Lecture 19: Review of iterators, classes, and object-oriented programming

Iterators

Recall that something is *iterable* if it supports the `iter` function—that is the method `__iter__` is defined—and returns an iterator. An *iterator* is something that

- supports the `next` function—that is, the method `__next__` is defined;
- throws a `StopIteration` when the iterator is empty; and
- returns itself under an `iter` call.

Iterators may be defined using *classes* or with *generators*.

An Iterator for Squares

```
1   class squares:
2
3       def __init__(self, threshold=None):
4           self._state = 1
5           self._threshold = threshold
6
7       def _below_threshold(self):
8           return self._threshold is None or self._state**2 < self._threshold
9
10      def __iter__(self):
11          return self
12
13      def __next__(self):
14          if self._below_threshold():
15              sq = self._state**2
16              self._state += 1
17              return sq
18          else:
19              raise StopIteration()
```

A Generator for Squares

```
1  def squares_gen(threshold=None):
2      i = 1
3      while threshold is None or i**2 < threshold:
4          yield i**2
5          i += 1
```

Without getting too technical, the primary characteristics associated with object-oriented programming are

- inheritance;
- encapsulation; and
- polymorphism

Inheritance

```
class Shape:

class Rectangle(Shape):

class Square(Rectangle):
```

Encapsulation

```
class Shape:

class Rectangle(Shape):

  def __init__(self, width, height):
    self._width = width
    self._height = height
```

Polymorphism

```
class Shape:
  def area():
    pass
class Rectangle(Shape):

  def area():
    return self._width * self._height

class Square(Rectangle)

  def __init__(self, side):
    super().__init__(side, side)
```

```
>>> shape = Rectangle(10,20)
>>> shape.area()
200
>>> shape = Square(10)
>>> shape.area()
100
```

An Iterator for Even Squares

```
1  class even_squares(squares)
2
3      def __next__(self):
4  sq = next(super())
5  while (sq % 2 != 0):
6    sq = next(super())
7  return sq
```