Lecture 9 Announcements

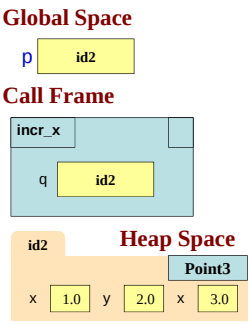
- A1 is graded. If your A1 is not perfect, your first grade is a 1.
  - This is a counter for how many times you have submitted.
  - It is not a permanent grade, can resubmit until March 2nd.
- Review the announcements from the end of Lecture 6 for policies:

http://www.cs.cornell.edu/courses/cs1110/2017sp/lectures/02-14-17/presentation-06.pdf

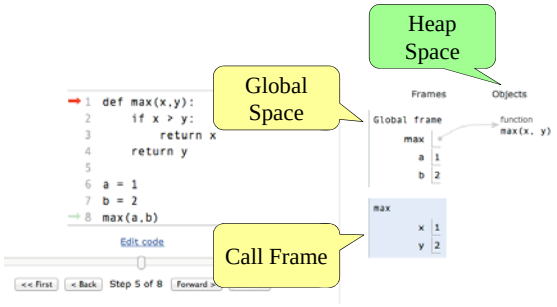
- Read section 2.3 of A1 carefully to understand how to revise.
- Lab 5 is released (note there is no Lab 4)
- Reading: Section 10.1-10.2, 10.4-10.6

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



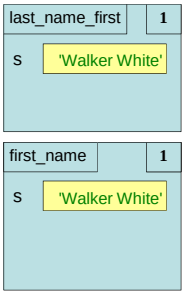
Memory and the Python Tutor



Frames and Helper Functions

```
def last_name_first(s):
    """Precondition: s in the form
    <first-name> <last-name>"""
1   first = first_name(s)
2   last = last_name(s)
3   return last + ',' + first

def first_name(s):
    """Prec: see last_name_first"""
1   end = s.find(' ')
2   return s[0:end]
```

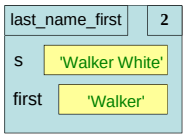
Call: last_name_first('Walker White'):



Frames and Helper Functions

```
def last_name_first(s):
    """Precondition: s in the form
    <first-name> <last-name>"""
1   first = first_name(s)
2   last = last_name(s)
3   return last + ',' + first

def first_name(s):
    """Prec: see last_name_first"""
1   end = s.find(' ')
2   return s[0:end]
```

Call: last_name_first('Walker White'):

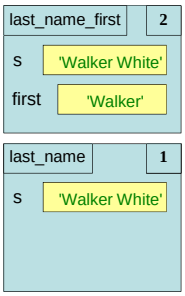


ERASE WHOLE FRAME

Frames and Helper Functions

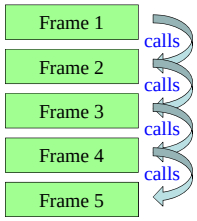
```
def last_name_first(s):
    """Precondition: s in the form
    <first-name> <last-name>"""
1   first = first_name(s)
2   last = last_name(s)
3   return last + '.' + first

def last_name(s):
    """Prec: see last_name_first"""
1   end = s.rfind(' ')
2   return s[end+1:]
```

Call: last_name_first('Walker White'):

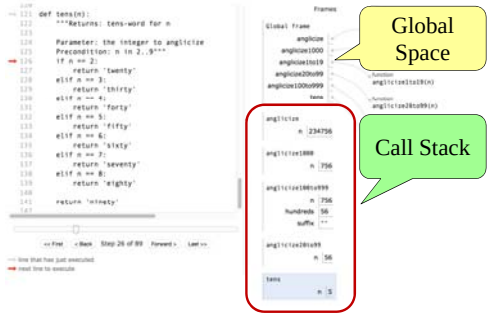


The Call Stack

- Functions are "stacked"
  - Cannot remove one above w/o removing one below
  - Sometimes draw bottom up (better fits the metaphor)
- Stack represents memory as a "high water mark"
  - Must have enough to keep the **entire stack** in memory
  - Error if cannot hold stack



Online Python Tutor Example



Errors and the Call Stack

```
# error.py

def function_1(x,y):
    return function_2(x,y)

def function_2(x,y):
    return function_3(x,y)

def function_3(x,y):
    return x/y # crash here

print function_1(1,0)
```

Crashes produce the call stack:

```
Traceback (most recent call last):
  File "error.py", line 20, in <module>
    print function_1(1,0)
  File "error.py", line 8, in function_1
    return function_2(x,y)
  File "error.py", line 12, in function_2
    return function_3(x,y)
  File "error.py", line 16, in function_3
    return x/y
```

Make sure you can see line numbers in Komodo. Preferences ➔ Editor

Errors and the Call Stack

```
#
def function_1(x,y):
    return function_2(x,y)

def function_2(x,y):
    return function_3(x,y)

def function_3(x,y):
    return x/y # crash here

p
```

Script code. Global space

Where error occurred (or where was found)

Crashes produce the call stack:

```
Traceback (most recent call last):
  File "error.py", line 20, in <module>
    print function_1(1,0)
  File "error.py", line 8, in function_1
    return function_2(x,y)
  File "error.py", line 12, in function_2
    return function_3(x,y)
  File "error.py", line 16, in function_3
    return x/y
```

Make sure you can see line numbers in Komodo. Preferences ➔ Editor

Try-Except is Very Versatile

```
def isfloat(s):
    """Returns: True if string s
    represents a float"""
    try:
        x = float(s)
        return True
    except:
        return False
```

Conversion to a float might fail

If attempt succeeds, string s is a float

Otherwise, it is not

Try-Except and the Call Stack

```
# recover.py

def function_1(x,y):
    try:
        return function_2(x,y)
    except:
        return float('inf')

def function_2(x,y):
    return function_3(x,y)

def function_3(x,y):
    return x/y # crash here
```

- Error "pops" frames off stack
  - Starts from the stack bottom
  - Continues until it sees that current line is in a try-block
  - Jumps to except, and then proceeds as if no error