On your way in...(on the side table)

Pick-up:
1. Your graded Homework 0
2. Homework 2, due Monday
3. POGIL 16: Lists
4. POGIL 11: Predefined Functions
Welcome to CS 134!

Introduction to Computer Science
Iris Howley

-Iteration-

Spring 2019
Iris’ Thursday TA Hours Shifted This Week

• Thursday, February 14 Iris Office Hours
  • 4-5:30p
  • (instead of 1-2:30p)
FRIDAY IS WINTER CARNIVAL
(NO CLASSES)
Swapping Values in Python

• $i, j = j, i$

**Is equivalent to (and much better than):**

• $\text{tmp} = i$
• $i = j$
• $j = \text{tmp}$
Lists

• Look at POGIL 16 – Lists, Question 2

    fruits = ['apple', 'banana', 'cantalope', 'pear']
    print(fruits[0])

• Discuss your responses with a partner to 2a, 2b, 2c
Lists

• Look at POGIL 16 – Lists, Question 2

    fruits = ['apple', 'banana', 'cantelope', 'pear']
    print(fruits[0])

• What will fruits[-1] print?
  • ‘pear’ Why?

• What about fruits[-2]?
  • ‘cantelope’ Why?
Boolean False Values

• All these will evaluate to False:
  - `bool(0)`
  - `bool("")`
  - `bool([])`
  - `bool(None)`

• What do these have in common?
  - Note: `bool([0,0,0,0])` will evaluate to True. Why?

• What is the shortest String you can make?
REVISITING

Abstraction makes programming GREAT

(abstraction, encapsulation, etc.)
From Monday

• When we did this:
  - `myb = Book('title', 'author name', 'opening line')`
  - `myb.open()`
  - `myb.read_book()`

• Did we have to know:
  - ...exactly what the Book initializer looked like?
  - ...the lines of code in .open()?
  - ...that .read_book() has a loop that reads through each letter and inserts a hyphen using ‘reading += letter + “-”’?

What part is the public interface?
What part is the private implementation?
TODAY'S LESSON

Everything in python is an object

(numbers, bools, strings are immutable objects.)
Mutability
A Balloon Metaphor...

>>> 3
A Balloon Metaphor...

>>> x = 3
A Balloon Metaphor...

>>> x = x + 1

NUMBERS ARE IMMUTABLE
A Balloon Metaphor...

>>> [5,16,18]
A Balloon Metaphor...

>>> mylist = [5, 16, 18]
A Balloon Metaphor...

```python
>>> mylist.append('dogge')

mylist
```

LISTS ARE MUTABLE
A Tale of Two Mutabilities...

- $x = 3$
- $y = 3$
- $x == y$
  - True
- $x$ is $y$
  - True

- $l = [1,2,3]$
- $m = [1,2,3]$
- $l == m$
  - True
- $l$ is $m$
  - False
A Balloon Metaphor...

```python
>>> l = [1,2,3]
>>> m = [1,2,3]
```
A Tale of Two Mutabilities...

- $x = 3$
- $y = x$
- $x == y$
  - True
- $x$ is $y$
  - True

- $l = [1, 2, 3]$
- $m = l$
- $l == m$
  - True
- $l$ is $m$
  - True
A Balloon Metaphor...

```python
>>> l = [1, 2, 3]
>>> m = l
```
__all__ special variable

- If the variable starts/ends with "__ __" it’s a special python variable
- We saw this with __name__

- __all__ is another special variable
- Whatever is stored in __all__ is imported when the user types:
  - from _____ import *
- Any function/variable/etc. that’s not included in __all__ can be imported explicitly
  - from <module name> import <not-included-in-star-variable/function>
QUESTIONS?
Leftover Slides
Lists

- myList = ['apples', 'oranges', 55]
- myList[0]
  - 'apples'
- myList[2]
  - 55
- myList[-1]
  - 55
- myList[-2]
  - 'oranges'
Mutability

• An immutable object can be changed after it’s created, and an immutable object cannot

• Numbers, booleans, strings in python are IMMUTABLE
  - x = 3
  - x = x + 1 \(\rightarrow\) Once you run this, x is now pointing at a different number, 4

• Lists are MUTABLE
  - mylist = [5,16]
  - mylist.append(18) \(\rightarrow\) we add an element to mylist, but it’s still pointing at the same list, it’s not a new list