Announcements

- Homework 8 due Wednesday 4/15 @ 11:59pm [small]
- Project 4 due Thursday 4/23 @ 11:59pm (big)
- Project 4/4 due Tuesday 4/14 @ 11:59pm
- Early point #1: Questions 1-12 submitted (correctly) by Friday 4/17 @ 11:59pm
- Early point #2: All questions (including Extra Credit) by Wednesday 4/22 @ 11:59pm
- If you want the first early submission point, you need to:
  - Pass the tests for the designated questions
  - Run `python3 ok --submit`
  - Log on to http://ok.cs61a.org and create a group with your partner

Ray Tracing

A technique for displaying a 3D scene on a 2D screen by tracing a path through every pixel

The Scene:

<table>
<thead>
<tr>
<th>Camera</th>
<th>Light</th>
<th>Sphere at origin</th>
</tr>
</thead>
</table>

Dramatization:

<table>
<thead>
<tr>
<th>Camera</th>
<th>Light</th>
<th>Sphere</th>
<th>Distance to Sphere</th>
</tr>
</thead>
</table>

Information Hiding

Attributes for Internal Use

An attribute name that starts with one underscore is not meant to be referenced externally.

```python
class FibIter:
    """An iterator over Fibonacci numbers."""
    def __init__(self):
        self._next = 0
        self._addend = 1
    def __next__(self):
        result = self._next
        self._next, self._addend = self._next, self._next + self._addend
        return result
```

Names in Local Scope

A name bound in a local frame is not accessible to other environments, except those that extend the frame

```python
def fib_generator():
    """A generator function for Fibonacci numbers."""
    yield 0
    previous, current = 0, 1
    while True:
        yield current
        previous, current = current, previous + current
```

Singleton Objects

A singleton class is a class that only ever has one instance

NoneType, the class of None, is a singleton class; None is its only instance

For user-defined singletons, some programmers re-bind the class name to the instance

```python
class empty_iterator:
    """An iterator over no values."""
    def __next__(self):
        raise StopIteration
empty_iterator = empty_iterator()
```

Streams
Streams are Lazy Linked Lists

A stream is a linked list, but the rest of the list is computed on demand

A stream is a linked list, but the rest of the list is computed on demand

Link (____________________, __________________)

First element can be anything

Second element is a Link instance or Link.empty

Stream (____________________, __________________)

First element can be anything

Second element is a zero-argument function that returns a Stream or Stream.empty

Once created, Streams and Links can be used interchangeably using first and rest methods

(Demo)

Integer Stream

An integer stream is a stream of consecutive integers

An integer stream starting at first is constructed from first and a function compute_rest that returns the integer stream starting at first+1

def integer_stream(first = 1):
    """Return a stream of consecutive integers, starting with first."""
    s = Stream(first, lambda: s)
    return s

def compute_rest():
    return integer_stream(first + 1)

return Stream(first, compute_rest)

(Demo)

Cross the Stream

Which definition will produce which row of elements after executing s = f()?

def f(x):
    return Stream([x], lambda: f([x]))

def f(x):
    return Stream(x, lambda: f(x + [1]))

def f(x):
    s = Stream([x], lambda: s)
    return s

def f(x):
    x.append(1)
    return Stream(x, lambda: f(x))

(Demo)

Stream Processing

Mapping a Function over a Stream

Mapping a function over a stream applies a function only to the first element right away; the rest is computed lazily

def map_stream(fn, s):
    """Map a function fn over the elements of a stream s."""
    if s is Stream.empty:
        return s
    def compute_rest():
        return map_stream(fn, s.rest)
    return Stream(fn(s.first), compute_rest)

This body is not executed until compute_rest is called

Not called yet

(Demo)

Stream Implementation

A stream is a linked list with an explicit first element and a rest-of-the-list that is computed lazily

class Stream:
    """A lazily computed linked list."""
    def __init__(self, first, compute_rest=lambda: Stream.empty):
        assert callable(compute_rest), 'compute_rest must be callable.'
        self.first = first
        self.compute_rest = compute_rest
    @property
    def rest(self):
        """Return the rest of the stream, computing it if necessary."""
        if self.compute_rest is not None:
            self._rest = self.compute_rest()
        self._compute_rest = None
        return self._rest

(Demo)
Filtering a Stream

When filtering a stream, processing continues until an element is kept in the output:

```python
def filter_stream(fn, s):
    """Filter stream s with predicate function fn.""
    if s is Stream.empty:
        return s
    def compute_rest():
        return filter_stream(fn, s.rest)
    if fn(s.first):
        return Stream(s.first, compute_rest)
    else:
        return compute_rest()
```

Actually compute the rest

A Stream of Primes

The stream of integers not divisible by any \( k \leq n \) is:

- The stream of integers not divisible by any \( k < n \)
- Filtered to remove any element divisible by \( n \)

This recurrence is called the Sieve of Eratosthenes:

\[
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, \ldots
\]

(Demo)