Lecture 4

Visualizing Functions
### Function Definition

```python
def foo(a, b):
    """Do something
    Param a: number
    Param b: number"
    x = a
    y = b
    return x*y+y
```

### Function Call

```python
>>> x = 2
>>> foo(3, 4)
```

What is in the box?
A Motivating Example

Function Definition

```python
def foo(a, b):
    # Do something
    Param a: number
    Param b: number

    x = a
    y = b
    return x*y+y
```

Function Call

```python
>>> x = 2
>>> foo(3, 4)
```

What is in the box?

A: 2
B: 3
C: 16
D: Nothing!
E: I do not know
## A Motivating Example

### Function Definition

```python
def foo(a, b):
    """Do something
    Param a: number
    Param b: number"
    x = a
    y = b
    return x*y+y
```

### Function Call

```python
>>> x = 2
>>> foo(3, 4)
```

What is in the box?

<table>
<thead>
<tr>
<th>A: 2</th>
<th>CORRECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>B: 3</td>
<td></td>
</tr>
<tr>
<td>C: 16</td>
<td></td>
</tr>
<tr>
<td>D: Nothing!</td>
<td></td>
</tr>
<tr>
<td>E: I do not know</td>
<td></td>
</tr>
</tbody>
</table>

9/18/15
How Do Functions Work?

- **Function Frame**: Representation of function call
- A **conceptual model** of Python

Draw parameters as variables (named boxes)

- Number of statement in the function body to execute next
- **Starts with 1**

Draw template on a piece of paper
Textbook

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

This Class

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

Definition: to_centigrade(x) → 50.0

Call: to_centigrade(50.0)
Example: to_centigrade(50.0)

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
Example: \texttt{to\_centigrade(50.0)}

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

\begin{verbatim}
def to_centigrade(x):
    return 5*(x-32)/9.0
\end{verbatim}
Example: `to_centigrade(50.0)`

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

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

Executing the return statement

The return terminates; no next line to execute
Example: \texttt{to\_centigrade(50.0)}

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

```
def to\_centigrade(x):
    return 5\*(x-32)/9.0
```

But don’t actually erase on an exam.
Call Frames vs. Global Variables

The specification is a **lie**:

```python
def swap(a, b):
    """Swap global a & b"""
    tmp = a
    a = b
    b = tmp
```

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

---

Global Variables

- **a**: 1
- **b**: 2

Call Frame

```
swap
```

---

Visualizing Functions
The specification is a lie:

```python
def swap(a, b):
    """Swap global a & b""
    tmp = a
    a = b
    b = tmp
```

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

9/18/15

Visualizing Functions
The specification is a lie:

```python
def swap(a, b):
    """Swap global a & b""
    tmp = a
    a = b
    b = tmp
```

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

Visualizing Functions

Global Variables

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
</tr>
</tbody>
</table>

Call Frame

<table>
<thead>
<tr>
<th>swap</th>
</tr>
</thead>
<tbody>
<tr>
<td>a 2</td>
</tr>
<tr>
<td>b 2</td>
</tr>
<tr>
<td>tmp 1</td>
</tr>
</tbody>
</table>
The specification is a **lie**:

```python
def swap(a, b):
    """Swap global a & b""
    tmp = a
    a = b
    b = tmp
```

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

```
>>> a = 1
>>> b = 2
>>> swap(a, b)
```
Call Frames vs. Global Variables

The specification is a lie:

```python
def swap(a, b):
    """Swap global a & b""
    tmp = a
    a = b
    b = tmp
```

```python
>>> a = 1
>>> b = 2
>>> swap(a, b)
```
Visualizing Frames: The Python Tutor

```python
1. def max(x, y):
2.     if x > y:
3.         return x
4.     return y
5.
6. a = 1
7. b = 2
8. max(a, b)
```

Visualizing Functions

- Frames
  - Global frame
    - max
    - a
    - b

- Objects
  - function max(x, y)
  - x
  - y

[Diagram showing the visualization of frames and objects in Python Tutor]
Visualizing Frames: The Python Tutor

```
def max(x, y):
    if x > y:
        return x
    return y

a = 1
b = 2
max(a, b)
```

Global Space

Call Frame
Visualizing Frames: The Python Tutor

```
def max(x, y):
    if x > y:
        return x
    return y

a = 1
b = 2
max(a, b)
```

Global Space

Call Frame

Variables from second lecture go in here
Visualizing Frames: The Python Tutor

```python
1 def max(x, y):
2     if x > y:
3         return x
4     return y
5
6 a = 1
7 b = 2
8 max(a, b)
```

Missing line numbers!
Visualizing Frames: The Python Tutor

Line number marked here (sort-of)

```python
1 def max(x, y):
2     if x > y:
3         return x
4     return y
5
6 a = 1
7 b = 2
8 max(a, b)
```

Missing line numbers!
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

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

def show_a():
    print a  # shows global
```

Global Space (for globals.py)

```
a  4
```

```
show_a  1
```
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

```python
# globals.py
"""Show how globals work"""

a = 4  # global space

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

#### Function Definition

```
def foo(a,b):
    """Do something
    Param x: a number
    Param y: a number""

x = a
y = b
return x*y+y
```

#### Function Call

```
>>> x = foo(3,4)
```

What does the frame look like at the start?
Which One is Closest to Your Answer?

A: 

```
<table>
<thead>
<tr>
<th>foo</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>4</td>
</tr>
</tbody>
</table>
```

B: 

```
<table>
<thead>
<tr>
<th>foo</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>4</td>
</tr>
</tbody>
</table>
```

C: 

```
<table>
<thead>
<tr>
<th>foo</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>4</td>
</tr>
<tr>
<td>x</td>
<td>3</td>
</tr>
</tbody>
</table>
```

D: 

```
<table>
<thead>
<tr>
<th>foo</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>4</td>
</tr>
<tr>
<td>x</td>
<td>3</td>
</tr>
<tr>
<td>y</td>
<td></td>
</tr>
</tbody>
</table>
```
Which One is Closest to Your Answer?

A: 

```
A: foo
  a 3  b 4
```

B: 

```
B: foo
  a 3  b 4
```

C: 

```
C: foo
  a 3
  x 3
```

E: 

```
\_ (ツ)_ \/
```

9/18/15 Visualizing Functions 25
def foo(a,b):
    """Do something
    Param x: a number
    Param y: a number"
    x = a
    y = b
    return x*y+y

>>> x = foo(3,4)

B:
**Exercise Time**

---

**Function Definition**

```python
def foo(a,b):
    
    """Do something
    
    Param x: a number
    Param y: a number"

1    x = a
2    y = b
3    return x*y+y
```

**Function Call**

```python
>>> x = foo(3,4)
```

**B:**

What is the next step?
Which One is Closest to Your Answer?

A:  
```
<table>
<thead>
<tr>
<th>foo</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>4</td>
</tr>
</tbody>
</table>
```

B:  
```
<table>
<thead>
<tr>
<th>foo</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>4</td>
</tr>
</tbody>
</table>
```

C:  
```
<table>
<thead>
<tr>
<th>foo</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>4</td>
</tr>
<tr>
<td>x</td>
<td>3</td>
</tr>
</tbody>
</table>
```

D:  
```
<table>
<thead>
<tr>
<th>foo</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>4</td>
</tr>
<tr>
<td>x</td>
<td>3</td>
</tr>
<tr>
<td>y</td>
<td></td>
</tr>
</tbody>
</table>
```
Exercise Time

**Function Definition**

```python
def foo(a, b):
    """Do something
    Param x: a number
    Param y: a number"
    x = a
    y = b
    return x*y+y
```

**Function Call**

```python
>>> x = foo(3, 4)
```

C:

![Diagram of function call and variable assignment](image)
Exercise Time

Function Definition

def foo(a,b):
    """Do something
    Param x: a number
    Param y: a number"
    x = a
    y = b
    return x*y+y

Function Call

>>> x = foo(3,4)

C:

What is the next step?
Which One is Closest to Your Answer?

A:

\[
\begin{array}{c|c|c}
\text{foo} & 3 \\
\hline
a & 3 & b \\
& 3 & y \\
& x & 4
\end{array}
\]

B:

\[
\begin{array}{c|c|c}
\text{foo} & 3 \\
\hline
a & 3 & b \\
& 3 & y \\
& x & 4
\end{array}
\]

RETURN

C:

\[
\begin{array}{c|c|c}
\text{foo} & 3 \\
\hline
a & 3 & b \\
& 3 & y \\
& x & 4
\end{array}
\]

RETURN 16

D:

\[
\begin{array}{c|c|c}
\text{foo} & 3 \\
\hline
a & 3 & b \\
& 3 & y \\
& x & 4
\end{array}
\]

RETURN

ERASE THE FRAME
Exercise Time

Function Definition

```python
def foo(a,b):
    """Do something"
    Param x: a number
    Param y: a number"
    x = a
    y = b
    return x*y+y
```

Function Call

```python
>>> x = foo(3,4)
```

A:

```
<table>
<thead>
<tr>
<th>foo</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>4</td>
</tr>
<tr>
<td>x</td>
<td>3</td>
</tr>
<tr>
<td>y</td>
<td>4</td>
</tr>
</tbody>
</table>
```
**Exercise Time**

### Function Definition

```python
def foo(a, b):
    """Do something
    Param x: a number
    Param y: a number"
    x = a
    y = b
    return x*y+y
```

### Function Call

```python
>>> x = foo(3,4)
```

A: What is the next step?
Which One is Closest to Your Answer?

A: 

```
| foo | 3 |
```

RETURN 16

B: 

```
| foo | 3 |
| a   | 3 |
| x   | 3 |
| b   | 4 |
| y   | 4 |
```

RETURN 16

C: 

```
| foo |
| a   | 3 |
| b   | 4 |
| x   | 3 |
| y   | 4 |
```

RETURN 16

D: 

```
| foo |
```

ERASE THE FRAME

9/18/15 Visualizing Functions 34
### Exercise Time

**Function Definition**

```python
def foo(a, b):
    """Do something
    Param x: a number
    Param y: a number"
    x = a
    y = b
    return x*y+y
```

**Function Call**

```python
>>> x = foo(3, 4)
```

**C:**

```
foo

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

RETURN 16
```
Exercise Time

**Function Definition**

```python
def foo(a, b):
    # Do something
    # Param x: a number
    # Param y: a number
    x = a
    y = b
    return x*y + y
```

**Function Call**

```python
>>> x = foo(3, 4)
```

C:

<table>
<thead>
<tr>
<th>foo</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>RETURN</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>16</td>
</tr>
</tbody>
</table>

What is the next step?
Which One is Closest to Your Answer?

A: foo

B: RETURN 16

C: foo

D: x 16
# Exercise Time

## Function Definition

```python
def foo(a,b):
    
    """Do something
    Param x: a number
    Param y: a number"
    
    x = a
    y = b
    return x*y+y
```

## Function Call

```python
>>> x = foo(3,4)
```

```
D: x 16
```

* ERASE THE FRAME *
# Exercise Time

## Function Definition

```python
def foo(a, b):
    """Do something
    Param x: a number
    Param y: a number"
    
    x = a
    y = b
    return x*y+y
```

## Function Call

```python
>>> x = foo(3, 4)
```

Variable in global space

D:

|x|
---|---|
1 | $x = a$ |
2 | $y = b$ |
3 | return $x \times y + y$ |

Erase the frame
A1: The Module urllib2

- Module urllib2 is used to read web pages
  - Function urlopen creates a url object
  - `u = urllib2.urlopen('http://www.cornell.edu')`

- url has a method called `read()`
  - Returns contents of web page
  - **Usage**: `s = u.read() # s is a string`