Data Structures

Iterators

DataLab

November 12, 2016

Outline

Outline

Iterator Method

Scenario

We often want to access every item in a data structure or collection in turn...

Example with the array representation

1. listSize = len(List)
2. for i in range(n)
3. print List[i]

Iterator Method

Scenario

We often want to access every item in a data structure or collection in turn...

Example with the array representation

1. listSize = len(List)
2. for i in range(n)
3. print List[i]

Motivation

Question

What if we have a **get** that is really complex?

For example the get in Chain List Representation

We will see that it has complexity $O(n)$!!!

Previous Code

What a waste of time!!! $O\left(n^2\right)$**!!!!**

Motivation

Question

What if we have a **get** that is really complex?

For example the get in Chain List Representation

We will see that it has complexity $O(n)$!!!

Previous Code

What a waste of time!!! $O\left(n^2\right)$!!!!

Motivation

Question

What if we have a **get** that is really complex?

For example the get in Chain List Representation

We will see that it has complexity $O(n)$!!!

Previous Code

What a waste of time!!! $O\left(n^2\right)$**!!!!**

Then

Something Notable

Iteration is such a common operation that we could include it as part of the ADT list.

However

We do not want to add another operation to the ADT each time we think of another way to use an iteration.

Then

Something Notable

Iteration is such a common operation that we could include it as part of the ADT list.

However

We do not want to add another operation to the ADT each time we think of another way to use an iteration.

Outline

In Java at java.util we have the interface Iterator

Interface

```
package java.util;
public interface Iterator<Item> {

public boolean hasNext();

public Item next();

// Optional method
public void remove();

} // end Iterator
```

Explanation

public boolean hasNext()

- It detects whether this iterator has completed its traversal and gone beyond the last entry in the collection of data.
- It returns true if the iterator has another entry to return.

Explanation

public boolean hasNext()

- It detects whether this iterator has completed its traversal and gone beyond the last entry in the collection of data.
- It returns true if the iterator has another entry to return.

public T next()

- It retrieves the next entry in the collection and advances this iterator by one position.
- It returns a reference to the next entry in the iteration, if one exists
- It throws NoSuchElementException if the iterator had reached the end already, that is, if hasNext() is false.

Explanation

public boolean hasNext()

- It detects whether this iterator has completed its traversal and gone beyond the last entry in the collection of data.
- It returns true if the iterator has another entry to return.

public T next()

- It retrieves the next entry in the collection and advances this iterator by one position.
- It returns a reference to the next entry in the iteration, if one exists
- It throws NoSuchElementException if the iterator had reached the end already, that is, if hasNext() is false.

Explanation

public boolean hasNext()

- It detects whether this iterator has completed its traversal and gone beyond the last entry in the collection of data.
- It returns true if the iterator has another entry to return.

public T next()

- It retrieves the next entry in the collection and advances this iterator by one position.
- It returns a reference to the next entry in the iteration, if one exists
- It throws NoSuchElementException if the iterator had reached the end already, that is, if hasNext() is false.

Explanation

public boolean hasNext()

- It detects whether this iterator has completed its traversal and gone beyond the last entry in the collection of data.
- It returns true if the iterator has another entry to return.

public T next()

- It retrieves the next entry in the collection and advances this iterator by one position.
- It returns a reference to the next entry in the iteration, if one exists
- It throws NoSuchElementException if the iterator had reached the end already, that is, if hasNext() is false.

Explanation

public void remove()

- It removes from the collection of data the last entry that next() returned.

Precondition: next() has been called, and remove() has not been called since then.

- It throws IllegalStateException if next() has not been called, or if remove() was called already after the last call to next().

Explanation

public void remove()

- It removes from the collection of data the last entry that next() returned.

  Precondition: next() has been called, and remove() has not been called since then.

  - It throws IllegalStateException if next() has not been called, or if remove() was called already after the last call to next().

Explanation

public void remove()

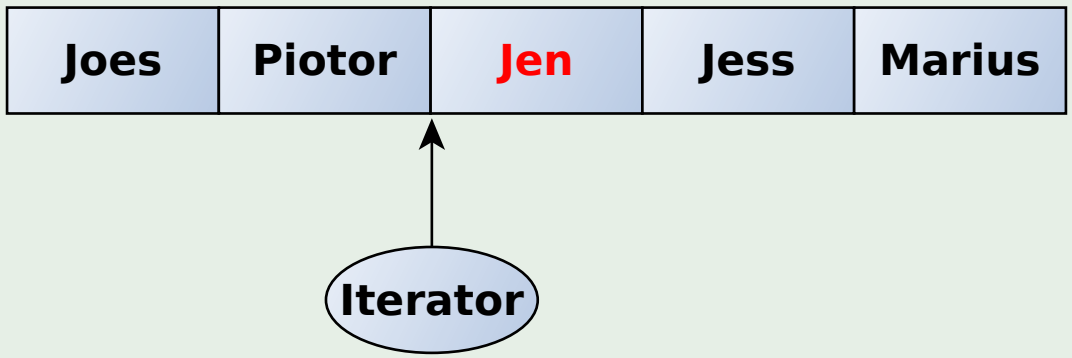
- It removes from the collection of data the last entry that next() returned.

  Precondition: next() has been called, and remove() has not been called since then.

  - It throws IllegalStateException if next() has not been called, or if remove() was called already after the last call to next().

Effect in a list

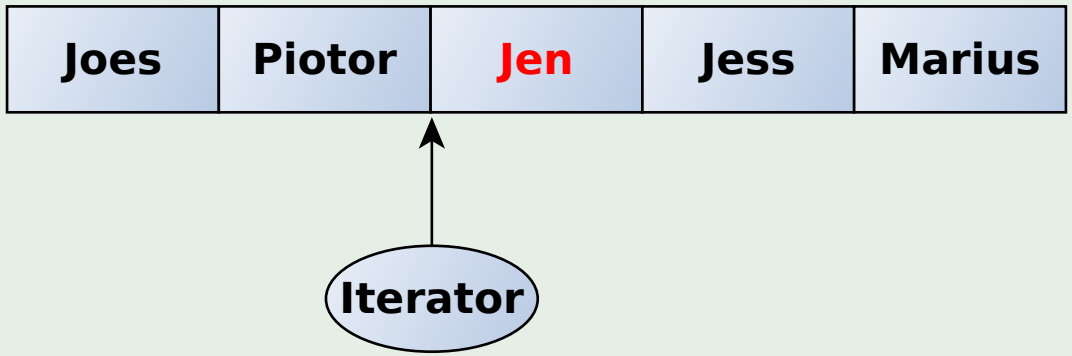
The List Before Any Operation with the Iterator



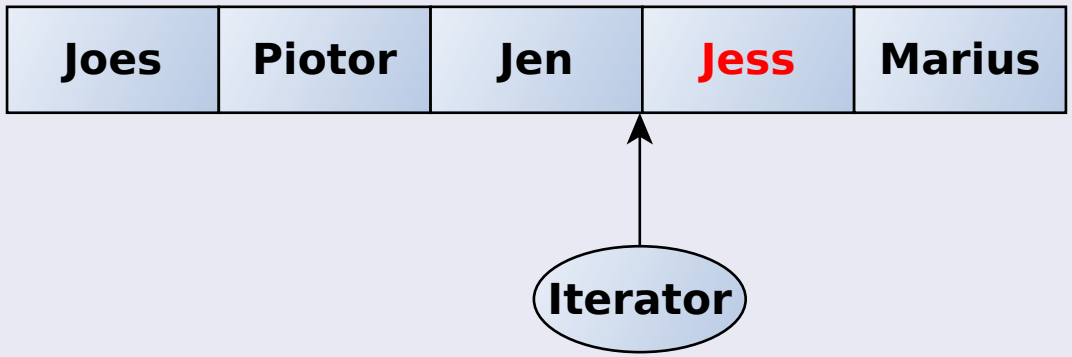
After the next() operation

Effect in a list

The List Before Any Operation with the Iterator

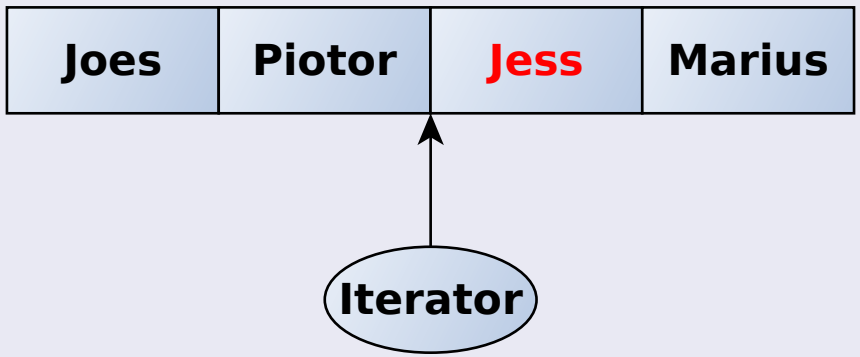


After the next() operation



Effect in a list

Now, we issue a remove()



Outline

We have the Iter Protocol

We have the following

- Operation “iter”

We have the following

We have

```
class yrange:
    def __init__(self, n):
        self.i = 0
        self.n = n

    def __iter__(self):
        return self

    def next(self):
        if self.i < self.n:
            i = self.i
            self.i += 1
            return i
        else:
            raise StopIteration()
```

Explanation

We have

- The ___iter___ method is what makes an object iterable.
- Behind the scenes, the iter function calls ___iter___ method on the given object.

Outline

Problems

Problem 1

Write an iterator class reverse_iter, that takes a list and iterates it from the reverse direction.

Problems 2

Write an iterator class, that takes a list and iterates over the even indexes

Problems

Problem 1

Write an iterator class reverse_iter, that takes a list and iterates it from the reverse direction.

Problems 2

Write an iterator class, that takes a list and iterates over the even indexes