Announcements

- Homework 3 deadline extended to Wednesday 10/2 @ 11:59pm.
- Optional Hog strategy contest due Thursday 10/3 @ 11:59pm.
- Homework 4 due Tuesday 10/8 @ 11:59pm.
- Project 2 due Thursday 10/10 @ 11:59pm.
- Guerrilla Section 2 this Saturday 10/5 & Sunday 10/6 10am–1pm in Soda.
  - Topics: Data abstraction, sequences, and non-local assignment.
  - Please RSVP on Piazza!
- Guest lecture on Wednesday 10/9, Peter Norvig on Natural Language Processing in Python.
Strings
Strings are an Abstraction

Representing data:

'200'   '1.2e-5'   'False'   '(1, 2)'

Representing language:

"""""And, as imagination bodies forth
The forms of things to unknown, and the poet's pen
Turns them to shapes, and gives to airy nothing
A local habitation and a name.
""""

Representing programs:

'curry = lambda f: lambda x: lambda y: f(x, y)'

(Demo)
String Literals Have Three Forms

>>> 'I am string!'
'I am string!'

>>> "I've got an apostrophe"
"I've got an apostrophe"

>>> '您好'
'您好'

>>> """"The Zen of Python
claims, Readability counts.
Read more: import this."""
'The Zen of Python\nclaims, Readability counts.\nRead more: import this.'

A backslash "escapes" the following character

"Line feed" character represents a new line

Single-quoted and double-quoted strings are equivalent
Strings are Sequences

```python
>>> city = 'Berkeley'
>>> len(city)
8
>>> city[3]  # An element of a string is itself a string, but with only one character!
'k'
```

**Length.** A sequence has a finite length.

**Element selection.** A sequence has an element corresponding to any non-negative integer index less than its length, starting at 0 for the first element.

(Demo)
String Membership Differs from Other Sequence Types

The "in" and "not in" operators match substrings

```python
>>> 'here' in "Where's Waldo?"
True
>>> 234 in (1, 2, 3, 4, 5)
False
```

Why? Working with strings, we usually care about words more than characters

The count method also matches substrings

```python
>>> 'Mississippi'.count('i')
4
>>> 'Mississippi'.count('issi')
1
```

the number of non-overlapping occurrences of a substring
Encoding Strings
### Representing Strings: the ASCII Standard

**American Standard Code for Information Interchange**

<table>
<thead>
<tr>
<th>0</th>
<th>NUL</th>
<th>SOH</th>
<th>STX</th>
<th>ETX</th>
<th>EOT</th>
<th>ENQ</th>
<th>ACK</th>
<th>BEL</th>
<th>BS</th>
<th>HT</th>
<th>LF</th>
<th>VT</th>
<th>FF</th>
<th>CR</th>
<th>SO</th>
<th>SI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DLE</td>
<td>DC1</td>
<td>DC2</td>
<td>DC3</td>
<td>DC4</td>
<td>NAK</td>
<td>SYN</td>
<td>ETB</td>
<td>CAN</td>
<td>EM</td>
<td>SUB</td>
<td>ESC</td>
<td>FS</td>
<td>GS</td>
<td>RS</td>
<td>US</td>
</tr>
<tr>
<td>2</td>
<td>!</td>
<td>&quot;</td>
<td>#</td>
<td>$</td>
<td>%</td>
<td>&amp;</td>
<td>(</td>
<td>)</td>
<td>*</td>
<td>+</td>
<td>,</td>
<td>-</td>
<td>.</td>
<td>/</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>:</td>
<td>;</td>
<td>&lt;</td>
<td>=</td>
<td>&gt;</td>
<td>?</td>
</tr>
<tr>
<td>4</td>
<td>@</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>M</td>
<td>N</td>
<td>O</td>
</tr>
<tr>
<td>5</td>
<td>P</td>
<td>Q</td>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
<td>V</td>
<td>W</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>[</td>
<td>\</td>
<td>]</td>
<td>^</td>
<td>_</td>
</tr>
<tr>
<td>6</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>u</td>
<td>v</td>
<td>w</td>
<td>x</td>
<td>y</td>
<td>z</td>
<td>{</td>
<td></td>
<td></td>
<td>}</td>
<td>DEL</td>
</tr>
</tbody>
</table>

- 8 rows: 3 bits
- 16 columns: 4 bits

- Layout was chosen to support sorting by character code
- Rows indexed 2–5 are a useful 6-bit (64 element) subset
- Control characters were designed for transmission

**Demo**
Representing Strings: the Unicode Standard

- 109,000 characters
- 93 scripts (organized)
- Enumeration of character properties, such as case
- Supports bidirectional display order
- A canonical name for every character

U+0058 LATIN CAPITAL LETTER X

U+263a WHITE SMILING FACE

U+2639 WHITE FROWNING FACE

http://ian-albert.com/unicode_chart/unichart-chinese.jpg
Representing Strings: UTF-8 Encoding

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Correspondence between characters and integers

UTF-8: Correspondence between those integers and bytes

A byte is 8 bits and can encode any integer 0–255.

<table>
<thead>
<tr>
<th>bytes</th>
<th>integers</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>0</td>
</tr>
<tr>
<td>00000001</td>
<td>1</td>
</tr>
<tr>
<td>00000010</td>
<td>2</td>
</tr>
<tr>
<td>00000011</td>
<td>3</td>
</tr>
</tbody>
</table>

Variable-length encoding: integers vary in the number of bytes required to encode them.

In Python: `string` length is measured in characters, `bytes` length in bytes.

(Demo)
Sequence Processing
Consider two problems:

- Sum the even members of the first n Fibonacci numbers.
- List the letters in the acronym for a name, which includes the first letter of each capitalized word.

**Sequence Processing**

enumerate naturals: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11.

map fib: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55.

filter even: 0, 2, 8, 34.

accumulate sum: ., ., ., ., =44
Consider two problems:

- Sum the even members of the first $n$ Fibonacci numbers.
- List the letters in the acronym for a name, which includes the first letter of each capitalized word.

enumerate words: 'University', 'of', 'California', 'Berkeley'

filter capitalized: 'University', 'California', 'Berkeley'

map first: 'U', 'C', 'B'

accumulate tuple: ('U', 'C', 'B')
Mapping a Function over a Sequence

Apply a function to each element of the sequence

```python
>>> alternates = (-1, 2, -3, 4, -5)
>>> tuple(map(abs, alternates))
(1, 2, 3, 4, 5)
```

The returned value of `map` is an iterable map object

A constructor for the built-in map type

The returned value of `filter` is an iterable filter object

(Demo)
Iteration and Accumulation
Iterable Values and Accumulation

*Iterable* objects give access to their elements in order.

Similar to a sequence, but does not always allow element selection or have finite length.

Many built-in functions take iterable objects as argument.

- **tuple**: Return a tuple containing the elements
- **sum**: Return the sum of the elements
- **min**: Return the minimum of the elements
- **max**: Return the maximum of the elements

For statements also operate on iterable values.
Reducing a Sequence

Reduce is a higher-order generalization of max, min, & sum.

>>> from operator import mul
>>> from functools import reduce
>>> reduce(mul, (1, 2, 3, 4, 5))
120

First argument: A two-argument function
Second argument: an iterable object

Similar to accumulate from Homework 2, but with iterable objects.
Generator Expressions

One large expression that evaluates to an iterable object

\[(\text{<map exp> for <name> in <iter exp> if <filter exp>})\]

• Evaluates to an iterable object.
• \(<\text{iter exp}>\) is evaluated when the generator expression is evaluated.
• Remaining expressions are evaluated when elements are accessed.

Short version: \[(\text{<map exp> for <name> in <iter exp>})\]

(Demo)