Agenda

• Announcements
• Functions – under the hood
• Strings
  — Problems solved via strings
  — The BIG SECRECY OF STRINGS
  — Often used operators
  — Often used functions
• For loop
• (list)
How Python executes a function call

1. Evaluate the argument (at the call site)
2. Assign the formal parameter name to the argument’s value
   – A new variable, not reuse of any existing variable of the same name
3. Evaluate the statements in the body one by one
4. At a return statement:
   – Remember the value of the expression
   – Formal parameter variable disappears – exists only during the call!
   – The call expression evaluates to the return value

```
def square(x):
    return x * x
```

```
Current expression:
1 + square(3 + 4)
1 + square(7)
1 + 49
50
```

```
Variables: x: 7
square(3 + 4)
```

```
1 + square(3 + 4)
1 + square(7)
1 + 49
50
```
Strings & Avatar

• “Sometimes your whole life boils down to one insane move!” (Jake Sully, Avatar, 2009)
• Oftentimes the core logic of a program boils down to one big string move!
Problems that are solved via strings

• Problem 1: Web crawler
  – 1) Get all the links from a web page.
  – 2) Go to the web pages found and do 1)

• Problem 2: Password checker
  – Are there 8 characters?
  – Is there at least one number?
  – Is there at least one capital letter?
  – Is there at least one symbol?

• Problem 3: APT “Forming Acronyms”
  – How do you pick out each word?
  – How do you pick out the first letter in a word?
The Secret of Strings

- Python sees a String as a sequence
- Each character gets a number (2)

\[
\begin{array}{c}
0 & 1 & 2 & 3 & 4 \\
-5 & -4 & -3 & -2 & -1 \\
\end{array}
\]

boy = “Peter”

- boy[0] = P
- boy[-2] = e
- boy[0:2] = Pe
A few string operators

<table>
<thead>
<tr>
<th>Type</th>
<th>*</th>
<th>+</th>
</tr>
</thead>
<tbody>
<tr>
<td>str</td>
<td>Repeat</td>
<td>Append</td>
</tr>
<tr>
<td>int</td>
<td>Multiplication</td>
<td>Addition (sum)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Get / assign</td>
</tr>
<tr>
<td>==</td>
<td>Equals</td>
</tr>
<tr>
<td>.</td>
<td>Call function</td>
</tr>
<tr>
<td>in</td>
<td>Answer true/false if string is inside another string</td>
</tr>
</tbody>
</table>
# String functions

- `res = ‘Team Blue Devil wins’`

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Result with string <code>res = ‘Team Blue Devils’</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>res.upper()</code></td>
<td>returns string upper case version of <code>string res</code></td>
<td>TEAM BLUE DEVILS</td>
</tr>
<tr>
<td><code>res.count(‘am’)</code></td>
<td>returns int number of (non-overlapping) occurrences of <code>sub</code> in <code>s</code></td>
<td>1</td>
</tr>
<tr>
<td><code>res.find(‘m’)</code></td>
<td>returns int first index at which <code>sub</code> occurs in <code>s</code> or -1 if no occurrence</td>
<td>3</td>
</tr>
<tr>
<td><code>res.split()</code></td>
<td>returns list of <code>s</code> split on whitespace</td>
<td>[‘Team’, ‘Blue’, ‘Devils’]</td>
</tr>
<tr>
<td><code>res.split(‘l’)</code></td>
<td>returns list of <code>s</code> split on <code>sep</code>, a delimiter</td>
<td>[‘Team B’, ‘ue Devi’, ‘s’]</td>
</tr>
<tr>
<td><code>res.strip()</code></td>
<td>returns copy of <code>res</code> without leading and trailing whitespace</td>
<td>Team Blue Devils</td>
</tr>
<tr>
<td><code>len(res)</code></td>
<td>returns the number of characters in the string (including white spaces)</td>
<td>16</td>
</tr>
<tr>
<td><code>res.isupper()</code></td>
<td>returns true if all the characters are in upper case</td>
<td>False</td>
</tr>
</tbody>
</table>
Sir Tim Berners-Lee

I want you to realize that, if you can imagine a computer doing something, you can program a computer to do that.

Unbounded opportunity... limited only by your imagination. And a couple of laws of physics.

- HTTP, URL, HTML
  - How, Why, What, When?
Loops

<table>
<thead>
<tr>
<th>for element in sequence:</th>
<th>while statement:</th>
</tr>
</thead>
<tbody>
<tr>
<td>do this........</td>
<td>do this.........</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
</tr>
</tbody>
</table>

For each element in the sequence please do the following...

While the statement is true please do the following...

Great for list with well known number of elements

Great for all other cases with unknown number of elements
Under the hood of loops

- Loops is the answer to how you go through a string

```
adr = "pete@duke.edu"
for letter in adr:
    print letter

result = ""
for char in red:
    if char.isupper():
        result = result + char
print result
```