Modeling Storage in Python

- **Global Space**
  - What you “start with”
  - Stores global variables
  - Also modules & functions!
  - Lasts until you quit Python

- **Call Frame**
  - Variables in function call
  - Deleted when call done

- **Heap Space**
  - Where “folders” are stored
  - Have to access indirectly

Functions and Global Space

- A function definition…
  - Creates a global variable (same name as function)
  - Creates a folder for body
  - Puts folder id in variable

Variable vs. Call

```python
>>> to_centigrade
<function to_centigrade at 0x100498de8>

```  

```python
def to_centigrade(x):
    return 5*(x-32)/9.0

```  

Modules and Global Space

- Importing a module:
  - Creates a global variable (same name as module)
  - Puts contents in a folder
  - Module variables
  - Module functions
  - Puts folder id in variable

```
from keyword import dumps
```  

Modules vs Objects

<table>
<thead>
<tr>
<th>Module</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>math</td>
<td>p</td>
</tr>
<tr>
<td>id2</td>
<td>id3</td>
</tr>
<tr>
<td>pi 3.141592</td>
<td>5.0</td>
</tr>
<tr>
<td>e 2.718281</td>
<td>Point</td>
</tr>
<tr>
<td>functions</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>math</td>
<td>p</td>
</tr>
<tr>
<td>id2</td>
<td>id3</td>
</tr>
<tr>
<td>pi 3.141592</td>
<td>5.0</td>
</tr>
<tr>
<td>e 2.718281</td>
<td>Point</td>
</tr>
<tr>
<td>functions</td>
<td></td>
</tr>
</tbody>
</table>

```python
math.pi
math.cos(1)
```  

```python
p.x
p.clamp(-1,1)
```
When Do We Need to Draw a Folder?

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Variable holds a</td>
<td>• Variable holds a</td>
</tr>
<tr>
<td>* function</td>
<td>* base type</td>
</tr>
<tr>
<td>* module</td>
<td>* bool, int, float, str</td>
</tr>
<tr>
<td>* object</td>
<td>* (more???)</td>
</tr>
</tbody>
</table>

Recall: Call Frames

1. Draw a frame for the call
2. Assign the argument value to the parameter (in frame)
3. Execute the function body
   - Look for variables in the frame
   - If not there, look for global variables with that name
4. Erase the frame for the call

Call: to_centigrade(80.0)

Aside: What Happens Each Frame Step?

• The instruction counter *always* changes
• The contents only *change* if
  * You add a new variable
  * You change an existing variable
  * You delete a variable
• If a variable refers to a *mutable object*
  * The contents of the folder might change

Call Frames vs. Global Variables

• The specification is false:
  ```python
def swap(a,b):
    """Swap vars a & b""
    tmp = a
    a = b
    b = tmp
  
  >>> a = 1
  >>> b = 2
  >>> swap(a,b)
```

Function Access to Global Space

• All function definitions are in some module
• Call can access global space for *that module*
  * math.cos: global for math
  * temperature.to_centigrade uses global for temperature
• But *cannot* change values
  * Assignment to a global makes a new local variable!
  * Why we limit to constants

Global Variables

```python
a
b
```

Call Frame

```
swap
```

Call Frames and Objects

• Mutable objects can be altered in a function call
  * Object vars hold names!
  * Folder accessed by both global var & parameter

Example:

```python
def incr_x(q):
    q.x = q.x + 1

>>> p = Point(0,0,0)
>>> incr_x(p)
```

Global Space

```
a
```

Heap Space

```
<table>
<thead>
<tr>
<th>id5</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0.0</td>
</tr>
<tr>
<td>Point</td>
<td></td>
</tr>
</tbody>
</table>
```

Call Frame

```
incr_x
```