Iterators

Presented by: Brady Auen, Paul Kirilchuk, Kevin Vo

PEP 234 - Iterators

Why Propose

- This proposal proposes an iteration interface that objects can offer to control the behavior of ‘for’ loops.

Reasons for Propose:

- 1. Gives an extensible iterator interface.

Reasons for Propose:

- 1. Gives an extensible iterator interface.
- 2. Performance improvements to list iteration.

Reasons for Propose:

- 1. Gives an extensible iterator interface.
- 2. Performance improvements to list iteration.
- 3. Massive performance improvements to dictionary iteration.

Reasons for Propose:

- 1. Gives an extensible iterator interface.
- 2. Performance improvements to list iteration.
- 3. Massive performance improvements to dictionary iteration.
- 4. Provides an interface for iterating without pretending to offer random access to elements.

Reasons for Propose:

- 1. Gives an extensible iterator interface.
- 2. Performance improvements to list iteration.
- 3. Massive performance improvements to dictionary iteration.
- 4. Provides an interface for iterating without pretending to offer random access to elements.
- 5. Backward compatible with all existing user-defined classes and extension objects

Reasons for Propose:

- 1. Gives an extensible iterator interface.
- 2. Performance improvements to list iteration.
- 3. Massive performance improvements to dictionary iteration.
- 4. Provides an interface for iterating without pretending to offer random access to elements.
- 5. Backward compatible with all existing user-defined classes and extension objects
- 6.Code iterating over non-sequence collections more brief and clear.

How it Works:

- The iterator gives a ‘get next value’ operation that makes the next item in the sequence each time it is called.
- The operation gives an exception when no more things are accessible.
- There is only one required method, **next()**, which takes no arguments and returns the next value.
- When no values are left to be returned, calling next() should give the **StopIteration** exception.

How it Worked (Pseudo Iterator) Before Iterators(1):

- Before this update there wasn’t a clear way for the user to iterate through the contents of objects in Python.
- If the user wanted to create a “for item in object” sort of function, the method would look sort of like this:

How it Worked (Pseudo Iterator) Before Iterators(2):

```
def __getitem__(self, index):
return <next item>
```

How it Works Now (With Iterator):

-We have two built in functions that we can use to get iterators, the first is:

iter(obj)

and the second is:

iter(C, sentinel)

What Does the Iterator Feature Add?:

- A new exception is defined, **StopIteration**, which signals the end of an iteration.
- A new slot called **tp_iter**, that adds an iterator to the type object structure.
- Another new slot is added to the type structure called **tp_iternext**. It's for getting the next value in the iteration.

Iterating in Dictionaries:

-CODE EXAMPLE:

```
>>> m = {'Jan': 1, 'Feb': 2, 'Mar': 3, 'Apr': 4, 'May': 5, 'Jun': 6, 'Jul': 7, 'Aug': 8,
 'Sep': 9, 'Oct': 10, 'Nov': 11, 'Dec': 12}

>>> for key in m: print key, m[key]
Mar 3 Feb 2 Aug 8 Sep 9 May 5 Jun 6 Jul 7 Jan 1 Apr 4 Nov 11 Dec
```

Iterating in Files(1):

- Files also provide an iterator, which calls the **readline( )** method until there are no more lines in the file.
- This offers us with a good answer to the problem of iterate over the lines in a slow and nasty fashion.
- Ultimately, using an iterator is faster and more clear.

Iterating in Files(2):

#Iterating in Files Example:

Files implement a tp_iter slot that is equivalent to:

iter(f.readline, "").

Iterating in Files(3):

#Iterating in Files Example:

Files implement a tp_iter slot that is equivalent to:

iter(f.readline, "").

Iterating in Files(4):

-This means that we can write for line in file, which is equivalent to but faster than:

```
while 1:
    line = file.readline()
    if not line:
    break
```

How This Applies to Concepts Learned in CSCI 3155

- syntax that makes certain common tasks easier or less error prone in the language.

- perhaps describe the syntax in the context of allowed grammar productions.

Why the Community Passed this Proposal:

- A clearer, faster and more user friendly method to traverse a list, dictionary or file.