PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Iterators in Python

Prof. Sindhu R Pai
PCPS Theory Anchor - 2024
Department of Computer Science and Engineering

Introduction

An iterator is an object that allows iteration through a sequence of elements, one at a time. It implements two main methods: **__iter__()** and **__next__()**.

__iter__():: returns the iterator object itself and is called when the iterator is initialized.

__next__():: returns the next item in the sequence.

When there are no more elements to return, it raises the **StopIteration** exception.

Iterators lazy object or Eager object?

Lazy Evaluation: Iterators follow the principle of lazy evaluation, meaning they generate values on-demand rather than computing all values at once.

This can be memory-efficient when dealing with large datasets as it only retrieves elements as needed.

Custom Iterable Objects

The container class (like list) should support a function

1. __iter__(callable as iter(container-object)) which returns an object of a class called an iterator.

2. __next__(callable as next(iterator_object))

These two functions are interfaces which can be implemented by traversing through a container

Ex. we may visit only elements in odd position or elements satisfying a boolean condition – like elements greater than 100

Examples

Example 1:Creates an object of MyContainer whose attribute mylist refers to the list a.

```
class MyContainer:
        def __init__(self, mylist):
                self.mylist = mylist
        def __iter__(self):
                self.i=0
                return self
        def __next__ (self):
                self.i += 1
                if self.i <= len(self.mylist):
                        return self.mylist[self.i - 1]
                else:
                                raise StopIteration
```

Examples

```
a = ['apple', 'banana', 'orange', 'dates',  'cherry’]

c = MyContainer(a)

for w in c :
        print(w)

c = MyContainer(a)
```

Here, observe that it creates an object of MyContainer whose attribute mylist refers to the list a

Working of iterators

The for statement calls iter(c) which is changed to MyContainer.__iter__(c)

This __iter__ function adds a position attribute i to the object and then returns the MyContainer object itself as the iterator object.

The for statement keeps calling next on this iterable object.

The __next__ function has the logic to return the next element from the list and update the position and also raise the exception stop iteration when the end of the list is reached.

Examples

Example 2

```python
class SquareNum:
    def __init__(self, n):
        self.n = n
        self.current = 0

    def __iter__(self):
        return self
    def __next__(self):
        if self.current >= self.n:
            raise StopIteration
        square = self.current ** 2
        self.current += 1
        return square
```

```python
squares = SquareNum(5)
# Using the iterable class

for num in squares:
    print(num)
```

SquareNum is a class that generates a sequence of squares of numbers from 0 to n-1
It has __iter__() and __next__() methods implemented, making it iterable.
When instance of this class in a for loop, it iterates through the sequence, printing the squares of the numbers from 0 to 4

THANK YOU

Department of Computer Science and Engineering

Dr. Shylaja S S, Director, CDSAML & CCBD, PESU
Prof. Sindhu R Pai – sindhurpai@pes.edu
Prof. C N Rajeswari