building our own

RECURSIVE DATA STRUCTURES

WEEK-AT-A-GLANCE

Introduction to Computer Science
Iris Howley
"It would be no small advantage if every college were thus located in a Glow/Canvas LMS."

Adapted from Thoreau (1844)
<table>
<thead>
<tr>
<th>Date</th>
<th>Assignment/Activity</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mon Apr 13, 2020</td>
<td>Lecture 19: Class Attributes &amp; Inheritance</td>
<td>due by 9:30am</td>
</tr>
<tr>
<td></td>
<td>POGIL: Inheritance (Encouraged, pre-lecture)</td>
<td>due by 9:30am</td>
</tr>
<tr>
<td></td>
<td>Homework 05</td>
<td>due by 11pm</td>
</tr>
<tr>
<td>Wed Apr 15, 2020</td>
<td>Lecture 20: Inheritance &amp; Methods</td>
<td>due by 9:30am</td>
</tr>
<tr>
<td></td>
<td>POGIL: Calling Super Methods (Encouraged, pre-lecture)</td>
<td>due by 9:30am</td>
</tr>
<tr>
<td>Thu Apr 16, 2020</td>
<td>Lab 7: Creating a Class</td>
<td>due by 11pm</td>
</tr>
<tr>
<td>Fri Apr 17, 2020</td>
<td>Lecture 21: Ciphers</td>
<td>due by 9:30am</td>
</tr>
<tr>
<td></td>
<td>POGIL: Type Conversion (Encouraged, pre-lecture)</td>
<td>due by 9:30am</td>
</tr>
<tr>
<td></td>
<td>Quiz 1</td>
<td>due by 11:59pm</td>
</tr>
<tr>
<td>Mon Apr 20, 2020</td>
<td>Brief Overview of this Week</td>
<td>due by 9 am</td>
</tr>
<tr>
<td></td>
<td>Lecture 22: Introduction to Recursion</td>
<td>due by 9:30am</td>
</tr>
<tr>
<td></td>
<td>POGIL: Recursion (Optional, pre-lecture)</td>
<td>due by 9:30am</td>
</tr>
<tr>
<td></td>
<td>Homework 06</td>
<td>due by 11pm</td>
</tr>
<tr>
<td>Wed Apr 22, 2020</td>
<td>Lecture 23: Fruitful and Graphical Recursion</td>
<td>due by 9:30am</td>
</tr>
<tr>
<td>Thu Apr 23, 2020</td>
<td>Lab 8: Classes and Inheritance</td>
<td>due by 11pm</td>
</tr>
<tr>
<td>Fri Apr 24, 2020</td>
<td>Lecture 24: Graphical Recursion II</td>
<td>due by 9:30am</td>
</tr>
<tr>
<td></td>
<td>Quiz 2</td>
<td>due by 11:59pm</td>
</tr>
<tr>
<td>Mon Apr 27, 2020</td>
<td>Homework 07</td>
<td>due by 11pm</td>
</tr>
</tbody>
</table>
HAPPENING THIS WEEK

• There is no quiz this week!
  • (unless you're watching this on Friday, April 24)

• Homework 7 is due Monday, April 27
  • Homework 8 will be released Wednesday

• Lab 9 was released Friday, April 24
  • And it's due Thursday, April 30

• Lab 10 (extra credit) will be released Friday, May 1
THIS WEEK’S LESSON

Building our own recursive data structures

(We have the tools to build our own data structures)
LECTURES THIS WEEK

• Monday
  o Week Overview
  o Building our own data structures
  o Elements

• Wednesday
  o Element Methods
  o Introducing the LinkedList wrapper class
  o Building out the LinkedList class

• Friday
  o Binary Trees
  o Using Binary Trees
  o Extra Credit Lab Intro
Steps for Recursion

1. Know when to stop.
2. Decide how to take one, repeated step.
3. Break the journey down into that step plus a smaller journey.

Recursive Approach

- **REDUCE** the problem to smaller subproblem(s) (smaller version(s) of itself)
- **DELEGATE** the smaller problems to the recursion fairy (*formally known as induction hypothesis*) and assume they're solved correctly
- **COMBINE** the solution(s) of the smaller subproblems to reach/return the solution
Prior to lecture videos...

Complete:

1. POGIL Activities: Element & LinkedList & Binary Trees
   • available under Glow > Modules
   • also posted to the course website under Remote Lectures

• Best done prior to watching lectures!
• Good for working with a partner (virtually, too!)
  • But will work without a partner, as well
Prior to this week's lessons...

Be able to:

1. Build & instantiate new classes & objects
   • ...with attributes and methods

2. Implement recursive functions
NO BOOK CHAPTERS THIS WEEK
Consult POGILs, slides, Lecture Notes

Highly recommended
QUESTIONS?

Please contact me!
Building Our Own Data Structures

Introduction to Computer Science
Iris Howley
TODAY’S LESSON
Building our own list class

(We have the tools to build our own data structures)
What is a list?

```python
class list(object):
    list() -> new empty list
    list(iterable) -> new list initialized from iterable's items

Methods defined here:

    __add__(self, value, /)
    Return self+value.

    __contains__(self, key, /)
    Return key in self.

    __delitem__(self, key, /)
    Delete self[key].

    __eq__(self, value, /)
```
What is a list?
What is a list?

What is the last elephant holding onto?

None
What is a list?

```python
class Element:
    _value
    _next
```

9 -> 17 -> 2012
SPECIAL METHODS

• We're familiar with `__str__(self)` which is called implicitly with `str(object)` and `__init__(self)` which is called implicitly when instantiating objects

• POGIL 27. Special Methods gives you broader exposure to more!

• Think: *every* built-in function we call + *every* operator
Special Methods

- `len(object)`
- `ex: len('hello, world!')`
  - Returns the length of the sequence, if possible

```python
def __len__(self):
    # Write your own code
    # that calculates & returns
    # the length of the object, self
```
Special Methods

- `indexableSequence[index]`
- `ex: myList[5]`
  - Returns the object located at `index` of the `indexableSequence`, if possible

```
def __getitem__(self, index):
    # Write your own code
    # that finds the item at index
    # and returns it
```
Special Methods

• `indexableSequence[index] = val`
• `ex: myList[5] = 'Something else.'`
  ▪ Assigns the object located at `index` to the value, `val`, if possible

  ▪ `def __setitem__(self, index, val):
    # Write your own code
    # that finds the item at `index`
    # and sets its value to `val`
Special Methods

- `val in collection`
- `ex: 's' in 'iris'`
  - Returns True if `val` exists in `collection`, False otherwise

```python
def __contains__(self, val):
    # Write your own code
    # that finds if val exists in self
    # and returns True if found
```
Special Methods

• for item in iterableCollection:
• ex: for word in wordList:
  ▪ Iterate across the items of the list

  ▪ def __iter__(self):
    o # Write your own code
    o # to yield the next object in self
Some Common List Functions

- `def append(self, val):` Add `val` to the end of the list
- `def extend(self, seq):` Extend list by adding elements of `seq`
- `def pop(self, index=None):` Returns and removes the object located at `index` of the list, if possible
- `def reverse(self):` Reverse the list (destructively)
- `def sort(self):` Sort the list (destructively)
Common Features of All Classes

• Docstrings
• __all__
• __slots__
• Hidden attributes ➔ @property, @____.setter
Tuples, Strings, other built-in types aren’t particularly special!

You can build your own!
QUESTIONS?

Please contact me!
Elements of a Linked List

Introduction to Computer Science
Iris Howley
TODAY’S LESSON

A private class holding values in a list

(Building the Element class)
Linked Lists – Element Class

• See example code on the course website!

`LinkedList.py`
Testing `@property` + initializer in interactive python

```python
>>> from LinkedList import Element
>>> ele1 = Element('a')
>>> ele1.value
'a'
>>> ele1.next
>>> ele2 = Element('b', ele1)
>>> ele2.value
'b'
>>> ele2.next
<LinkedList.Element object at 0x10feee8d0>
>>> ele2.next.value
'a'
```
Testing `@next.setter` in interactive python

```python
>>> ll = Element(3)
>>> ll.next = Element(7)
>>> ll.value
3
>>> ll.next
<LinkedList.Element object at 0x10feeebe0>
>>> ll.next.value
7
>>> ll.next.next = Element(1715)
>>> ll.next.next.value
1715
>>> ll.next.next.next.value
3
>>> ll.next.next.next.value
7
>>> ll.next.next.next.value
1715
>>> ll.next.next.next.value
3
>>> ll.next
<LinkedList.Element object at 0x10feeebe0>
```

Careful! We can make an infinite list by connecting the end to the beginning!
QUESTIONS?

Please contact me!
Leftover Slides
Steps for Recursion

• Know when to stop.
• Decide how to take one step.
• Break the journey down into that step plus a smaller journey.