CS1110 Spring 2018 Announcements
Check course page for announcements!
http://www.cs.cornell.edu/courses/cs1110/2018sp

ENGRG 1010. AEW workshops still space
  – can enroll through Student Center
  • 1-credit S/U course
  • 2-hour weekly workshop
  • work on related problem sets
Full? Or need a different time?
https://tinyurl.com/aew-request

Things to Do Before Next Class

• Read Sections 3.4-3.11

• If you haven’t already:
  • install Anaconda Python & Komodo on your machine
  • play around with python a bit!

Function Calls
• Python supports expressions with math-like functions
  • A function in an expression is a function call
  • Will explain the meaning of this later
• Function expressions have the form:
  \[ \text{fun}(x,y,\ldots) \]
  \begin{itemize}
    \item function
    \item argument
  \end{itemize}
• Some math functions built into Python:
  \begin{itemize}
    \item \text{round}(2.34)
    \item \text{max}(a+3,24)
  \end{itemize}

Modules
• “Libraries” of functions and variables
• To access a module, use the import command:
  \text{import <module name>}
  Can then access functions like this:
  \text{<module name>.<function name>(<arguments>)}
• Example:
  >>> import math
  >>> math.cos(2.0)
  -0.4161468365471424

Module Variables
• Modules can have variables, too
• Can access them like this:
  \text{<module name>.<variable name>}
• Example:
  >>> import math
  >>> math.pi
  3.141592653589793
Interactive Shell vs. Modules

Python Interactive Shell
- Type python at command line
- Type commands after `>>>`
- Python executes statements when import is called

Module
- Written in text editor
- Loaded through `import`
- Python executes statements when import is called

Section 2.4 in your textbook discusses a few differences

**my_module.py**

```python
# my_module.py

"""This is a simple module. It shows how modules work"

x = 1+2
x = 3*x

Docstring (note the Triple Quotes) Acts as a multi-line comment Useful for code documentation

Commands Executed on import
```

**You Must import**

```python
C:\> python
>>> import my_module
>>> my_module.x
9
```

**Dangers of Importing Everything**

```python
C:\> python
>>> e = 12345
>>> from math import *
>>> e
2.718281828459045
```

```python
 e was overwritten!
```

**Modules vs. Scripts**

<table>
<thead>
<tr>
<th>Module</th>
<th>Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provides functions, variables</td>
<td>Behaves like an application</td>
</tr>
<tr>
<td>Import it into Python shell</td>
<td>Run it from command line</td>
</tr>
</tbody>
</table>

Files look the same. Difference is how you use them.

**Next Time: Defining Functions**

- Today we created a module with a variable
- Have not discussed how to make a function
- **Example:**
  ```python
  >>> import math
  >>> math.cos(2.0)
  -0.4161468365471424
  ```
  we want to make functions like this