Ảnh có chứa ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện, văn bản

Mô tả được tạo tự động  
  
This Python code provides an implementation of a singly linked list data structure along with some basic operations. Let's go through the code step by step and explain each part:

Class **Node**:

This class represents a single node in the linked list. Each node has a **data** attribute that stores the value of the node, and a **next** attribute that points to the next node in the list.

Class **LinkedList**:

This class represents the linked list itself. It has an initial **head** attribute, which points to the first node of the linked list. If the list is empty, the **head** is set to **None**.

**append(data)** method:

This method adds a new node with the given **data** at the end of the linked list. It traverses the list until it reaches the last node and then appends the new node to the **next** attribute of the last node.

**prepend(data)** method:

This method adds a new node with the given **data** at the beginning of the linked list. It creates a new node and sets its **next** attribute to the current **head** of the list. Then, it updates the **head** to point to the new node, making it the new first node.

**delete(key)** method:

This method deletes the first occurrence of a node with the given **key** value from the linked list. It starts traversing the list from the **head** and looks for the node with the matching **key**. If found, it updates the **next** pointer of the previous node to skip the node to be deleted.

**search(key)** method:

This method searches for the first occurrence of a node with the given **key** in the linked list. It starts from the **head** and iterates through the list until it finds the node with the matching **key**. If found, it returns **True**; otherwise, it returns **False**.

**insert\_after\_node(prev\_node, data)** method:

This method inserts a new node with the given **data** after a specified node (**prev\_node**). It first checks if the **prev\_node** exists in the list. If it does, it creates a new node and updates the **next** pointers to insert the new node after **prev\_node**.

**print\_list()** method:

This method prints the elements of the linked list in order. It starts from the **head** and traverses the list, printing the **data** of each node until the end.

Example Usage:

The example usage section creates a linked list, appends and prepends nodes, and then performs a search operation to find a specific key in the list. Finally, it inserts a new node after a given node and prints the linked list.

The linked list data structure is a fundamental data structure used in computer science and plays a crucial role in various applications. It allows for efficient insertion and deletion of elements compared to arrays, and it can dynamically grow or shrink based on the number of elements it contains.