Today's topics

Notes from Zachary Dodds' CS 5 course at Harvey Mudd

Upcoming
  > More Python

Reading
  How to Think, 7.1-7.7 & 9.1-9.12

Data and lists?

Can all information be represented using lists?

Networks

Images/Video

Sounds/Speech

Text

Data and lists?

{ 2, 3, 5, 7, 11 }

Sets

Can all information be represented using lists?

Inside the machine…

What’s happening in python:
  \( x = 41 \)
  \( y = x + 1 \)

What is happening behind the scenes:

"variables as containers"

Python. Data Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Example</th>
<th>What is it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>3.14</td>
<td>values with a fractional part</td>
</tr>
<tr>
<td>long</td>
<td>10**100</td>
<td>integers &gt; 2147483647</td>
</tr>
<tr>
<td>int</td>
<td>42</td>
<td>integers &lt;= 2147483647</td>
</tr>
<tr>
<td>bool</td>
<td>True</td>
<td>the results from a comparison: ==, !=, &lt;, &gt;, &lt;=, =&gt;</td>
</tr>
<tr>
<td></td>
<td>False</td>
<td>&quot;Boolean value&quot;</td>
</tr>
</tbody>
</table>
Datatypes as genes...

<table>
<thead>
<tr>
<th>Dominant</th>
<th>Recessive</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>41 + True</td>
</tr>
<tr>
<td>long</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td></td>
</tr>
</tbody>
</table>

What will these results be?

- $1.0 / 5$
- $10^{100} - 10^{100}$
- $1 / 5$

Python Operators

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Caution Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>( )</td>
<td>Highest</td>
</tr>
<tr>
<td>**</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td></td>
</tr>
<tr>
<td>+ / %</td>
<td></td>
</tr>
<tr>
<td>&gt; &lt; ==</td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>Lowest</td>
</tr>
</tbody>
</table>

- $42$

% the "mod" operator

$x \% y$ returns the remainder when $x$ is divided by $y$

- $7 \% 3$
- $8 \% 3$
- $9 \% 3$
- $16 \% 7$

For what values of $x$ are these True?

- $x \% 2 == 0$
- $x \% 2 == 1$
- $x \% 4 == 0$

Computer Memory

Random Access Memory (RAM)

is a long list of memory locations

- bit = 1 "bucket" of charge
- byte = 8 bits
- word = 4 bytes = 32 bits

name: x

| type: int |
| LOC: 300  |

4 bytes for an int
**String functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>str()</code></td>
<td>Converts input to a string</td>
<td><code>str(42)</code> returns <code>'42'</code></td>
</tr>
<tr>
<td><code>len()</code></td>
<td>Returns the string's length</td>
<td><code>len('42')</code> returns 2</td>
</tr>
<tr>
<td>+</td>
<td>Concatenates strings</td>
<td><code>'XL' + 'II'</code> returns <code>'XLII'</code></td>
</tr>
<tr>
<td>*</td>
<td>Repeats strings</td>
<td><code>'VI'*7</code> returns <code>'VIVIVIVIVIVI'</code></td>
</tr>
</tbody>
</table>

Given these strings:

```
{s1 = "ha"
 s2 = "t"}
```

What are:
```
s1 + s2
2*s1 + s2 + 2*(s1+s2)
```

**String surgery**

```
s = 'duke university baby'
```

<table>
<thead>
<tr>
<th>Index</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>'d'</td>
<td>Read &quot;s-of-zero&quot; or &quot;s-zero&quot;</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>'e'</td>
<td>Which index returns 'e'?</td>
</tr>
<tr>
<td>len(s)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
s[len(s)] returns python != English
```

**More on strings**

- Negative indices count from the back:
  ```
s[-1] returns 'y'
s[-11] returns
s[-0] returns
```

- Slicing:
  ```
s[ : ] slices the string, returning a substring
```

- Skip-slicing:
  ```
s[ : : ] skip-slices, returning a subsequence
  the third index is the "stride" length
```

**Lists ~ Strings of anything**

```
L = [ 3.14, [2,40], 'third', 42 ]
```

- Indexing:
  ```
  L[0]
  L[0:1]
  L[len(L)]
  ```

- Slicing:
  ```
  L[ : ] 
  L[ : : ]
  ```

- How could you extract from L:
  ```
  'hi'
  ```
Raising and razing lists

"Quiz"

pi = [3,1,4,1,5,9]  Q = [ 'pi', 'isn't', [4,2] ]

message = 'You need parentheses for chemistry !'

Part 1

\[
\begin{align*}
\text{What are} & \quad \text{len(pi)} \\
\text{What are} & \quad \text{len(Q)} \\
\text{What slice of pi is} & \quad [3,1,4] \\
\text{What slice of pi is} & \quad [3,4,5] \\
\text{What are} & \quad \text{pi[0]} \times (\text{pi[1]} + \text{pi[2]}) \\
& \quad \text{and} \\
& \quad \text{pi[0]} \times (\text{pi[1:2]} + \text{pi[2:3]})
\end{align*}
\]

Part 2

\[
\begin{align*}
\text{What are} & \quad Q[0] \\
\text{What are} & \quad Q[0:1] \\
\text{What is} & \quad \text{message[9:15]} \\
\text{What is message[:5]} \\
\end{align*}
\]

Extra! Mind Muddlers

What is \( pi[pi[2]] \)?

How many nested pis before \( pi[pi[0]] \) produces an error?

Functioning in Python

Some basic, built-in functions:

- abs
- max
- min
- sum
- range
- round

\[
\begin{align*}
\text{abs} & \quad \text{absolute value} \\
\text{max} & \quad \text{of lists} \\
\text{min} & \quad \text{of lists} \\
\text{sum} & \quad \text{these change data from} \\
\text{range} & \quad \text{one type to another} \\
\text{round} & \quad \text{creates lists} \\
\end{align*}
\]

only as accurately as it can!

These are the most important: help dir

Functioning in Python

Far more are available in separate files, or modules:

```
import math
math.sqrt(1764)
dir(math)
```

accesses \texttt{math.py}'s functions

list\texttt{s all of} \texttt{math.py}'s functions

```
from math import *
pi
sin(\pi/2)
```

same, but without typing \texttt{math.} all of the time...

```
def dbl( x ):
    """ returns double its input, x ""
    return 2*x
```

# my own function!
Functioning in Python

```python
# my own function!
def dbl(x):
    """ returns double its input, x """
    return 2*x
```

Some of Python's baggage...

**Keywords**
- `def` starts the function
- `return` stops it immediately and sends back the return value

**Docstrings**
They become part of python's built-in help system!
With each function be sure to include one that
1. describes overall what the function does, and
2. explains what the inputs mean/are

Functioning in Python

```python
# is it dis-0 or dis-O, anyway?
def undo(s):
    """ this "undoes" its string input, s """
    return 'de' + s

>>> undo('caf')
>>> undo(undo('caf'))
```

strings, lists, numbers ... all data are fair game

random thoughts

```python
import random
random.choice(range(0, 100))
```

These are the most important: help dir

DNA

Our DNA's nucleotides:

http://genome.ucsc.edu/cgi-bin/hgGateway

AGCT