | cat | boa |

```
things = ["emu", pig]
stuff = ["dog", "cat", "boa"]

#for loop

for item in stuff:
        print(item)
```

dog

cat

boa

# Creating Empty Lists

- Often we will want to create an empty list before a loop, at the start of our program, etc.

- Syntax
  ```
  numbers = list()
  ```

  *or*

  ```
  numbers = []
  ```

# Lists are Mutable!

- Assign a new list element by index
- Assign a new list slice
  - Replace multiple items with one item
- Delete a list element
  - Doesn't create a gap in a sequence
  - All the elements "slide down" one position
- Delete a list slice
  - Delete multiple elements

# Lists are Mutable!

nums        =  [12, -3, 5]

nums[0]     =  46

nums[1]     =  324

*Note* **[1]** *refers to index position, not the value*

| nums | 0 | 1 | 2 |
|------|----|----|----|
|      | 12 | -3 | 5 |

| nums | 0 | 1 | 2 |
|------|----|----|----|
|      | 46 | -3 | 5 |

| nums | 0 | 1 | 2 |
|------|----|-----|----|
|      | 46 | 324 | 5 |

USC

# List Methods

| Method | Description |
| --- | --- |
| `someList.append(value)` | Adds value to end of a list. |
| `someList.remove(value)` | Removes the first occurrence of value from the list. |
| *The following will be covered next week* | |
| `someList.sort()` | Sorts the elements, smallest value first. |
| `someList.reverse()` | Reverses the order of a list. |
| `someList.count(value)` | Returns the number of occurrences of value. |
| `someList.index(value)` | Returns the first position number of where value occurs. |
| `someList.insert(i, value)` | Inserts value at position i. |
| `someList.pop([i])` | Returns value at position i and removes value from the list. Position number i is optional; omitting will remove last element and return it. |
| `del someList[i]` | Removes the element at the specified index |

# someList.append(someValue)

- Adds value to end of a list

- Example
  ```
  numbers = [3, 5, -12]
  ```

| 0 | 1 | 2 |
|---|---|---|
| 3 | 5 | -12 |

# someList.append(someValue)

- Adds value to end of a list

- Example

```
numbers = [3, 5, -12]
numbers.append(40)
```

| 0 | 1 | 2 | 3 |
|---|---|----|----|
| 3 | 5 | -12 | 40 |

# someList.remove(someValue)

- Removes the first occurrence of a *value* from list

- Example

  `numbers = [3, 5, -12, 40, 5]`

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 3 | 5 | -12 | 40 | 5 |

# someList.remove(someValue)

- Removes the first occurrence of a *value* from list

- Example

```
numbers = [3, 5, -12, 40, 5]
numbers.remove(5)
```

| 0 | 1 | 2 | 3 |
|---|-----|-----|---|
| 3 | -12 | 40 | 5 |

# someList.remove(someValue)

- Removes the first occurrence of a *value* from list

- Example

  ```
  numbers = [3, 5, -12, 40, 5]
  numbers.remove(5)
  numbers.remove(5)
  ```

  | 0 | 1 | 2 |
  |---|---|---|
  | 3 | -12 | 40 |

# someList.remove(someValue)

- Removes the first occurrence of a *value* from list

- Example

```
numbers = [3, 5, -12, 40, 5]
numbers.remove(5)
numbers.remove(5)
numbers.remove(5)
        ValueError: list.remove(x): x not in list
```

| 0 | 1 | 2 |
|---|---|---|
| 3 | -12 | 40 |

Important: Always check **if** value is in list before removing it