## Recursion and Advanced Data Structures: Takeaways

by Dataquest Labs, Inc. - All rights reserved © 2019

## **Syntax**

• Using recursion to return the Fibonacci number:

```
def fib(n):
   if n == 0 or n == 1:
      return 1
return fib(n - 1) + fib(n - 2)
```

• Getting the length of the linked list using iteration:

```
def length_iterative(ls):
    count = 0
    while not ls.is_empty():
        count = count + 1
        ls = ls.tail()
    return count
```

• Using recursion to find the length of a linked list:

```
def length_recursive(ls):
   if ls.is_empty():
      return 0

return 1 + length_recursive(ls.tail())
```

## Concepts

- Recursion is the method of repeating code without using loops. An example of recursion would be the factorial function in mathematics.
  - We denote a factorial using the ! sign. For example n! denotes multiplying n by all the positive integers less than n . However, 0! is defined as 1.
    - For example: 5! = 5 \* 4 \* 3 \* 2 \* 1.

- A linked list is made up of many nodes. In a singly linked list, each node contains its data as well as the next node.
- A linked list is a type a recursive data structure since each node contains the data and then points to another linked list.
- An advantage of using linked lists is that you need to modify very few nodes when inserting or deleting because the update only requires a constant amount of changes.

## Resources

- Recursion
- Linked Lists



Takeaways by Dataquest Labs, Inc. - All rights reserved © 2019