http://www.cs.cornell.edu/courses/cs1110/2018sp

Lecture 9: Memory in Python

CS 1110

Introduction to Computing Using Python

Cornell CIS

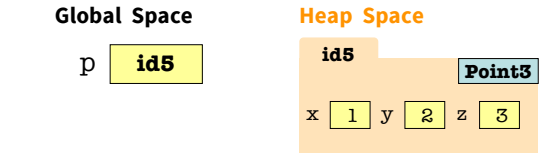
COMPUTING AND INFORMATION SCIENCE

[E. Andersen, A. Bracy, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White]

Folders Live on the Heap

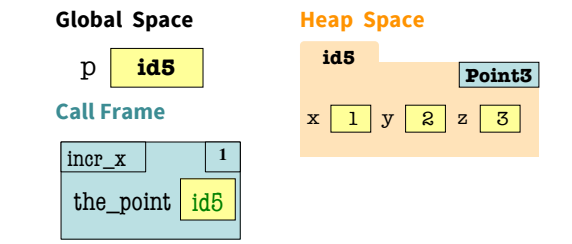
```
p = shapes.Point3(1,2,3)
```

p lives in the Global Space. Its folder lives on the Heap.



Calling a Function Creates a Call Frame

```
p = shapes.Point3(1,2,3)
incr_x(p)
```



Frames and Helper Functions

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

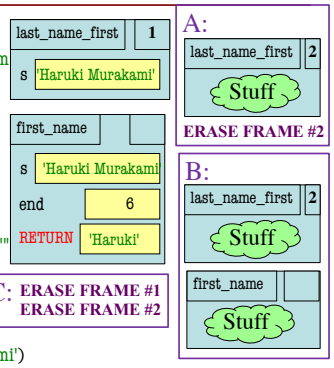
```
def last_name_first(s):
    """Precondition: s in the form
    <first-name> <last-name>"""
1   first = first_name(s)
2   last = last_name(s)
3   return last + ',' + first
```

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

rfind gets the *last* instance of substring

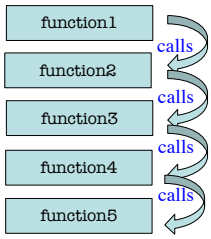
Question: What Happens Next?

```
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"""
4   end = s.find(' ')
5   return s[0:end]

last_name_first('Haruki Murakami')
```



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")



Call Stack Example (4)

```
1  def happy_birthday():
2      print("Happy Birthday")
3  def dear():
4      print("Dear James")
5  def to_you():
6      print("to you")
7  def line_with_name():
8      happy_birthday()
9      dear()
10 def basic_line():
11     happy_birthday()
12     to_you()
13 def song():
14     basic_line()
15     basic_line()
16     line_with_name()
17     basic_line()
18 song()
```

Call Frame Stack

song	14
basic_line	11
happy_birthday	
RETURN	None

```
Happy Birthday
```

Call Stack Example (7)

```
1  def happy_birthday():
2      print("Happy Birthday")
3  def dear():
4      print("Dear James")
5  def to_you():
6      print("to you")
7  def line_with_name():
8      happy_birthday()
9      dear()
10 def basic_line():
11     happy_birthday()
12     to_you()
13 def song():
14     basic_line()
15     basic_line()
16     line_with_name()
17     basic_line()
18 song()
```

Call Frame Stack

song	14
basic_line	12
to_you	
RETURN	None

```
Happy Birthday
to you
```

Errors and the Call Stack, in Color!

```
1  def happy_birthday():
2      print("Happy Birthday")
3  def dear():
4      print("Dear "+name)
5  def to_you():
6      print("to you")
7  def line_with_name():
8      happy_birthday()
9      dear()
10 def basic_line():
11     happy_birthday()
12     to_you()
13 def song():
14     basic_line()
15     basic_line()
16     line_with_name()
17     basic_line()
18 song()
```

```
Happy Birthday
to you
Happy Birthday
to you
Happy Birthday
Traceback (most recent call last):
  File "birthday_error.py", line 18, in <module>
    song()
  File "birthday_error.py", line 16, in song
    line_with_name()
  File "birthday_error.py", line 9, in line_with_name
    dear()
  File "birthday_error.py", line 4, in dear
    print("Dear "+name)
NameError: name 'name' is not defined
```

Script code. Global space

Where error was found

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

```
INCHES_PER_FT = 12
def get_feet(ht_in_inches):
    return ht_in_inches // INCHES_PER_FT
```

Body

Global Space

INCHES_PER_FT	12
get_feet	id6

Heap Space

id6 — function — Body

See for yourself: https://tinyurl.com/get-feet

Modules and Global Space

import

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

```
import math
```

Global Space

math	id5

Heap Space

id5 — module

pi	3.141592	e	2.718281

functions

Storage in Python (Final Version!)

- **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