CIS 122

Looping for a while

All the while...

- Yesterday we learned how to loop in Python
- While condition is true
  - Run code
  - Repeat

```
while x < 10:
    print x
    x = x + 1
```

- Iteration
  - Doing the same task over and over...
  - Not to be confused with **recursion**...

Looping Over Letters

- How could we tell if a string contains the character 'a' ?
- Easy to check a single character of our string
  - Is the ith character of our string an 'a' ?
  - if s[i] == 'a'
- Want to do that for each character in string
  - Let's set up a loop!

Looping Over Letters

- Initialization
  - Set a character counter to 0
- Loop Body
  - Check current character
    - If we see an 'a', return True
  - Increment counter
- Terminating Condition
  - Loop until counter reaches length of string
  - If we haven't seen an 'a' yet, return False

Looping Over Letters

```
def contains_a(string):
    """Returns True if string contains 'a'; False otherwise"""

    i = 0                               # Initialization

    while i < len(string):              # Condition
        if string[ i ] == 'a':
            return True                 # Body
        i = i + 1

    return False                        # Afterwards
```

Looping Over Letters

- Looping over sequences is a really common task
  - Python provides a special loop for doing just that
- The for loop allows us to perform some task **for** each element in a sequence
  - Useful for iterating over strings

Looping Over Letters

```
for char in "abcde":
    print "The current letter is:"
    print char
```

- Set char to first element of sequence
  - Perform loop body
- Set char to second element of sequence
  - Perform loop body
- Repeat for each element of sequence

Looping Over Letters

- Let's rewrite our a-checker with a for loop
- For each character in our string
  - If that character is an 'a', return True
- If we haven't returned True by the end of the loop
  - There must be no 'a'
  - Return False

Looping Over Letters

```
def contains_a(string):
    """Returns True if string contains "a"; False otherwise"""

    for c in string:            # For each character c in string
        if c == 'a':
            return True         # If we've found an 'a', return True

    return False                # If we never find an 'a', return False
```

Starting a Collection

- For loops are used for iterating over sequences
- What is a sequence?
  - Any type containing multiple elements
  - Strings
- Strings are a very specific type of sequence
  - Hold multiple characters
- What if we wanted a general sequence
  - Hold multiple values of any type

Starting a Collection

- Lists
  - [1, 2, 3]
  - ["apple", "banana", "carrot"]
  - [True, 'B', 3]
- What is a list?
  - A collection of values
  - Surrounded by braces
  - Separated by commas

Starting a Collection

- What can we put in a list?
  - Anything we want
  - Values
  - Variables
  - Expressions
  - Other lists!
- How do we get stuff out?
  - Just like strings
  - Indexing
  - Slicing

Take it to the Max

- Let's write a general max function
  - Takes a list of elements (any number)
  - Returns the largest
- How would you find the largest element of a list
  - If you don't know how long it is...

Take it to the Max

- Let's write a general max function
  - Takes a list of elements (any number)
  - Returns the largest
- How would you find the largest element of a list
  - If you don't know how long it is...
- Use a for loop to iterate through list
- Keep track of the largest element we've seen

Take it to the Max

```
def max(myList):
    """Returns the largest value in a list"""

    maxValue = myList[ 0 ]    # Assume first element is the largest

    for element in myList:
        if element > maxValue:
            maxValue = element    # Unless we find a larger one...

    return maxValue
```