



Doubly Linked Lists

Cheatsheets / Linear Data Structures

TOPICS Linked Lists Doubly Linked Lists Linked List Practice Searching Arrays Queues

Stacks

Adding to the Tail

A Python DoublyLinkedList class can implement an .add_to_tail() instance method for adding new data to the tail of the list. .add_to_tail() takes a single new_value argument. It uses new_value to create a new Node which it adds to the tail of the list.

def add_to_tail(self, new_value): new_tail = Node(new_value) current_tail = self.tail_node if current_tail != None: current_tail.set_next_node(new_tail) new_tail.set_prev_node(current_tail) self.tail_node = new_tail if self.head_node == None: self.head_node = new_tail

Adding to the Head

A Python DoublyLinkedList class can implement an _add_to_head() instance method for adding new data to the head of the list. .add_to_head() takes a single new_value argument. It uses new_value to create a new Node which it adds to the head of the list.

def add_to_head(self, new_value): new_head = Node(new_value) current_head = self.head_node if current_head != None: current_head.set_prev_node(new_head) new_head.set_next_node(current_head) self.head_node = new_head if self.tail_node == None: self.tail_node = new_head

Removing the Tail

A Python DoublyLinkedList class can implement a .remove_tail() instance method for removing the head of the list. .remove_tail() takes no arguments. It removes and returns the tail of the list, and sets the tail's previous node as the new tail.



Removing the Head

A Python DoublyLinkedList class can implement a .remove_head() instance method for removing the head of the list. .remove_head() takes no arguments. It removes and returns the head of the list, and sets the head's next node as the new head.



Removing by Value

A Python DoublyLinkedList class can implement a .remove_by_value() instance method that takes value_to_remove as an argument and returns the node that matches value_to_remove, or None if no match exists. If the node exists, .remove_by_value() removes it from the list and correctly resets the pointers of its surrounding nodes.



A Python DoublyLinkedList class constructor

Constructor

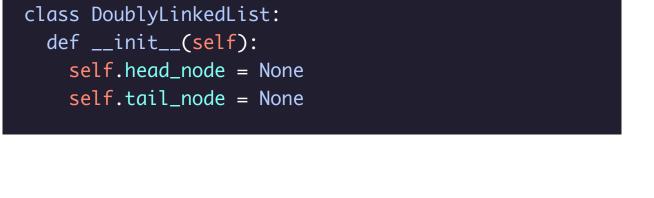
should store: A head_node property to store the head of

- the list • A tail_node property to store the tail of the
- list
- as their defaults.

The head_node and tail_node are set to None

Updated Node Class Doubly linked lists in Python utilize an updated

Node class that has a pointer to the previous node. This comes with additional setter and previous node.



getter methods for accessing and updating the



def __init__(self, value, next_node=None,

class Node:

prev_node=None):

A DoublyLinkedList class in Python has the following functionality:

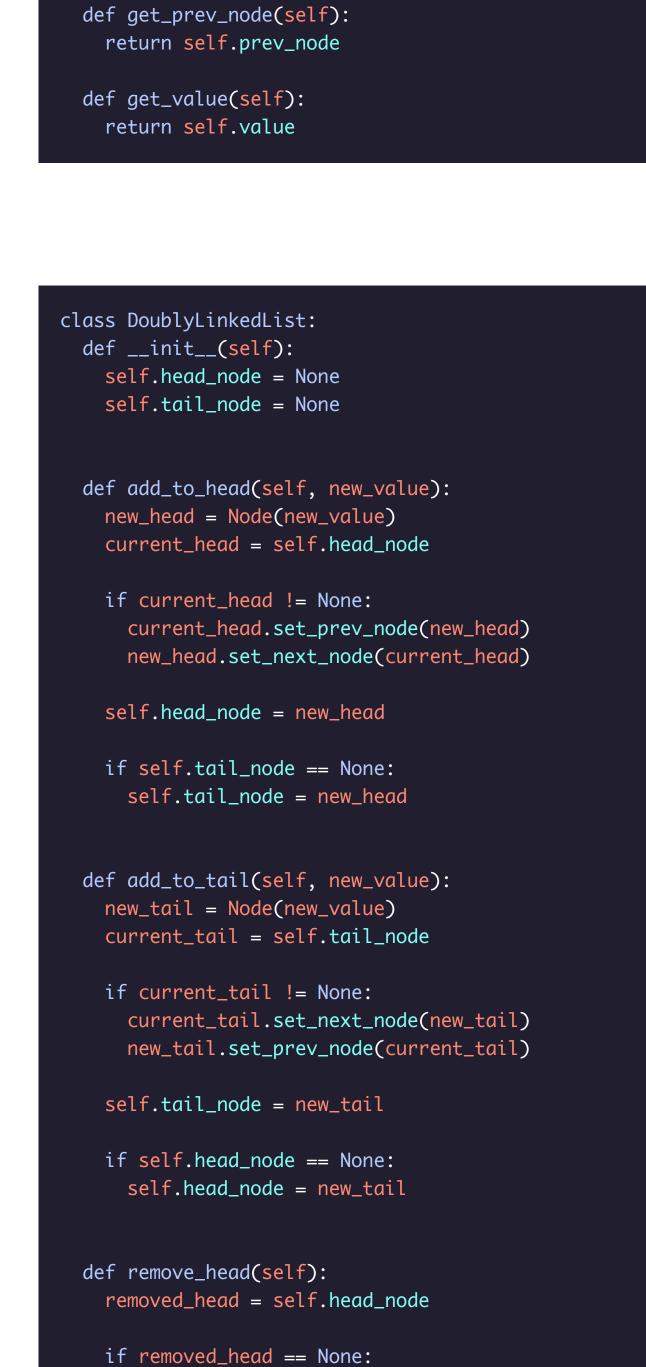
Doubly Linked List Overview

 A constructor with head_node and tail_node properties

- An .add_to_head() method to add new nodes to the head
- An .add_to_tail() method to add new nodes to the tail
- A .remove_head() method to remove the head node • A .remove_tail() method to remove the tail
- node • A .remove_by_value() method to remove a

node that matches the value_to_remove

passed in



return removed_head.get_value() def remove_tail(self): removed_tail = self.tail_node if removed_tail == None: return None self.tail_node = removed_tail.get_prev_node()

self.head_node = removed_head.get_next_node()

self.head_node.set_prev_node(None)

if removed_head == self.tail_node:

return None

if self.head_node != None:

self.remove_tail()

if self.tail_node != None: self.tail_node.set_next_node(None) if removed_tail == self.head_node: self.remove_head() return removed_tail.get_value()

def remove_by_value(self, value_to_remove):

node_to_remove = None current_node = self.head_node while current_node != None: if current_node.get_value() == value_to_remove: node_to_remove = current_node

break

current_node = current_node.get_next_node() if node_to_remove == None: return None if node_to_remove == self.head_node:

self.remove_head() elif node_to_remove == self.tail_node: self.remove_tail() else: next_node = node_to_remove.get_next_node() prev_node = node_to_remove.get_prev_node()

next_node.set_prev_node(prev_node)

prev_node.set_next_node(next_node)

return node_to_remove

Next →

← Previous

PRO Path Pass the Technical Interview with

Python

Download on the App Store

Google Play

Related Courses

Keep Going Enrolled...

code cademy **RESOURCES** COMMUNITY from skillsoft **Projects** Forums **About** Interview Challenges Discord Careers Docs Chapters Affiliates Cheatsheets **Events** Shop **Articles Learner Stories** y f C You **Videos** Blog Career Center **MOBILE ENTERPRISE PLANS** INDIVIDUAL PLANS

Pro Membership

For Students

SUPPORT

Help Center

Privacy Policy | Cookie Policy | Do Not Sell My Personal Information | Terms

Business Solutions

COURSE CATALOG Subjects Web Development Data Science **Computer Science Developer Tools** Machine Learning **Code Foundations** Web Design Full Catalog Beta Content

Roadmap

Languages HTML & CSS C++ Python JavaScript C# PHP Java SQL Go Bash/Shell Swift Kotlin Ruby