

EECS 12: Lecture 3

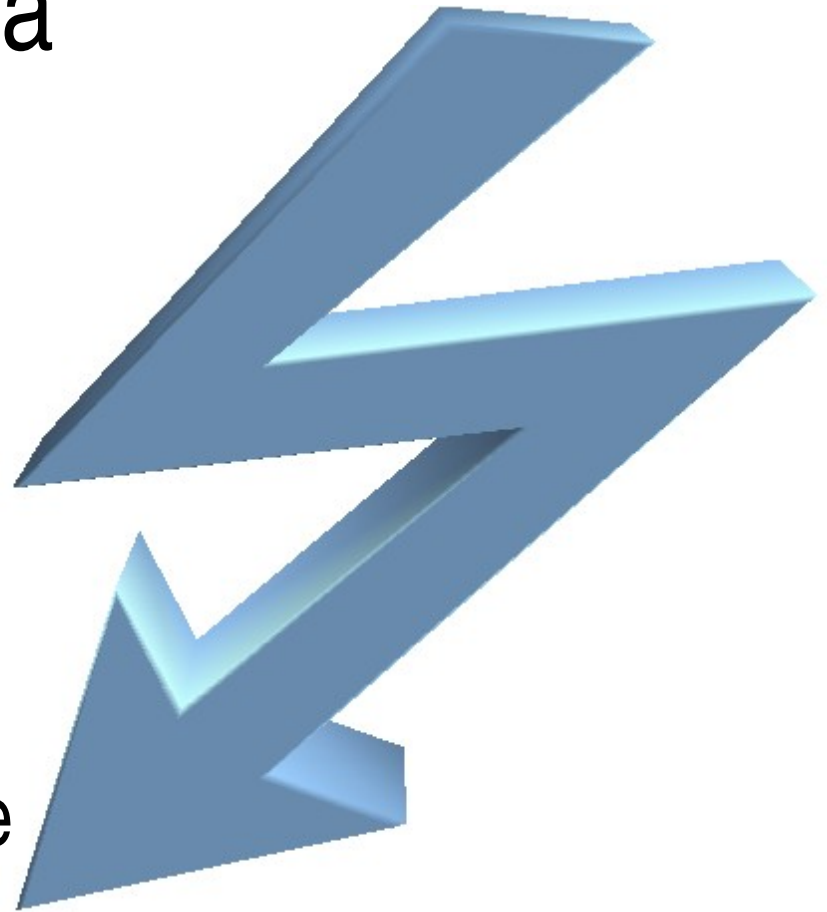
Iteration and Strings

Mark E. Phair
mphair@gmail.com
UC Irvine EECS

July 5th, 2006

Agenda

- Tabbed Browsing is awesome
- Program structure
- `for` loops
- `while` loops
- Strings and the `string` module
- Slicing
- Python factoid of the day



Is everyone reading the book? If not, you are missing out!
Section 7.8 has a poop joke! *sigh*

Aside: Tabbed Browsing Is Awesome

- Firefox and Opera both have tabbed browsing, and there are rumors that the new version of IE will FINALLY get this very useful feature
- In Firefox:
 - <Ctrl>+T = New Tab
 - <Ctrl>+Click link = Open link in new Tab
 - <Ctrl>+<TAB> = next tab

Aside: Program Structure

```
# boilerplate header info
# be sure to include author: line
import ...

##### FUNCTIONS #####

def ...

##### MAIN #####

statement

statement
```

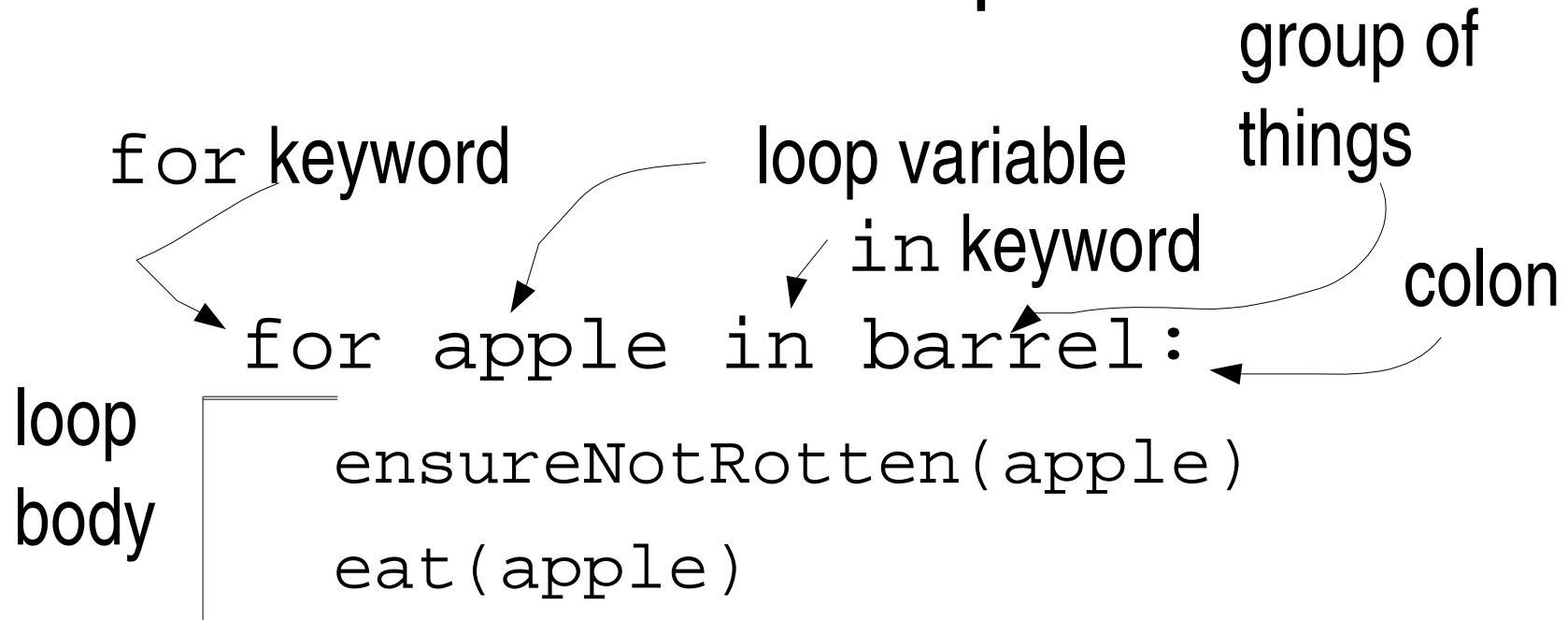
Iteration

- Doing a certain task repeatedly
- Something that computers are REALLY good at
- Called “looping.” Noun: a *loop*
- Give a man a fish and you feed him for a day. Teach a man to fish and he can work for your multinational fishing conglomerate.

for Loops

```
for apple in barrel:  
    ensureNotRotten(apple)  
    eat(apple)
```

for Loops



The `range` function

Constructs a `list` of integers

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

What does `range` have to do with `for`?

```
>>> for x in range(5):  
    print x
```

0

1

2

3

4

Let's explore...

Write a function that prints the first n perfect squares

The first line should look like this:

```
def firstNPerfectSquares(n):
```

while loops

```
>>> while(raw_input('? ') != 'stop'):  
        print 'still going...'  
?  
blah  
still going...  
?  
ack!  
still going...  
?  
stop  
>>>
```

while can act like for

```
>>> ii = 0
>>> while (ii < 4):
    print ii
    ii = ii + 1
```

0

1

2

3

When to use `for` versus `while`

- Use `for` when your program knows beforehand how many times it should do the loop
- Use `while` when your program does not know beforehand how many times it should do the loop

Side note about other languages

- the `for` loop shows up in most programming languages, but python's version of it is sometimes called `for` `each` instead
- Other language's for loop is more of a fancy `while` loop

Strings

- We've used strings for awhile now, but what *are* they?
- `int` and `float` can be thought of as being made up of only one “part” ... whereas strings are made up of a list of characters... they are *compound* data types
- Each character is a single ASCII or UNICODE value
- Use `len(aString)` to find length of `aString`

The `string` module

- Has a bunch of useful functions that you might end up needed to manipulate strings
- Examples:
 - `string.find`: find where in a string a substring occurs
 - `string.lower`: returns a copy of a string in lower case

Slicing

```
>>> aString = 'This is a string'
```

```
>>> aString[1]
```

```
'h'
```

```
>>> aString[1:3]
```

```
'hi'
```

More Slicing

```
>>> aString = 'This is a string'
```

```
>>> aString[1:len(aString)]
```

```
'his is a string'
```

```
>>> aString[1:-1]
```

```
'his is a strin'
```

```
>>> aString[2:-3]
```

```
'is is a str'
```

Even more slicing

```
>>> aString = "This is a string"
```

```
>>> aString[:]
```

```
"This is a string"
```

```
>>> aString[1:]
```

```
"his is a string"
```

```
>>> aString[:-2]
```

```
"This is a str"
```

Iteration and Strings

```
>>> aString = 'hello'
```

```
>>> for character in aString:  
    print character
```

h

e

l

l

o

The `in` keyword

- Saw it already in the context of a `for` loop
- Can also be used for testing...

```
>>> 'a' in 'abc'
```

```
True
```

```
>>> 'a' in 'xyz'
```

```
False
```

```
>>>
```

Mutability, Immutability, and Strings

- *Mutable* means *changeable*
- *Immutable* is the opposite: not changeable
- Strings are **IMMUTABLE**
- *Variables* containing strings can be changed... but the string values cannot be changed.

Python factoid of the day

$a \leq b \leq c$ is possible...

but look out in other languages: probably won't work!

```
>>> a, b, c = 1, 2, 3
```

```
>>> a <= b <= c
```

```
True
```

```
>>> c >= b >= a
```

```
True
```

```
>>>
```