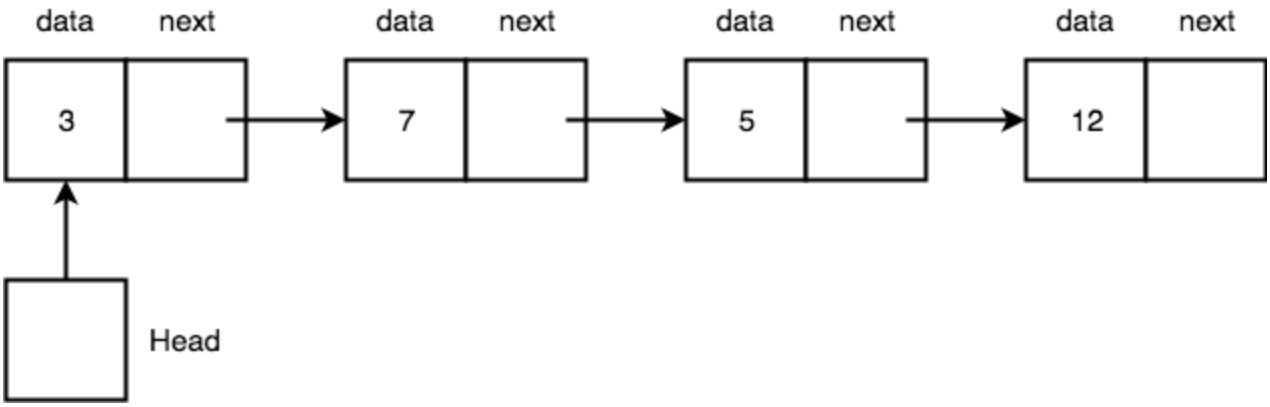
Linked List

What is a Linked List?

> A linked list is a sequence of data elements, which are connected together via **links**. Each data elements contains a connection to another data element in form of a pointer.



Some difference between traditional arrays and linked lists are,

Advanges / Disadvatages	Arrays	Linked List
Random Access	We can Randomly access array values with array indexes. For instance, `arr[35]` would give us the value stored in the 36^{th} index	We need to touch the previous index or object to goto

Advanges / Disadvatages	Arrays	Linked List
	and we don't need to access or touch any other index or values in the array.	the next object or index.

Linked List - Basic Operations

List Iteration

```
n = head
while n is not None:
    # Do Something
    n = n.next
```

Count function

Given a list, returns the number of nodes are present in the list.

```
def countNode(head):
    count = 0
    n = head
    while n is not None:
        count += 1
        n = n.next

    return count
```

Get function

Given head of the linked list, and an index, returns the value stored into the index of the list. If the index is invalid returns `-1`

```
def get(head, index):
    count = 0
    n = head
    while n is not None:
        if count == index:
            return n.element

        count += 1
        n = n.next
```

```
    return -1
```

Node At function

Takes Node Head, list size and index as input and returns the address of the node, that is stored in the index. Returns `None` if the index is invalid.

```
def nodeAt(head, size, index):
    if (index < 0 or index >= size):
        return None

    n = head
    for _ in range(0, index):
        n = n.next

    return n
```

Set function

Given an node index, give us the option to change the value stored in that index. (Assuming that the input index is right)

```
def set(head, index, element):
    count = 0
    n = head
    while n is not None:
        if count == index:
            n.element = element

        count += 1
        n = n.next
```

Search function

Given the linked list and an element, if the element can be found in the linked list, returns the position of that element. If the element is not in the list reutrns `-1` .

```
def indexOf(head, element):
    index = 0
    n = head
    while n is not None:
        if (n.element == element):
            reuturn index
```

```
            index += 1
            n = n.next

        return -1
```

Insert Elements in the list

At the beginning

At middle

At the End