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._addend, self._next = self._next, self._next + self._addend
        return result

>>> fibs = FibIter()
>>> [next(fibs) for _ in range(10)]
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

This naming convention is not enforced, but is typically respected.

A programmer who designs and maintains a public module may change internal-use names.

Starting a name with two underscores enforces restricted access from outside the class.

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

>>> fib = fib_generator()
>>> [next(fib) for _ in range(10)]
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

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()
```

The instance The class
Streams are Lazy Recursive Lists

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

Streams

Rlist( __________________ , __________________ )

First element

Second element

is an Rlist or

Rlist.empty

Stream( __________________ , __________________ )

First element

Second element

is anything

a zero-argument

function that

returns a Stream

or Stream.empty

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

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 = integer_stream()
    s.first = first
    s.rest = compute_rest()
    return s

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

Stream Processing

Demo

Stream Implementation

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

class Stream:
    """A lazily computed recursive 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):
        if self._compute_rest is None:
            self._compute_rest = self._compute_rest()  # compute rest
        return self._rest

Demo

Stream Implementation
Higher-Order Functions on Streams

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.

```python
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)
```

```python
>>> s = integer_stream(3)
>>> s
Stream(3, <...>)
>>> m = map_stream(lambda x: x * x, s)
>>> first_k(m, 5)
[9, 16, 25, 36, 49]
```

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()
```

```python
>>> s = integer_stream()
>>> s
Stream[...]
>>> m = map_stream(lambda x: x % 10, s)
>>> first_k(m, 7)
[0, 0, 5, 6, 0, 2, 5, 0, 6, 1, 0]
```

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.

```text
2, 3, 5, 7, 11, 13
```