Global Space

- Global Space
  - What you “start with”
  - Stores global variables
  - Lasts until you quit Python

\[ x = 4 \]
Enter Heap Space

- **Global Space**
  - What you “start with”
  - Stores global variables
  - Lasts until you quit Python

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

```
x = 4
p = shape.Point2(1,2)
q = shape.Point2(10,7)
```

*p & q* live in Global Space. Their folders live on the Heap.
Calling a Function Creates a Call Frame

What’s in a Call Frame?
• Boxes for parameters **at the start of the function**
• Boxes for variables local to the function **as they are created**

```python
def adjust_x_coord(pt, n):
    pt.x = pt.x + n

x = 4
p = shape.Point2(1,2)
adjust_x_coord(p, x)
```
Calling a Function Creates a Call Frame

What’s in a Call Frame?

• Boxes for parameters at the start of the function.
• Boxes for variables local to the function as they are created.

```python
def adjust_x_coord(pt, n):
    pt.x = pt.x + n

x = 4
p = shape.Point2(1, 2)
adjust_x_coord(p, x)
```

Global Space

| x | 4 |
| n | None |

Heap Space

| id1 | Point2 |
| x   | 1 5  |
| y   | 2   |

Call Frame

```
adjust_x_coord
```

```
pt
```

```
4
```
Putting it all together

• **Global Space**
  - What you “start with”
  - Stores global variables
  - Lasts until you quit Python

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

• **Call Frames**
  - Parameters
  - Other variables local to function
  - Lasts until function returns
2 Points Make a Line!

```
start = shape.Point2(0,0)
stop = shape.Point2(0,0)
print("Where does the line start?")
x = input("x: ")
start.x = int(x)
y = input("y: ")
start.y = int(y)
print("Where does the line stop?")
x = input("x: ")
stop.x = int(x)
y = input("y: ")
stop.y = int(y)
print("The line stops at ("+x+ ","+y+ ").")
```

Where does the line start?
x: 1
y: 2
The line starts at (1,2).

Where does the line stop?
x: 4
y: 6
The line stops at (4,6).
Redundant Code is BAAAD!

```
start = shape.Point2(0,0)
stop = shape.Point2(0,0)

print("Where does the line start?")
x = input("x: ")
start.x = int(x)
y = input("y: ")
start.y = int(y)
print("The line starts at (" + x + "," + y + ").")

print("Where does the line stop?")
x = input("x: ")
stop.x = int(x)
y = input("y: ")
stop.y = int(y)
print("The line stops at (" + x + "," + y + ").")
```
Let’s make a function!

def configure(pt, role):
    print("Where does the line " + role + "?")
    x = input("x: ")
    pt.x = int(x)
    y = input("y: ")
    pt.y = int(y)
    print("The line " +role+ "s at ("+x+ ","+y+ ")." )

start = shape.Point2(0,0)
stop = shape.Point2(0,0)
configure(start, "start")
configure(stop, "stop")
def configure(pt, role):
    print("Where does the line " + role + "?"")
    x = input("x: ")
    pt.x = int(x)
    y = input("y: ")
    pt.y = int(y)
    print("The line " + role + "s at ("+x+ "," +y+ ").")

start = shape.Point2(0,0)
stop = shape.Point2(0,0)
configure(start, "start")
configure(stop, "stop")
Yay, Helper Functions!

```python
def get_coord(name):
    x = input(name+"": ")
    return str(x)  # Actual bug I wrote in my code. Not staged!

def configure(pt, role):
    print("Where does the line "+ role + "?"
    pt.x = get_coord("x")
    pt.y = get_coord("y")
    print("The line "+ role+ "s at ("+x+ ","+y+ ")." )

start = shape.Point2(0,0)
stop = shape.Point2(0,0)
configure(start, "start")
configure(stop, "stop")
```

Only have to fix 1 line. In the first version, I would have had to fix it in 4 places!
Frames and Helper Functions

- Functions can call each other!
- Each call creates a new call frame
- Writing the same several lines of code in 2 places? Or code that accomplishes some conceptual sub-task? Or your function is getting too long? Write a helper function! Makes your code easier to:
  - Read
  - Write
  - Edit
  - Debug
def get_coord(name):
    x = input(name+":
    return int(x)

def configure(pt, role):
    print("Where does the line " + role + "?")
    pt.x = get_coord("x")
    pt.y = get_coord("y")
    print("The line " +role+ "s at ("+str(pt.x)+
          ","+str(pt.y)+ ").")

start = shape.Point2(0,0)
configure(start, "start")
Q: what do you do next?

def get_coord(name):
    x = input(name+":\")
    return int(x)

def configure(pt, role):
    print("Where does the line "+role+"?")
    pt.x = get_coord("x")
    pt.y = get_coord("y")
    print("The line "+role+"s at ("+str(pt.x)+","+str(pt.y)+")")

start = shape.Point2(0,0)
call configure(start, "start")
```python
def get_coord(name):
    x = input(name+":\n")
    return int(x)

def configure(pt, role):
    print("Where does the line \"+
role+\"\?")
    pt.x = get_coord("x")
    pt.y = get_coord("y")
    print("The line \"+role+"s at ("+str(pt.x)+
    ","+str(pt.y)+\"\).")

start = shape.Point2(0,0)
configure(start, "start")
```

Drawing Frames for Helper Functions (2)

Call Frames

Not done! Do not cross out!!
def get_coord(name):
    x = input(name+"": ")
    return int(x)

def configure(pt, role):
    print("Where does the line " + role + "?")
    pt.x = get_coord("x")
    pt.y = get_coord("y")
    print("The line " +role+ "s at ("+str(pt.x)+
        ","+str(pt.y)+ ")." )

start = shape.Point2(0,0)
configure(start, "start")
def get_coord(name):
    x = input(name+": ")
    return int(x)

def configure(pt, role):
    print("Where does the line "+role+"?"")
    pt.x = get_coord("x")
    pt.y = get_coord("y")
    print("The line "+role+"s at ("+str(pt.x)+",")"+str(pt.y)+");")

start = shape.Point2(0,0)
configure(start, "start")
The Call Stack

• Functions frames are “stacked”
  ▪ Cannot remove one above w/o removing one below

• Python must keep the entire stack in memory
  ▪ Error if it cannot hold stack ("stack overflow")
Q: what does the call stack look like at this point in the execution of the code?

def f3():
    print("f3")

def f2():
    print("f2")
f3()
f3()

def f1():
    print("f1")
f2()
f1()
Q: what does the call stack look like at this point in the execution of the code?

<table>
<thead>
<tr>
<th>def f3():</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>print(“f3”)</td>
<td>f1</td>
<td>f1</td>
<td>f1</td>
<td>f1</td>
<td>f1</td>
</tr>
<tr>
<td>def f2():</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>print(“f2”)</td>
<td>f2</td>
<td>f2</td>
<td>f2</td>
<td>f2</td>
<td>f2</td>
</tr>
<tr>
<td>f3()</td>
<td>f3</td>
<td>f3</td>
<td>f3</td>
<td>f3</td>
<td>f3</td>
</tr>
<tr>
<td>f3()</td>
<td>f3</td>
<td>f3</td>
<td>f3</td>
<td>f3</td>
<td>f3</td>
</tr>
<tr>
<td>f3()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>def f1():</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>print(“f1”)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f2()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f1()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
def get_coord(name):
    x = input(name+": ")
    return int(x)

def configure(pt, role):
    print("Where does the line "+role+"?"
    pt.x = get_coord("x")
    pt.y = get_coord("y")
    print("The line "+role+"s at ("+x+","+y+")

start = shape.Point2(0,0)
configure(start, "start")
import

- Creates a global variable (same name as module)
- Puts variables, functions in a folder
- Puts folder id in variable

Global Space

```
import math
```

Heap Space

```
math  id5

id5
pi  3.141592
e  2.718281
functions
```

22
```python
>>> import math
>>> math.pi
3.141592653589793

>>> p = shapes.Point3(5,2,3)
>>> p.x
5

Global Space

math id5

p id3

Heap Space

id5

math

pi 3.141592

e 2.718281

functions

id3

Point3

x 5

y 2

z 3
Storage in Python

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

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

• **Call Frame Stack**
  - Parameters
  - Other variables local to function
  - Lasts until function returns