structs Documentation

Release 0.0.1

Jonathan Nappi

January 29, 2015

Contents

Release v0.0.1.

Python Data Structures for Humans

Generic, pure Python implementations of some more common types of data structures.

User Guide:

CHAPTER 1

structs package

1.1 structs.arrays module

structs.arrays.**prev**(*iterable*)
Iterate in reverse over an iterable type supporting the __prev__ API

class structs.arrays.**BaseList**(**args*, ***kwargs*)
Bases: list

Custom list subclass with some additional iteration functionality

next()
Handle next iteration functionality

prev()
Add basic implementation of the prev API

class structs.arrays.**BitArray**(*iterable=()*)
Bases: list

A bit array (also known as bitmap, bitset, bit string, or bit vector) is an array data structure that compactly stores bits. It can be used to implement a simple set data structure. A bit array is effective at exploiting bit-level parallelism in hardware to perform operations quickly.

append(*p_object*)
Append the logical bitwise representation of *p_object*

extend(*iterable*)
Append the logical bitwise representation of the objects in *iterable*

insert(*index*, *p_object*)
Append the logical bitwise representation of *p_object* to *index*

class structs.arrays.**SortedList**(*iterable=()*, *key=None*, *reverse=False*)
Bases: list

A list implementation that always maintains a sorted order

append(*p_object*)
Add *p_object* into ordered place in the list

extend(*iterable*)
Append each item in *iterable* into it's sorted location in the list

insert(*p_object*, **args*)
Insert *p_object* at it's calculated index

class structs.arrays.**CircularArray**(**args*, ***kwargs*)
Bases: structs.arrays.BaseList

A list subclass that will continually iterate until explicitly broken out of. ie, you're probably going to want a return or break in a loop over a CircularArray

1.2 structs.maps module

class structs.maps.**Dict**
Bases: dict

Overriden dict type with iadd functionality which will allow you to append two dictionaries together. ie:

```
>>> d = Dict(a=1, b=2)
>>> d += {'c': 3, 'd': 4}
>>> d
```

... {'a': 1, 'b': 2, 'c': 3, 'd': 4}

class structs.maps.**BiDirectionalMap**(*iterable=None*, ***kwargs*)
Bases: structs.maps.Dict

a bidirectional map, or hash bag, is an associative data structure in which the (key, value) pairs form a one-to-one correspondence. Thus the binary relation is functional in each direction: value can also act as a key to key. A pair (a, b) thus provides a unique coupling between a and b so that b can be found when a is used as a key and a can be found when b is used as a key.

get(*k*, *d=None*)
Return self[k] if k is in this bidirectionaldict, otherwise return *d*

Parameters

- **k** – A key to return from this bidirectionaldict
- **d** – The default value to return if *k* is not in this bidirectionaldict

Returns The value mapped to by *k* or *d* if *k* is not in this bidirectionaldict

items()
Return a 2-tuple of the (key, value) pairs in this BiDirectionalDict

keys()
Return a generator of the keys in this BiDirectionalDict

values()
Return a generator of the values in this BiDirectionalDict

class structs.maps.**MultiMap**
Bases: structs.maps.Dict

A MultiMap is a generalization of a dict type in which more than one value may be associated with and returned for a given key

setdefault(*k*, *d=None*)
If *k* is not contained in this MultiMap then store the value *d* in it.

Parameters

- **k** – The key to set the value for
- **d** – The default value to assign to key *k*

Returns The value stored at key *k*

update(*other=None*, ***kwargs*)
Update this MultiMap with either the

Parameters

- **other** – Another dict to merge into this MultiMap or an iterable of (key, value) 2-tuples
- **kwargs** – Arbitrary keyword args to merge into this MultiMap

CHAPTER 2

Indices and tables

- *genindex*
- *modindex*
- *search*

S