Call Frames
Final Review
Spring 2018
CS 1110
The Big Issue

• Cannot answer questions on this topic without
  ▪ drawing variables
  ▪ drawing frames for function calls
  ▪ drawing objects when they are created

• Learning to do this is useful in general
  ▪ Helps you “think like a computer”
  ▪ Easier to find errors in your programs.
What Do You Need to Know?

- Major topics
  - *local variables* *(in a function body)*
  - *function call* *(call frames, call stack)*
  - *class folders, inheritance, and constructor calls*
- Examples from previous exams
  - Question 3 on prelim 1
  - Question 6 on prelim 2
Important

- Code execution is an important part of the final
- You need to know how to
  - draw variables
  - draw call frames
  - draw objects

The purpose of such questions on executing statements with constructs and function calls is to test your understanding of how Python programs are executed.
The Frame (box) for a Function Call

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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>function name</td>
<td>instruction counter</td>
</tr>
<tr>
<td>parameters</td>
<td></td>
</tr>
<tr>
<td>local variables (when assigned)</td>
<td></td>
</tr>
</tbody>
</table>

- Number of statement in the function body to execute **next**
- Starts with function’s first executable statement

Draw parameters as variables (named boxes)
def countVowels(word):
    """Returns: The number of vowels in the string s.
    Precondition: s is a string""
    x = 0
    count = 0
    while x < len(word):
        if word[x] in ['a','e','i','o','u']:
            count += 1
        x += 1
    return count

Call: e = countVowels('hi')
def countVowels(word):
    """Returns: The number of vowels in the string s.
    Precondition: s is a string""
    x = 0
    count = 0
    while x < len(word):
        if word[x] in ['a','e','i','o','u']:
            count += 1
        x += 1
    return count

Call: e = count_vowels('hi')
```python
def countVowels(word):
    # Returns: The number of vowels in the string s.
    # Precondition: s is a string
    x = 0
    count = 0
    while x < len(word):
        if word[x] in ['a', 'e', 'i', 'o', 'u']:
            count += 1
        x += 1
    return count
```

Iterative Example

Call: e = count_vowels('hi')

Global Space:
def countVowels(word):

    """Returns: The number of vowels in the string s.
    Precondition: s is a string"

    x = 0
    count = 0
    while x < len(word):
        if word[x] in ['a','e','i','o','u']:
            count += 1
        x += 1
    return count

Call: e = count_vowels('hi')
```python
def countVowels(word):
    """Returns: The number of vowels in the string s.
    Precondition: s is a string""
    x = 0
    count = 0
    while x < len(word):
        if word[x] in ['a','e','i','o','u']:
            count += 1
        x += 1
    return count
```

**Iterative Example**

Call: `e = count_vowels('hi')`

```
Call Stack:

<table>
<thead>
<tr>
<th>countVowels</th>
<th>1, 2, 3, 4, 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>word</td>
<td>'hi'</td>
</tr>
<tr>
<td>count</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>0</td>
</tr>
</tbody>
</table>

Global Space:
```
def countVowels(word):
    """Returns: The number of vowels in the string s.
    Precondition: s is a string"""
    x = 0
    count = 0
    while x < len(word):
        if word[x] in ['a', 'e', 'i', 'o', 'u']:
            count += 1
        x += 1
    return count

call: e = count_vowels('hi')
def countVowels(word):
    """Returns: The number of vowels in the string s.
    Precondition: s is a string""
    x = 0
    count = 0
    while x < len(word):
        if word[x] in ['a', 'e', 'i', 'o', 'u']:
            count += 1
        x += 1
    return count

Call: e = count_vowels('hi')
def countVowels(word):
    """Returns: The number of vowels in the string s.
    Precondition: s is a string""
    x = 0
    count = 0
    while x < len(word):
        if word[x] in ['a', 'e', 'i', 'o', 'u']:
            count += 1
        x += 1
    return count

Call: e = count_vowels('hi')
def countVowels(word):
    """Returns: The number of vowels in the string s.
    Precondition: s is a string"""
    x = 0
    count = 0
    while x < len(word):
        if word[x] in ['a','e','i','o','u']:
            count += 1
        x += 1
    return count

Call: e = countVowels('hi')
def countVowels(word):
    """Returns: The number of vowels
    in the string s.
    Precondition: s is a string""
    x = 0
    count = 0
    while x < len(word):
        if word[x] in ['a', 'e', 'i', 'o', 'u']:
            count += 1
        x += 1
    return count

Call: e = count_vowels('hi')
def countVowels(word):
    """Returns: The number of vowels in the string s.
    Precondition: s is a string""
    x = 0
    count = 0
    while x < len(word):
        if word[x] in ['a','e','i','o','u']:
            count += 1
        x += 1
    return count

Call: e = count_vowels('hi')
def countVowels(word):
    """Returns: The number of vowels in the string s.
    Precondition: s is a string"""
    x = 0
    count = 0
    while x < len(word):
        if word[x] in ['a','e','i','o','u']:
            count += 1
        x += 1
    return count

Call: e = count_vowels('hi')
def countVowels(word):
    """Returns: The number of vowels in the string s.
    Precondition: s is a string"""
    x = 0
    count = 0
    while x < len(word):
        if word[x] in ['a','e','i','o','u']:
            count += 1
            x += 1
    return count

Call: e = count_vowels('hi')
Subcall Example

Global Space:

```python
def printWeather(wkList):
    for item in wkList:
        dayWeather(item)

def dayWeather(day):
    if day == 'Sunny':
        print ('Time for a picnic!')
    if day == 'Rainy':
        print ('Grab your umbrella!')
```

Call: `printWeather(a)`

Heap Space:

```
Call Stack:
1
```

```
printWeather
id1
wkList
```

```
list
id1
```

```
0
'Rainy'
1
'Sunny'
```
def printWeather(wkList):
    for item in wkList:
        dayWeather(item)

def dayWeather(day):
    if day == 'Sunny':
        print ('Time for a picnic!')
    if day == 'Rainy':
        print ('Grab your umbrella!')

Call: printWeather(a)
def printWeather(wkList):
    for item in wkList:
        dayWeather(item)

def dayWeather(day):
    if day == 'Sunny':
        print ('Time for a picnic!')
    elif day == 'Rainy':
        print ('Grab your umbrella!')

Call: printWeather(a)
def printWeather(wkList):
    for item in wkList:
        dayWeather(item)

def dayWeather(day):
    if day == 'Sunny':
        print ('Time for a picnic!')
    if day == 'Rainy':
        print ('Grab your umbrella!')

Call: printWeather(a)
Subcall Example

Global Space:

```
def printWeather(wkList):
    for item in wkList:
        dayWeather(item)

def dayWeather(day):
    if day == 'Sunny':
        print ('Time for a picnic!')
    if day == 'Rainy':
        print ('Grab your umbrella!')
```

Call: printWeather(a)

Heap Space:

```
<table>
<thead>
<tr>
<th>id1</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
```

Call Stack:

```
printWeather
  id1
wkList
  id1
item
    'Rainy'
dayWeather
  id1
day
    'Rainy'
```

Subcall Example

Call

```
printWeather(a)
```

Heap Space:

```
<table>
<thead>
<tr>
<th>id1</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
```

Call Stack:

```
printWeather
  id1
wkList
  id1
item
    'Rainy'
dayWeather
  id1
day
    'Rainy'
```
Subcall Example

Global Space:
def printWeather(wkList):
    for item in wkList:
        dayWeather(item)
def dayWeather(day):
    if day == 'Sunny':
        print ('Time for a picnic!')
    if day == 'Rainy':
        print ('Grab your umbrella!')

Call: printWeather(a)

Output:
Grab your umbrella!

Heap Space:

<table>
<thead>
<tr>
<th>Global Space</th>
<th>Heap Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>id1</td>
</tr>
<tr>
<td>id1</td>
<td>list</td>
</tr>
</tbody>
</table>

Call Stack:

- printWeather
- wkList | id1 |
- item | ‘Rainy’ |
- dayWeather
- day | ‘Rainy’ |
- RETURN | None |
Global Space:
def printWeather(wkList):
    for item in wkList:
        dayWeather(item)
def dayWeather(day):
    if day == 'Sunny':
        print ('Time for a picnic!')
    if day == 'Rainy':
        print ('Grab your umbrella!')

Call: printWeather(a)

Output: Grab your umbrella!
Global Space:
def printWeather(wkList):
    for item in wkList:
        dayWeather(item)
def dayWeather(day):
    if day == 'Sunny':
        print ('Time for a picnic!')
    if day == 'Rainy':
        print ('Grab your umbrella!')

Call: printWeather(a)

Output:
Grab your umbrella!
Subcall Example

Global Space:

```
def printWeather(wkList):
    for item in wkList:
        dayWeather(item)

def dayWeather(day):
    if day == 'Sunny':
        print ('Time for a picnic!')
    if day == 'Rainy':
        print ('Grab your umbrella!')
```

Call: `printWeather(a)`

Output:
Grab your umbrella!
def printWeather(wkList):
    for item in wkList:
        dayWeather(item)

def dayWeather(day):
    if day == 'Sunny':
        print ('Time for a picnic!')
    if day == 'Rainy':
        print ('Grab your umbrella!')

Call: printWeather(a)

Output:
Grab your umbrella!
Global Space: a

def printWeather(wkList):
    for item in wkList:
        dayWeather(item)

def dayWeather(day):
    if day == 'Sunny':
        print ('Time for a picnic!')
    if day == 'Rainy':
        print ('Grab your umbrella!')

Call: printWeather(a)

Output:
Grab your umbrella!
Subcall Example

Global Space:
def printWeather(wkList):
    for item in wkList:
        dayWeather(item)
def dayWeather(day):
    if day == 'Sunny':
        print ('Time for a picnic!')
    if day == 'Rainy':
        print ('Grab your umbrella!')

Call: printWeather(a)

Output:
Grab your umbrella!
Time for a picnic!
Global Space:

```
def printWeather(wkList):
    for item in wkList:
        dayWeather(item)
```  

```
def dayWeather(day):
    if day == 'Sunny':
        print ('Time for a picnic!')
    if day == 'Rainy':
        print ('Grab your umbrella!')
```  

Heap Space:

```
0 'Rainy'
1 'Sunny'
```  

Subcall Example

Call:
```
printWeather(a)
```  

Output:
Grab your umbrella!
Time for a picnic!
Subcall Example

Global Space:
def printWeather(wkList):
  for item in wkList:
    dayWeather(item)
def dayWeather(day):
  if day == 'Sunny':
    print ('Time for a picnic!')
  if day == 'Rainy':
    print ('Grab your umbrella!'

Call: printWeather(a)

Output:
Grab your umbrella!
Time for a picnic!
Subcall Example

Global Space:

```
def printWeather(wkList):
    for item in wkList:
        dayWeather(item)
```

Heap Space:

```
def dayWeather(day):
    if day == 'Sunny':
        print ('Time for a picnic!')
    if day == 'Rainy':
        print ('Grab your umbrella!')
```

Call: `printWeather(a)`

Output:

Grab your umbrella!
Time for a picnic!
Diagramming Objects (Folders)

**Object Folder**
- **Folder Name (arbitrary)**
- **id4**
  - **classname**
- **Instance Attributes**
  - Draw attributes as named box w/ value

**Class Folder**
- **No folder name**
- **classname**
- **Class Attributes Method Names**
- Parameters are required in methods
3 steps to evaluating the call C(args)

- **Create a new folder** (object) of class C
  - Give it with a unique name (any number will do)
  - Folder goes into heap space
- Execute the *method* `__init__(args)`
- Yield *the name* of the object as *the value*
  - A constructor call is an *expression*, not a command
  - Does not put name in a variable unless you **assign it**
Diagramming Subclasses

**Important Details:**
- Make sure you put the superclass-name in parentheses
- Do not duplicate inherited methods and attributes
- Include initializer and and other special methods (as applicable)
- Method parameters are required
- Class attributes are a box with (current) value

---

**Declared in Superclass:**
- Class Attributes
- Method Names

**Declared in Subclass:**
- Class Attributes
- Method Names
Two Example Classes

```python
class A(object):
    x=3
    y=5
    def __init__(self,y):
        self.y = y
    def f(self):
        return self.g()
    def g(self):
        return self.x+self.y

class B(A):
    y=4
    z=10
    def __init__(self,x,y):
        super().__init__(y)
        self.x = x
    def g(self):
        return self.x+self.z
    def h(self):
        return 42
```
Class Folders

These folders will still exist in the following slides, but will not be redrawn; they exist in the heap space along with the object folders.
Constructor Examples

Global Space:

class A(object):
    x = 3
    y = 5
    def __init__(self, y):
        self.y = y

class B(A):
    y = 4
    z = 10
    def __init__(self, x, y):
        super().__init__(y)
        self.x = x

Call: a = A(1)
      b = B(7, 3)
Constructor Examples

Global Space:

```python
class A(object):
    x = 3
    y = 5
    def __init__(self, y):
        self.y = y

class B(A):
    y = 4
    z = 10
    def __init__(self, x, y):
        super().__init__(y)
        self.x = x
```

Heap Space:

```
call a = A(1)
    b = B(7, 3)
```

Call Stack:

```
A.__init__

self = id8
id8

y = 1

RETURN
```
Constructor Examples

Global Space:

class A(object):
    x = 3
    y = 5
    def __init__(self,y):
        self.y = y

class B(A):
    y = 4
    z = 10
    def __init__(self,x,y):
        super().__init__(y)
        self.x = x

Call: a = A(1)
b = B(7, 3)
Constructor Examples

Global Space:

class A(object):
    x = 3
    y = 5
    def __init__(self, y):
        self.y = y

class B(A):
    y = 4
    z = 10
    def __init__(self, x, y):
        super().__init__(y)
        self.x = x

Heap Space:

Call Stack:

1

2

3

Call: a = A(1)
b = B(7, 3)
Constructor Examples

Global Space:

class A(object):
    x = 3
    y = 5
    def __init__(self, y):
        self.y = y

class B(A):
    y = 4
    z = 10
    def __init__(self, x, y):
        super().__init__(y)
        self.x = x

Call: a = A(1)
      b = B(7, 3)

Heap Space:

Call Stack:
Constructor Examples

```python
class A(object):
    x = 3
    y = 5
    def __init__(self, y):
        self.y = y

class B(A):
    y = 4
    z = 10
    def __init__(self, x, y):
        super().__init__(y)
        self.x = x

Call: a = A(1)
b = B(7, 3)
```

Call Stack:
Constructor Examples

Global Space:

```python
class A(object):
    x = 3
    y = 5
    def __init__(self, y):
        self.y = y

class B(A):
    y = 4
    z = 10
    def __init__(self, x, y):
        super().__init__(y)
        self.x = x
```

Heap Space:

```
Call: a = A(1)
b = B(7, 3)
```

Call Stack:
Constructor Examples

Global Space:

class A(object):
    x = 3
    y = 5
    def __init__(self, y):
        self.y = y

class B(A):
    y = 4
    z = 10
    def __init__(self, x, y):
        super().__init__(y)
        self.x = x

Call: a = A(1)
b = B(7, 3)

Heap Space:

Call Stack:

1

A.__init__

self

y

A

id8

A

id8

RETURN

1

B.__init__

self

y

x

B

id4

B

id4

RETURN

2, 3

A.__init__

self

y

A

id4

A

id4

RETURN

1
Global Space:

```
class A(object):
    x = 3
    y = 5
    def __init__(self, y):
        self.y = y

class B(A):
    y = 4
    z = 10
    def __init__(self, x, y):
        super().__init__(y)
        self.x = x
```

Heap Space:

```
A

id8

y

1

B

id4

y

3

x

7
```

Call Stack:

```
A.__init__

self

id8

y

1

RETURN

id8

B.__init__

self

id4

x

7

RETURN

id4

A.__init__

self

id4

y

3

RETURN

id4
```

Call: 

```
a = A(1)
b = B(7, 3)
```

1

2

3

4