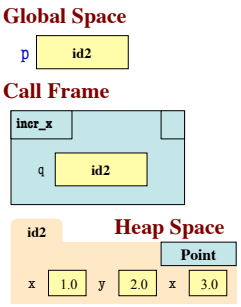
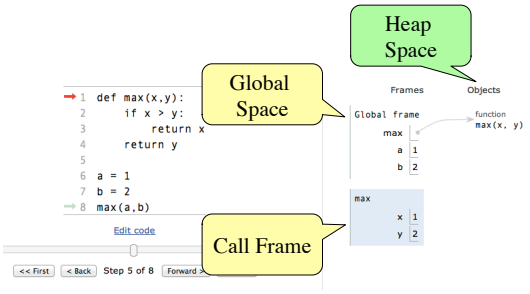
Modeling Storage in Python

- **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



Memory and the Python Tutor



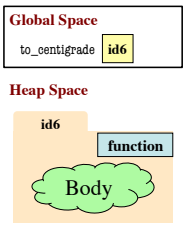
Functions and Global Space

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

```
>>> to_centigrade
<fun to_centigrade at 0x100498de8>
>>> to_centigrade(32)
0.0
```

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

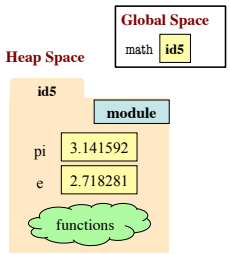
Body



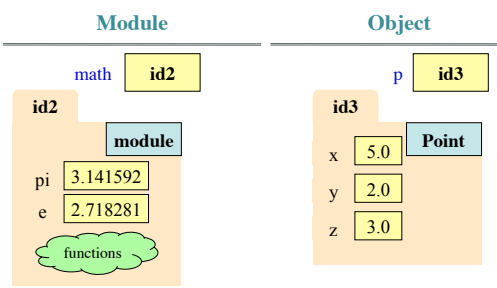
Modules and Global Space

- 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 dumps contents to global space

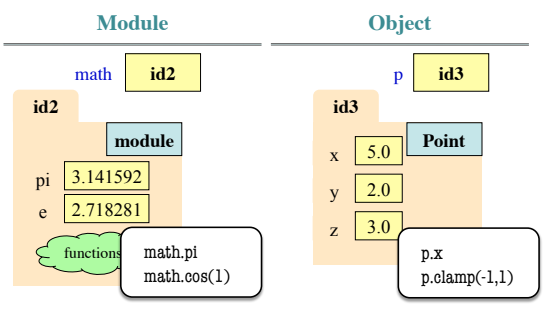
```
import math
```



Modules vs Objects

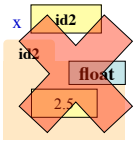


Modules vs Objects



When Do We Need to Draw a Folder?

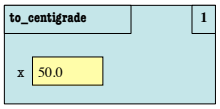
Yes	No
• Variable holds a ▪ function ▪ module ▪ object ▪ (more????)	• Variable holds a ▪ base type ▪ bool, int, float, str



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(50.0)



What is happening here?

Only at the End!

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

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

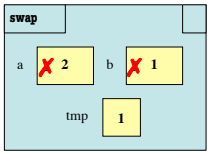
- The specification is false:

```
def swap(a,b):
    """Swap vars a & b"""
1   tmp = a
2   a = b
3   b = tmp
```

```
>>> a = 1
>>> b = 2
>>> swap(a,b)
```

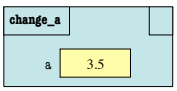
Global Variables

Call Frame



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




```
# globals.py
"""Show how globals work"""
a = 4 # global space

def change_a():
    a = 3.5 # local variable
```

Call Frames and Objects

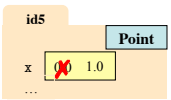
- Mutable objects can be altered in a function call
  - Object vars hold names!
  - Folder accessed by both global var & parameter
- **Example**:

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

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

Global Space

Heap Space



Call Frame