Advanced iteration.

1. Announcements.
   (a) Very slightly revised lab handout posted.
   (b) Pre-registration discussion this afternoon, Wege, 2:35 pm.
   (c) Early seating of final exam:
       Tuesday, December 14, 9:30am-noon.
   (d) Questions?

2. We’re interested in developing strategies for writing iterators that are versatile enough to simulate Python’s generators. This lecture explores some ideas.

3. AbstractIterators.
   (a) Recall: Java has two important interfaces: Iterator and Iterable.
   (b) An Iterator is any object that provides hasNext and next methods for generating a stream of values.
   (c) An Iterable is any object that provides a iterator() method. The focus of iterated for loops is an Iterable.
   (d) If you’re interested in developing a standalone value that produces a stream of values for an iterated for loop, you must develop object(s) that support both of these interfaces. This is a subtle observation.
   (e) The AbstractIterator class seeks to implement both of these interfaces:
      i. Iterator: The reset, hasNext, get, and next methods are abstract. You must provide a definition for each of these.
      ii. Iterable: The iterator() method returns this. It is declared final: you cannot change this behavior.
   (f) Extending the AbstractIterator class will allow you to design objects that can be the subjects of a for loop.

4. Implementing the Biterator: an iterator that returns count binary digits (Integers) of of a value. Focus: how to implement reset.

5. Implementing PrimeFactors: an iterator that returns the primes Factors of a value n. Focus: making progress at the appropriate time.

6. Implementing something like Python’s range object. Focus: building appropriate static factory methods.

7. Implementing Some: an iterator that returns some of the values of a subordinate iterable. Focus: making sure you know when you’re done.

8. Implementing Orbit: an object that maintains an externally specified state and its transition function, a Successor lambda.

Notes: