Input node in a sorted linked List
5->7->10->
add 9
5->7->9->10->

Questions to ask:

- Is it already sorted?
- What if it is an empty list?
- If not specified: ask if in increasing or decreasing order.
- Any repeating and how to deal with it

Initial algorithm:

- If the list is empty, add the node and make it the head.
- Add element to the list, start from the head and check if the item is bigger than the current one.
    - If it is larger, go to next element, and repeat
    - If smaller
        - If it is at head, make it the head
        - If its anywhere else make connections

Things to consider:

- What is the complexity of this algorithm, O(n)
- What sort of linked list should this be?
    - Singly Linked List
    - Doubly Linked List
    - Circular Linked List
        - Should only need a Singly linked list because Since I dont need to go back, I dont need a double or circular linked list.
- Do I need just head or do I add tail
    - For this no need for tail because either way its going to be O(n)

Python Code:

```
#Node class
class Node:
	def __init__(self, data):
		self.data = data
		self.next = None

#Linked List Class
class linked_list:
	def __init__(self):
		self.head = None

	def add_node(self, node):
		#if list is empty
		if self.head == None:
			node.next = self.head
			self.head = node
		#if the head is the one to switch
		elif self.head >= node:
			node.next = self.head
			self.head = node
		else:
			current = self.head
			while current.next != None and current.next.data < node.data:
			# ^ This line WAS bad CHANGED IS HIGHLIGHTED
				current = current.next

			node.next = current.next
			current.next = node


	def print_list(self):
		current = self.head
		print("head->")
		while current.next != None:
			print(f"{current.data}->")
		print("None")

#Testing
def main():
   list = linked_list()
   new_node = Node(5)
   list.add_node(new_node)
```

```
    list.print_list()

    new_node = Node(10)
    list.add_node(new_node)
    list.print_list()

    new_node = Node(7)
    list.add_node(new_node)
    list.print_list()

    new_node = Node(3)
    list.add_node(new_node)
    list.print_list()

    new_node = Node(1)
    list.add_node(new_node)
    list.print_list()

    new_node = Node(9)
    list.add_node(new_node)
    list.print_list()

if __name__ == "__main__":
    main()
```