CSE 8A: Intro to Programming in Python

Fall 2021

Lecture 12 - Stack frames

UC San Diego

Announcement

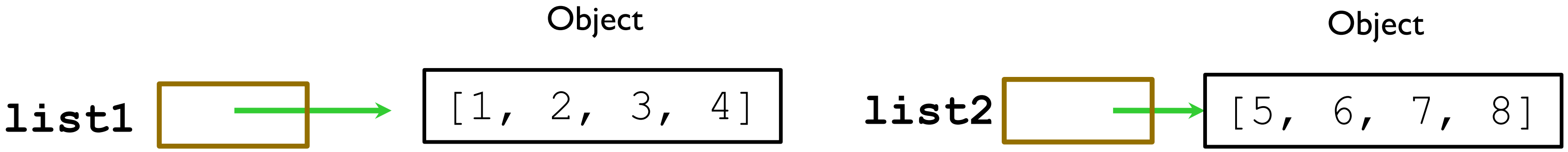
- Midterm graded
    - See it on canvas (gradescope)

Topics for Today

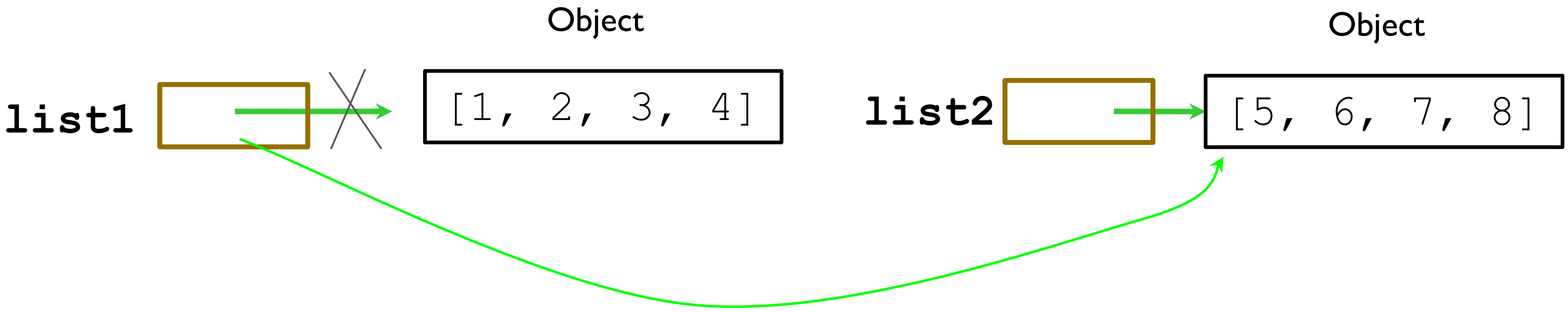
- exercises for references and methods
- Stack frames and scopes

CS Concepts: References

```
list1 = [1, 2, 3, 4]
list2 = [5, 6, 7, 8]
```



```
list1 = list2
```



Memory Model

implicit lst = nums

```
def clear_list(lst):
    for idx in range(len(lst)):
        lst[idx - 1] = 0

nums = [11, 12, 13]
clear_list(nums)
```

arg

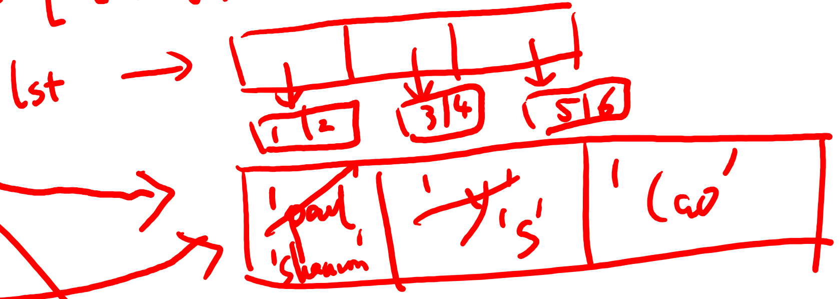
[link here]

Exercise: Modifying a String within a Function

What will happen when you run the program below?

lst = [[1,2], [3,4], [5,6]]

```
def updates(name):
    name[0] = 'shannon'
    name[1] = 's'
name = ['paul', 'y', 'cao']
updates(name)
print(name)
```



[✓ A) The program will print `['shannon','s','cao']`]

B) The program will print `['paul,'y','cao']`

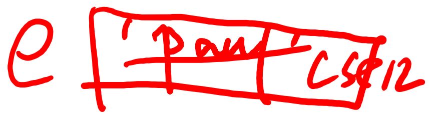
C) The program will print `['shannon', 's']`

D) The program will print `['paul', 'y']`

D) The program will throw an error

for e in name:

e = 'cse12'



Methods

- Objects in Python are packed with functionalities --> methods
- You access object’s methods using its reference or an object with the dot operator
    - `‘   paul   ’.strip()`
    - `nums = [1, 2, 3]`
      `nums.append(4)`

'paul'

'␣␣Paul␣␣␣'.
strip()

Exercise: Methods

What is the value of nums?

```
nums = []
nums.append(2)
nums.append(2**2)
nums.append(8)
nums.append(2**4)
print(nums)
```

A) [2, 2**2, 8, 2**4]

[✓ B)] [2, 4, 8, 16]

C) [16, 8, 4, 2]

D) [2**4, 8, 2**2, 2]

E) Error: Cannot append to an empty list!

Exercise: Methods

What will be printed?

```
chars = "a;b;c;d;".split(";")
len(chars)
```

A) 3 B) 4 [✓ C) 5] D) 8

E) Error: Cannot split using a semicolon (;)

Exercise: Methods

What will be printed?

```
words = "you:-are:-awesome".split(":-")
words
```

[✓ A)] ['you', 'are', 'awesome']

B) ['you:', 'are:', 'awesome:']

C) ['you-', 'are-', 'awesome-']

D) ['you:-', 'are:-', 'awesome:-']

E) Error: Cannot split using more than one character

Exercise: Methods

What will be printed?

```
"+".join([‘1’, ‘2’, ‘3’])
```

[✓ A) '1+2+3']

B) '123'

C) 6

D) '1+2+3+'

E) Error: Cannot join a list of integers

Small Challenge: Methods

What is the value of new_message?

```
>>> message = "live love laugh"
>>> words = message.split(" ")
>>> lst = []
>>> for i in range(len(words)-1, 0, -1):
                lst.append(words[i])
>>> new_message = ",".join(lst)
>>> new_message
```

range(2 ,0, -1)

new_msg 'la, lo'

message "li lo la'

words → 'li' 'lo' 'la'

[✓ A)] 'laugh,love'

B) 'live,love,laugh'

C) 'laugh,love,live'

D) 'live,love'

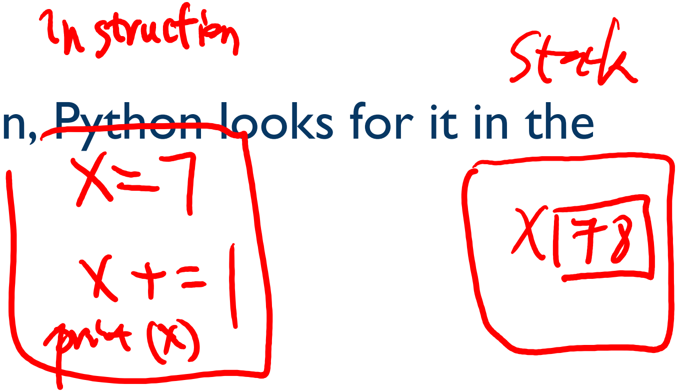
E) I don’t know! :(

lst → 'la' 'lo'

Stack Frames

Every time a function is invoked (i.e., called), the invocation gets a new "frame" for holding variables

- The parameters also exist in a frame
- When a variable name is used within a function, Python looks for it in the current frame first
- We call these variables local variables



Global frame

- There is always one global frame that all functions can access
- When a variable name is used, Python looks two places:
  1. the function invocation's frame (first)
  2. the global frame (only if not found before)

Stack Frame

What happens when a function is called?

```
def foo(num):
  if num > 0:
    return 1
  else:
    return -1

foo(4)
val = 7
foo(val)
```

Exercise: Call Stack with One Stack Frame

What will happen when we run this code?

```
def set_x():
      x = 100

print(x)
```

A) 100 will be printed

B) 0 will be printed

C) Error: variable x is not defined

D) I don’t know! :(

Exercise: Call Stack with One Stack Frame

What will happen when we run this code?

```
def set_x():
    x = 100

print(x)
```

A) 100 will be printed

B) 0 will be printed

C) Error: variable x is not defined

D) I don’t know! :(

Functions do not execute unless they are called! (only in Python)

Exercise: Call Stack with One Stack Frame

What will happen when we run this code?

```
def set_x():
      x = 100


set_x()
print(x)
```

A) 100 will be printed

B) 0 will be printed

C) Error: variable x is not defined

D) I don’t know! :(

Exercise: Call Stack with One Stack Frame

What will happen when we run this code?

```
def set_x():
      x = 100


set_x()
print(x)
```

A) 100 will be printed

B) 0 will be printed

C) Error: variable x is not defined

D) I don’t know! :(

Variables in a function’s stack frame only exists when the function frame exists

Exercise: Call Stack with One Stack Frame

What will happen when we run this code?

```
def count():
    x = 1
    x += 1
    print(x)

count()
count()
count()
```

A) The program will print 2, 3, 4

B) The program will print 2, 2, 2

C) Error: variable x is not defined

D) I don’t know! :(

Exercise: Call Stack with One Stack Frame

What will happen when we run this code?

```
def count():
    x = 1
    x += 1
    print(x)

count()
count()
count()
```

A) The program will print 2, 3, 4

B) The program will print 2, 2, 2

C) Error: variable x is not defined

D) I don’t know! :(

Variables start fresh every time a function is called again