For loops

Lecture #9 - COMP 364
February 3, 2010
Derek Ruths

Task 1: Working with all the elements of a list

- Often we will want to do something to each element in a list
  - In the case of “gene <start>..<end>”, extract <start> and <end> and subtract them to get the length of a gene
  - In the case of a bunch of data points, add them together and divide by the number of data points
- Version 1: Print out all the elements
- Version 2: Print out all the elements, one element per line
- Version 3: Compute the sum of a list of numbers
- Version 4: Compute the average of a list of numbers

The for loop: handling one element at a time

- In a for loop we define a set of actions that are done exactly once on each element in a list.

- All the steps must be identically indented under the “for...” statement

the variable that contains the value of the list element

the list to loop over

```
for x in sys.argv:
print x
```

```
for y in [1,2,3,4,5]:
print y
```

Exercise: computing the sum of a list of numbers

Exercise: counting the length of a list

Exercise: computing the average of a list of numbers

If-then-else statements

Lecture 10 - COMP 364
February 5, 2010
Derek Ruths

Recall...

Task 1: Working with all the elements of a list

- Often we will want to do something to each element in a list
  - In the case of “gene <start>..<end>”, extract <start> and <end> and subtract them to get the length of a gene
  - In the case of a bunch of data points, add them together and divide by the number of data points
- Version 1: Print out all the elements
- Version 2: Print out all the elements, one element per line
- Version 3: Compute the sum of a list of numbers
- Version 4: Compute the average of a list of numbers

Task 2: Working with specific elements of a list

- Often (potentially even more often than simple looping we will want to do something to a subset of the element in the list
  - In the case of “gene <start>..<end>”, we might want to print out only genes longer than 100 bps
  - We might want to average only data points that are less than some threshold
  - Count the number of data points that are in a specific range
- Version 1: Print out all strings in a list that are less than 5 characters long
- Version 2: Count the number of elements less than 10
- Version 3: Compute the sum of all numbers less than 20
- Version 4: Average all numbers less than 15

If-then-else statements

- **If**: Designate code that is executed only when a specific condition is True
- **Else**: Code that is executed when a specific condition is False

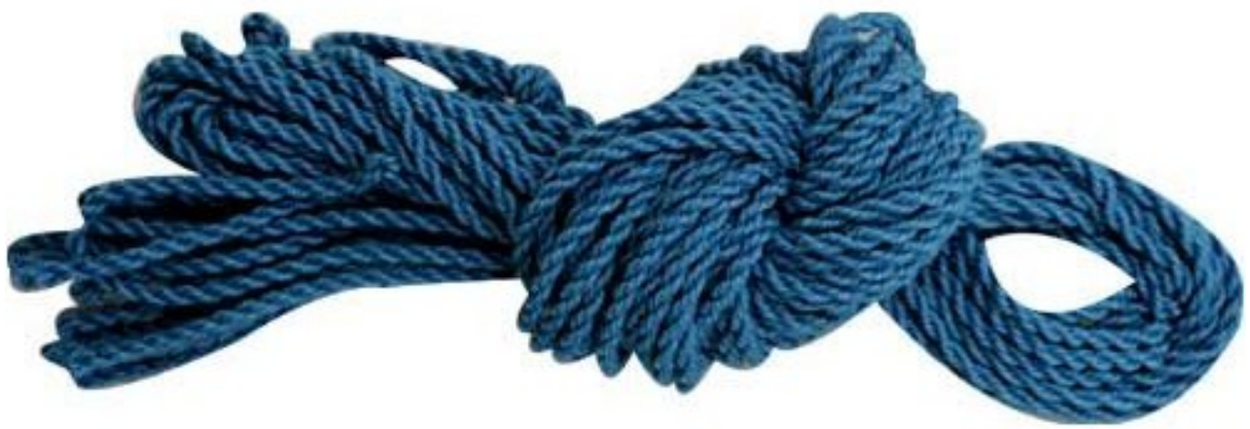
```
if len(sys.argv[1]) > 5:
print sys.argv[1]
else:
print 'Too short!'
```

```
if len(sys.argv[1]) > 5:
print sys.argv[1]
```

The else part is optional!

String functions

Lecture 12 - COMP 364
February 12, 2010
Derek Ruths



Extracting substrings

- x = ‘Hello world’
  - x[6] -> ‘w’
  - x[6:] -> ‘world’

Exercise: write a program that takes two files as input (each file has a DNA sequence) and identifies the SNPs in two sequences. For each SNP, the position and two values of the SNP should be printed out.

Modifying strings

- x.strip() - remove whitespace on either end of a string
  - x.lstrip(), x.rstrip()
- x.replace(‘foo’,’bar’)
  - “foobar”.replace(‘foo’,’bar’) -> ‘barbar’

Checking for content

- *x.startswith(y)* - does string x start with string y?
- *x.endswith(y)* - does string x end with string y?
- *y in x* - is the string y found anywhere in x?
- x.find(y) - finds the earliest instance of y in x. Returns the position.

Extracting substrings

- x.split() - return a list of all substrings in x separated by spaces
  - “x y z”.split() -> [“x”,”y”,”z”]
- x.split(‘,’)
  - “x,y z”.split(‘,’) -> [“x”,”y z”]

Exercise 1: reverse the order of information for each protein interaction

Exercise 2: print out interactions that have a score greater than X

Regular expressions

- *import re* - re is the regular expression module
  - *re.search('<regexp>',x)* - find exactly one place where the regular expression matches something in x
  - *re.findall('<regexp>',x)* - find all places where the regular expression matches something in x
  - These return *MatchObject*s

Exercise: count the number of genes in a genome file

The MatchObject

- m.groups(0) - return the matched string
- m.groups(i) - return the ith group in the match

- Examples:
  - *m = re.search('([0-9]+)([a-z]+)\.py','test33index.py')*
  - *m.groups(0) -> '33index.py'*
  - *m.groups(1) -> '33'*
  - *m.groups(2) -> 'index'*

Exercise: compute lengths of genes in a genome file.