CIS 122

Going Loopy

Logistics

- Midterms graded and recorded
  - Square root curve
  - sqrt(x)*10
- Assignment 3 not yet fully graded
  - Really neat fractals, though
  - I'll show them off tomorrow
- Assignment 4 posted
  - Check it out

The Fibonacci Function

```
def fibonacci(n):
    '''Returns nth fibonacci number'''
    #Initialize counters
    x = 0
    y = 1
    # Loop n times
    for i in range(n):
        z = x+y
        x = y
        y = z
    # There's our answer!
    return x
```

What's it all for?

```
for i in range(n):
    <do stuff>
```

There's a lot going on in this small line of code
Let's break it down...

What's it all for?

```
for iterator in sequence:
    <do stuff>
```

There's a lot going on in this small line of code
Let's break it down...

For each element in the given sequence,
Assign iterator to be that element
And do stuff
Then do it again

What's it all for?

```
for iterator in sequence:
    <do stuff>
```

- Iterator can be any variable name
  - i
  - ctr
  - accumulator
- Sequence can be any sequence type
  - strings
  - tuples
  - lists

What's it all for?

```
for char in "Hello World":
    print char
```

What's it all for?

```
for char in "Hello World":
    print char
```

Goes through each character in "Hello World"

Prints out each character in turn

What's it all for?

```
total = 0
for x in [1,2,3,4,5]:
    total = total + x
print total
```

What's it all for?

```
total = 0
for x in [1,2,3,4,5]:
    total = total + x
print total
```

Goes through each number in [1, 2, 3, 4, 5]

Adds each number to total

What's it all for?

```
total = 0
for x in [1,2,3,4,5]:
    print x
    total = total + x
print total
```

Goes through each number in [1, 2, 3, 4, 5]

Prints out each number
Adds each number to total

Wait up, was that a new type?!

- Lists
- Collections of objects
  - [1, 2, 3]
  - ["abc", "def", "ghi"]
  - [1, "b", True]
- Awfully similar to tuples
  - A few minor differences...

Wait up, was that a new type?!

- Lists are very similar to strings
  - They're both sequences
- We can find the length of a list
  - len([1, 2, 3])
  - len([ ])
- We can index and slice lists
  - ['a', 'b', 'c'][0]
  - ['a', 'b', 'c'][-1]
  - ['a', 'b', 'c'][1:]

Back to loops

```
total = 0
for x in [1,2,3,4,5]:
    print x
    total = total + x
print total
```

Goes through each number in [1, 2, 3, 4, 5]

Prints out each number
Adds each number to total

Back to loops

```
total = 0
for x in [1,2,3,4,5]:
    print total
    total = total + 10
print total
```

Goes through each number in [1, 2, 3, 4, 5]

Ignores that number
Prints out total
Adds 10 to total

- We can perform an action for each element in a sequence

- What does it take to perform an action n times?
  - Loop through a sequence of length n
  - If only we could easily construct such a list...

- Python gives us just the tool we need
  - range(n)
  - Returns a list of all integers
  - 0 up to (but not including) n

```
>>> range(5)
[0, 1, 2, 3, 4]

>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

>>> range(0)
[ ]
```

Coming Full Circle

```
for i in range(n):
    <do stuff>
```

- So what does this code do?
  - range(n) evaluates to a list of length n
  - We do stuff for each element in that list
  - We do stuff n times