

# Data Structures

Generated by Doxygen 1.8.9.1

Wed Aug 12 2015 19:31:21



# Contents

<b>1</b>	<b>Data Structures</b>	<b>1</b>
<b>2</b>	<b>Namespace Index</b>	<b>3</b>
2.1	Namespace List . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>Namespace Documentation</b>	<b>7</b>
4.1	BST Namespace Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.2	LinkedList Namespace Reference . . . . .	7
4.2.1	Detailed Description . . . . .	7
4.3	Queue Namespace Reference . . . . .	7
4.3.1	Detailed Description . . . . .	7
4.4	Stack Namespace Reference . . . . .	8
4.4.1	Detailed Description . . . . .	8
<b>5</b>	<b>Class Documentation</b>	<b>9</b>
5.1	LinkedList.LinkedList Class Reference . . . . .	9
5.1.1	Detailed Description . . . . .	9
5.1.2	Constructor & Destructor Documentation . . . . .	10
5.1.2.1	__init__ . . . . .	10
5.1.3	Member Function Documentation . . . . .	10
5.1.3.1	__getitem__ . . . . .	10
5.1.3.2	__len__ . . . . .	10
5.1.3.3	__repr__ . . . . .	10
5.1.3.4	append . . . . .	11
5.1.3.5	copy . . . . .	11
5.1.3.6	extend . . . . .	11
5.1.3.7	get . . . . .	11
5.1.3.8	insert . . . . .	11
5.1.3.9	pop . . . . .	11

5.1.3.10	search	11
5.2	LinkedList.Node Class Reference	11
5.2.1	Detailed Description	12
5.2.2	Constructor & Destructor Documentation	12
5.2.2.1	__init__	12
5.2.3	Member Function Documentation	12
5.2.3.1	__repr__	12
5.2.3.2	getnxt	12
5.2.3.3	getprev	12
5.2.3.4	setnxt	12
5.2.3.5	setprev	12
5.3	Queue.queue Class Reference	12
5.3.1	Detailed Description	13
5.3.2	Constructor & Destructor Documentation	13
5.3.2.1	__init__	13
5.3.3	Member Function Documentation	13
5.3.3.1	__getitem__	13
5.3.3.2	__repr__	14
5.3.3.3	copy	14
5.3.3.4	dequeue	14
5.3.3.5	enqueue	14
5.3.3.6	extend	14
5.3.3.7	peek	14
5.4	Stack.stack Class Reference	14
5.4.1	Detailed Description	15
5.4.2	Constructor & Destructor Documentation	15
5.4.2.1	__init__	15
5.4.3	Member Function Documentation	15
5.4.3.1	copy	15
5.4.3.2	extend	15
5.4.3.3	pop	15
5.4.3.4	push	15
	<b>Index</b>	<b>17</b>

# Chapter 1

## Data Structures

A set of useful data structures. This will keep expanding

### Stack

Implements a basic stack using a Python list. Basic stack functionality is easily implemented by Python already, this just binds it all together. Uses the doctest module for testing

### Queue

Implements a basic queue using a Python list. Same as [Stack](#), a lot of this is already implemented by Python and this just organizes it

### LinkedList

Implements a linked list and the basic functions (append, extend, insert, pop, search, get)



## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

<a href="#">BST</a>	7
<a href="#">LinkedList</a>	7
<a href="#">Queue</a>	7
<a href="#">Stack</a>	8





## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">LinkedList.LinkedList</a>	9
<a href="#">LinkedList.Node</a>	11
<a href="#">Queue.queue</a>	12
<a href="#">Stack.stack</a>	14



## Chapter 4

# Namespace Documentation

### 4.1 BST Namespace Reference

#### 4.1.1 Detailed Description

Implementation of a BST Data Structure  
Author: Yannis Mentekidis  
Date: August 12, 2015

### 4.2 LinkedList Namespace Reference

#### Classes

- class [LinkedList](#)
- class [Node](#)

#### 4.2.1 Detailed Description

Linked List implementation in Python  
Author: Yannis Mentekidis  
Date: August 12, 2015

### 4.3 Queue Namespace Reference

#### Classes

- class [queue](#)

#### Functions

- def **main** ()

#### 4.3.1 Detailed Description

Implements a basic Queue using a Python list  
Author: Yannis Mentekidis  
Date: August 12, 2015

## 4.4 Stack Namespace Reference

### Classes

- class [stack](#)

### Functions

- def **main** ()

#### 4.4.1 Detailed Description

Implements a Stack Data Structure using a Python list  
Author: Yannis Mentekidis  
Date: August 12, 2015

## Chapter 5

# Class Documentation

### 5.1 LinkedList.LinkedList Class Reference

#### Public Member Functions

- def `__init__`
- def `__len__` (self)
- def `__repr__` (self)
- def `__getitem__` (self, index)
- def `copy` (self)
- def `append` (self, item)
- def `extend` (self, alist)
- def `insert`
- def `search` (self, key)
- def `get` (self, idx)
- def `pop`

#### Public Attributes

- `length`
- `start`

#### 5.1.1 Detailed Description

Linked List data structure

Implements the basic functions of a linked list.

Testing:

```
>>> L = LinkedList(['a', 'b', 'c', 'd']) #test constructor and overloads
>>> print L
[0:a], [1:b], [2:c], [3:d]
>>> len(L)
4
>>> L[0]
'a'
>>> L[-1]
Traceback (most recent call last):
...
IndexError

>>> L.append('e') #test append, extend
>>> L
[0:a], [1:b], [2:c], [3:d], [4:e]
```

```

>>> L.extend(['f', 'g', 'h'])
>>> L[5]
'f'
>>> L[6]
'g'
>>> L[7]
'h'

>>> L.pop(0) #test pop
[0:a]
>>> print [L.pop(0) for i in range(5)]
[[0:b], [0:c], [0:d], [0:e], [0:f]]
>>> L.pop(len(L)-1)
[1:h]
>>> L.pop(1)
Traceback (most recent call last):
...
IndexError

>>> L = LinkedList() #test empty constructor and insert, search
>>> L.insert('b')
>>> L.insert('a', 0)
>>> L.insert('c')
>>> L
[0:a], [1:b], [2:c]

>>> L.search('b')
1
>>> L.search('f')
-1

>>> L2 = L.copy() #test copy functionality
>>> junk = L2.pop()
>>> L.insert('d')
>>> L2
[0:b], [1:c]
>>> L
[0:a], [1:b], [2:c], [3:d]

```

## 5.1.2 Constructor & Destructor Documentation

### 5.1.2.1 `def LinkedList.LinkedList.__init__( self, alist = list() )`

Constructor: Insert items from alist into the linked list

## 5.1.3 Member Function Documentation

### 5.1.3.1 `def LinkedList.LinkedList.__getitem__( self, index )`

Overload for [] operation

### 5.1.3.2 `def LinkedList.LinkedList.__len__( self )`

Overload for the len() operator

### 5.1.3.3 `def LinkedList.LinkedList.__repr__( self )`

Prints the LinkedList, separated by commas. Format: [idx:value]

#### 5.1.3.4 def LinkedList.LinkedList.append ( self, item )

Insert a single item at the end of the LinkedList

#### 5.1.3.5 def LinkedList.LinkedList.copy ( self )

Returns a copy of the Linked List (not a pointer)

#### 5.1.3.6 def LinkedList.LinkedList.extend ( self, alist )

Add all items in alist into the LinkedList

#### 5.1.3.7 def LinkedList.LinkedList.get ( self, idx )

Return the value of item number idx

#### 5.1.3.8 def LinkedList.LinkedList.insert ( self, item, position = None )

Add an item to a given point in the LinkedList

#### 5.1.3.9 def LinkedList.LinkedList.pop ( self, idx = 0 )

Remove item idx from the LinkedList

#### 5.1.3.10 def LinkedList.LinkedList.search ( self, key )

Sequential search for a key in LinkedList

The documentation for this class was generated from the following file:

- LinkedList.py

## 5.2 LinkedList.Node Class Reference

### Public Member Functions

- def [\\_\\_init\\_\\_](#)
- def [setnxt](#) (self, nxt)
- def [getnxt](#) (self)
- def [setprev](#) (self, prev)
- def [getprev](#) (self)
- def [\\_\\_repr\\_\\_](#) (self)

### Public Attributes

- **idx**
- **value**
- **prev**
- **nxt**

### 5.2.1 Detailed Description

Node of a LinkedList

Contains information about the index, location, and content of a LinkedList Node:

```
idx - ascending number of the node
value - the 'content' of the node
prev - previous node
nxt - next node
```

### 5.2.2 Constructor & Destructor Documentation

#### 5.2.2.1 `def LinkedList.Node.__init__( self, idx, value, prev=None, nxt=None )`

The constructor.

### 5.2.3 Member Function Documentation

#### 5.2.3.1 `def LinkedList.Node.__repr__( self )`

print operator overload

#### 5.2.3.2 `def LinkedList.Node.getnxt( self )`

Getter for the nxt attribute

#### 5.2.3.3 `def LinkedList.Node.getprev( self )`

Getter for the prev attribute

#### 5.2.3.4 `def LinkedList.Node.setnxt( self, nxt )`

Setter for the nxt attribute

#### 5.2.3.5 `def LinkedList.Node.setprev( self, prev )`

Setter for the prev attribute

The documentation for this class was generated from the following file:

- `LinkedList.py`

## 5.3 Queue.queue Class Reference

### Public Member Functions

- `def \_\_init\_\_`
- `def enqueue(self, item)`
- `def dequeue(self)`



- def `extend` (self, alist)
- def `peek`
- def `copy` (self)
- def `__getitem__` (self, index)
- def `__repr__` (self)

## Public Attributes

- `bottom`
- `top`
- `content`

### 5.3.1 Detailed Description

Queue data structure

Testing:

```
>>> alist = [1, 2, 3, 4, 5] #test constructor
>>> Q = queue(alist)
>>> print Q #test __repr__
1 2 3 4 5

>>> Q.enqueue(6) #test enqueue function
>>> print Q
1 2 3 4 5 6

>>> alist = [7, 8, 9]
>>> Q.extend(alist) #test extend function
>>> print Q
1 2 3 4 5 6 7 8 9

>>> print [Q.dequeue() for i in range(10)] #test dequeue function
[1, 2, 3, 4, 5, 6, 7, 8, 9, None]

>>> Q = queue() #test empty constructor
>>> Q.enqueue(1)
>>> Q.enqueue(2)
>>> Q.enqueue(3)
>>> print Q
1 2 3
>>> print [Q[i] for i in range(10)] #test __getitem__
[1, 2, 3, None, None, None, None, None, None, None]

>>> Q = queue([1, 2, 3, 4]) #test copy (modifying S leaves Q unaffected)
>>> S = Q.copy()
>>> S.enqueue(5)
>>> S
1 2 3 4 5
>>> Q
1 2 3 4
```

### 5.3.2 Constructor & Destructor Documentation

#### 5.3.2.1 def Queue.queue.\_\_init\_\_( self, alist = list() )

Turns a list into a new Queue

### 5.3.3 Member Function Documentation

#### 5.3.3.1 def Queue.queue.\_\_getitem\_\_( self, index )

Overload for the `[]` operator

#### 5.3.3.2 `def Queue.queue.__repr__( self )`

Overload for print

#### 5.3.3.3 `def Queue.queue.copy( self )`

Returns a copy of the stack

#### 5.3.3.4 `def Queue.queue.dequeue( self )`

Removes and returns item 0, if it exists. Returns None if not

#### 5.3.3.5 `def Queue.queue.enqueue( self, item )`

Adds a new item to the end of the queue

#### 5.3.3.6 `def Queue.queue.extend( self, alist )`

Adds all items on a list to the end of the queue

#### 5.3.3.7 `def Queue.queue.peek( self, i = 0 )`

Returns item i from the queue without removing it.

The documentation for this class was generated from the following file:

- Queue.py

## 5.4 Stack.stack Class Reference

### Public Member Functions

- `def \_\_init\_\_`
- `def push(self, item)`
- `def extend(self, alist)`
- `def pop(self)`
- `def copy(self)`
- `def \_\_repr\_\_(self)`
- `def \_\_getitem\_\_(self, index)`

### Public Attributes

- **content**
- **bottom**
- **top**

### 5.4.1 Detailed Description

Stack data structure

Testing:

```
>>> alist = [1, 2, 3, 4, 5]
>>> S = stack(alist)
>>> print S
1 2 3 4 5

>>> S.push(6)
>>> print S
1 2 3 4 5 6

>>> alist = [7, 8, 9]
>>> S.extend(alist)
>>> print S
1 2 3 4 5 6 7 8 9

>>> print [S.pop() for i in range(10)]
[9, 8, 7, 6, 5, 4, 3, 2, 1, None]

>>> S = stack()
>>> S.push(-1)
>>> S.push(-2)
>>> S.push(-3)
>>> print [S[i] for i in range(4)]
[-1, -2, -3, None]

>>> S = stack([1, 2, 3, 4])
>>> T = S.copy()
>>> T.push(5)
>>> T
1 2 3 4 5
>>> S
1 2 3 4
```

### 5.4.2 Constructor & Destructor Documentation

#### 5.4.2.1 `def Stack.stack.__init__( self, alist = list() )`

Initialize a stack with a list (default empty)

### 5.4.3 Member Function Documentation

#### 5.4.3.1 `def Stack.stack.copy( self )`

Return a copy of the stack (not a pointer)

#### 5.4.3.2 `def Stack.stack.extend( self, alist )`

Add all list items to top of the stack

#### 5.4.3.3 `def Stack.stack.pop( self )`

Remove top item or None if stack is empty

#### 5.4.3.4 `def Stack.stack.push( self, item )`

Add item to the top of the stack

The documentation for this class was generated from the following file:

- `Stack.py`

# Index

- `__getitem__`
    - `LinkedList::LinkedList`, 10
    - `Queue::queue`, 13
  - `__init__`
    - `LinkedList::LinkedList`, 10
    - `LinkedList::Node`, 12
    - `Queue::queue`, 13
    - `Stack::stack`, 15
  - `__len__`
    - `LinkedList::LinkedList`, 10
  - `__repr__`
    - `LinkedList::LinkedList`, 10
    - `LinkedList::Node`, 12
    - `Queue::queue`, 13
- `append`
  - `LinkedList::LinkedList`, 10
- `BST`, 7
- `copy`
  - `LinkedList::LinkedList`, 11
  - `Queue::queue`, 14
  - `Stack::stack`, 15
- `dequeue`
  - `Queue::queue`, 14
- `enqueue`
  - `Queue::queue`, 14
- `extend`
  - `LinkedList::LinkedList`, 11
  - `Queue::queue`, 14
  - `Stack::stack`, 15
- `get`
  - `LinkedList::LinkedList`, 11
- `getnxt`
  - `LinkedList::Node`, 12
- `getprev`
  - `LinkedList::Node`, 12
- `insert`
  - `LinkedList::LinkedList`, 11
- `LinkedList`, 7
- `LinkedList.LinkedList`, 9
- `LinkedList.Node`, 11
- `LinkedList::LinkedList`
  - `__getitem__`, 10
  - `__init__`, 10
  - `__len__`, 10
  - `__repr__`, 10
  - `append`, 10
  - `copy`, 11
  - `extend`, 11
  - `get`, 11
  - `insert`, 11
  - `pop`, 11
  - `search`, 11
- `LinkedList::Node`
  - `__init__`, 12
  - `__repr__`, 12
  - `getnxt`, 12
  - `getprev`, 12
  - `setnxt`, 12
  - `setprev`, 12
- `peek`
  - `Queue::queue`, 14
- `pop`
  - `LinkedList::LinkedList`, 11
  - `Stack::stack`, 15
- `push`
  - `Stack::stack`, 15
- `Queue`, 7
- `Queue.queue`, 12
- `Queue::queue`
  - `__getitem__`, 13
  - `__init__`, 13
  - `__repr__`, 13
  - `copy`, 14
  - `dequeue`, 14
  - `enqueue`, 14
  - `extend`, 14
  - `peek`, 14
- `search`
  - `LinkedList::LinkedList`, 11
- `setnxt`
  - `LinkedList::Node`, 12
- `setprev`
  - `LinkedList::Node`, 12
- `Stack`, 8
- `Stack.stack`, 14
- `Stack::stack`
  - `__init__`, 15
  - `copy`, 15
  - `extend`, 15
  - `pop`, 15

push, [15](#)