Data Strcture
LAB 5

PREPARED BY:

AYOOB ABDULMUNEM

Ahmed Eskander Mezher

Lecture Outline: -

1- insert a node at the beginning of the liked list

2- compute the length of the linked list

insert a node at the beginning of the liked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def insert_begin(self,data):
        nod=Node(data)
        if self.head==None:
            self.head=nod
            return
        nod.next=self.head
        self.head=nod

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
```

insert a node at the beginning of the liked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def insert_begin(self,data):
        nod=Node(data)
        if self.head==None:
            self.head=nod
            return
        nod.next=self.head
        self.head=nod

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
```

head = None

insert a node at the beginning of the liked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def insert_begin(self,data):
        nod=Node(data)
        if self.head==None:
            self.head=nod
            return
        nod.next=self.head
        self.head=nod

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
```

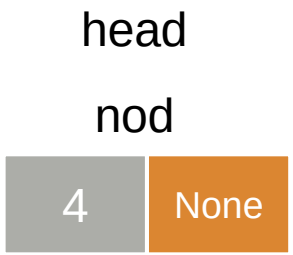
head = None

nod

insert a node at the beginning of the liked list

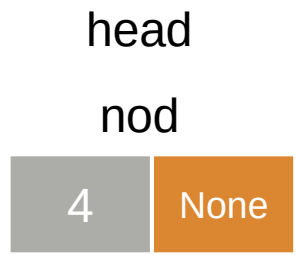
```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def insert_begin(self,data):
        nod=Node(data)
        if self.head==None:
            self.head=nod
            return
        nod.next=self.head
        self.head=nod

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
```

head = None ✓



insert a node at the beginning of the liked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def insert_begin(self,data):
        nod=Node(data)
        if self.head==None:
            self.head=nod
            return
        nod.next=self.head
        self.head=nod

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
```

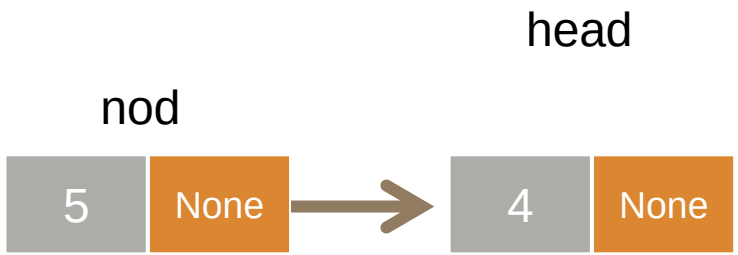


insert a node at the beginning of the liked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def insert_begin(self,data):
        nod=Node(data)
        if self.head==None:
            self.head=nod
            return
        nod.next=self.head
        self.head=nod

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
```

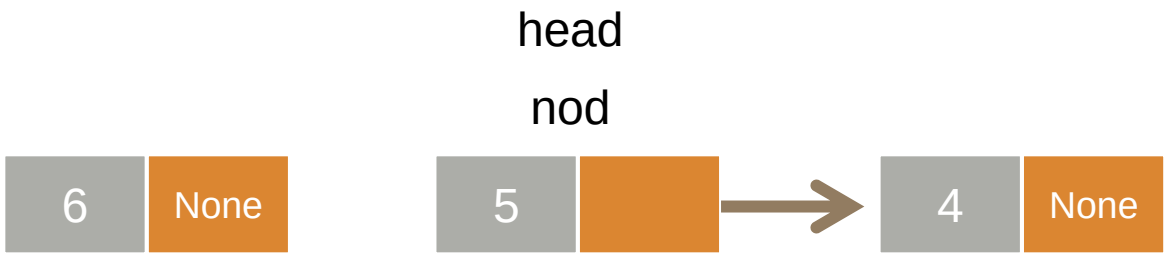
insert a node at the beginning of the liked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def insert_begin(self,data):
        nod=Node(data)
        if self.head==None:
            self.head=nod
            return
        nod.next=self.head
        self.head=nod

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
```



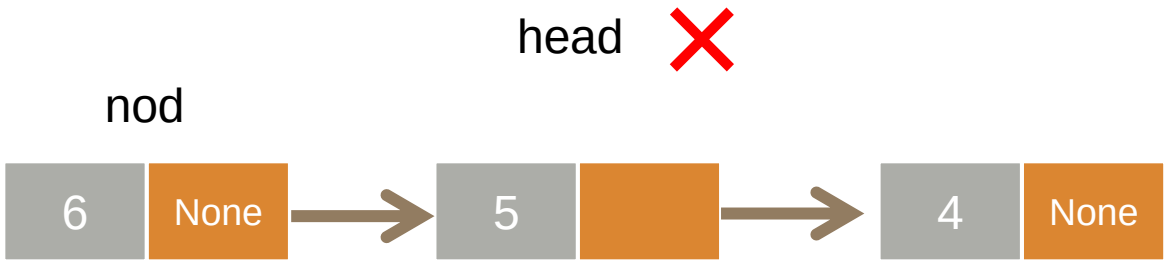
insert a node at the beginning of the liked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def insert_begin(self,data):
        nod=Node(data)
        if self.head==None:
            self.head=nod
            return
        nod.next=self.head
        self.head=nod

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
```



insert a node at the beginning of the liked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def insert_begin(self,data):
        nod=Node(data)
        if self.head==None:
            self.head=nod
            return
        nod.next=self.head
        self.head=nod

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
```



insert a node at the beginning of the liked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def insert_begin(self,data):
        nod=Node(data)
        if self.head==None:
            self.head=nod
            return
        nod.next=self.head
        self.head=nod

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
```

compute the length of the linked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def length(self):
        p=self.head
        count=0
        while p!= None:
            p=p.next
            count +=1
        return count

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
print(LL.length())
```

compute the length of the linked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def length(self):
        p=self.head
        count=0
        while p!= None:
            p=p.next
            count +=1
        return count

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
print(LL.length())
```

compute the length of the linked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def length(self):
        p=self.head
        count=0
        while p!= None:
            p=p.next
            count +=1
        return count

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
print(LL.length())
```

count = 0

p

head

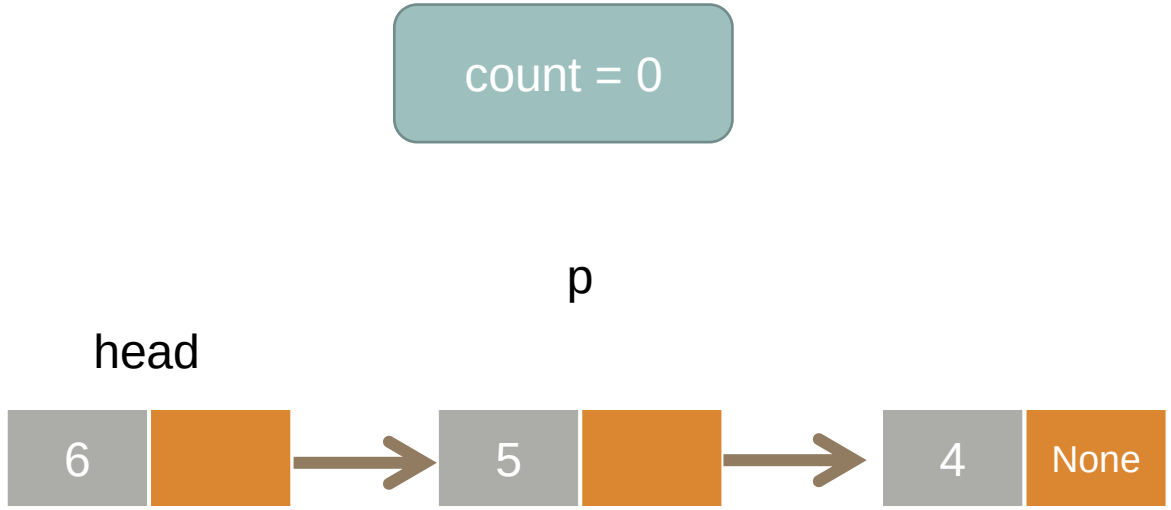
compute the length of the linked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def length(self):
        p=self.head
        count=0
        while p!= None:
            p=p.next
            count +=1
        return count

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
print(LL.length())
```

count = 0

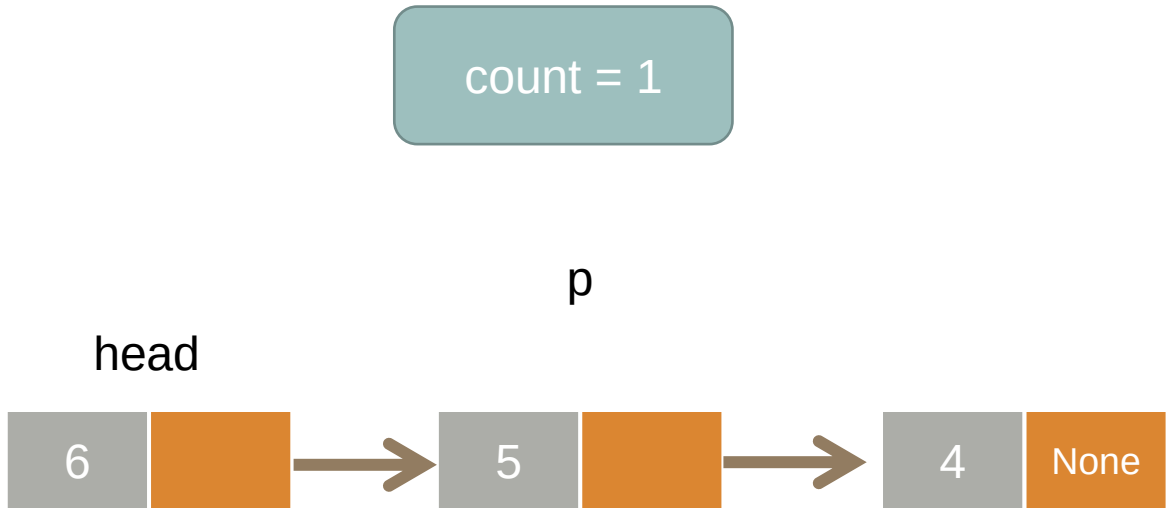
compute the length of the linked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def length(self):
        p=self.head
        count=0
        while p!= None:
            p=p.next
            count +=1
        return count

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
print(LL.length())
```



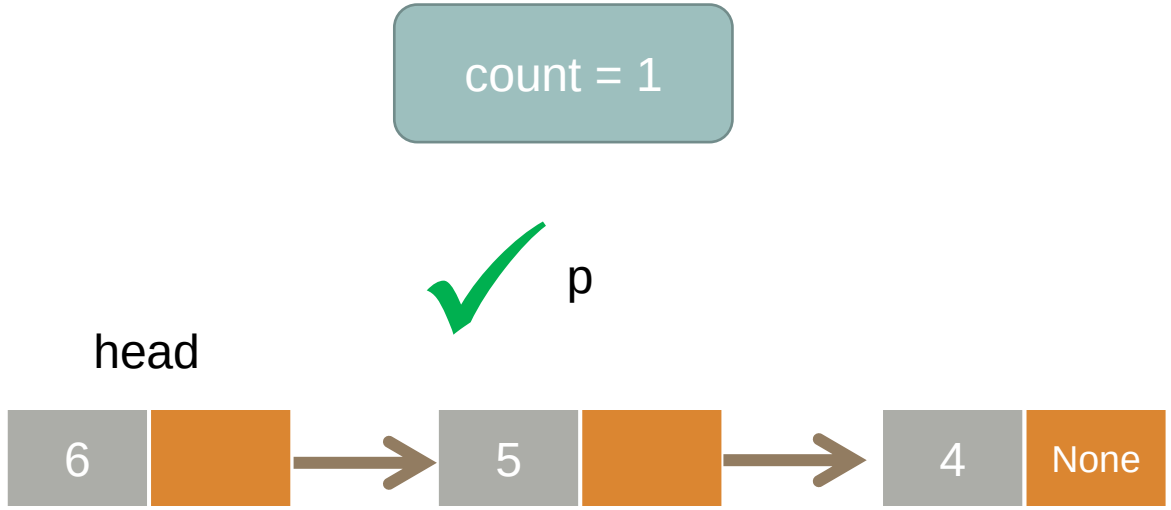
compute the length of the linked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def length(self):
        p=self.head
        count=0
        while p!= None:
            p=p.next
            count +=1
        return count

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
print(LL.length())
```



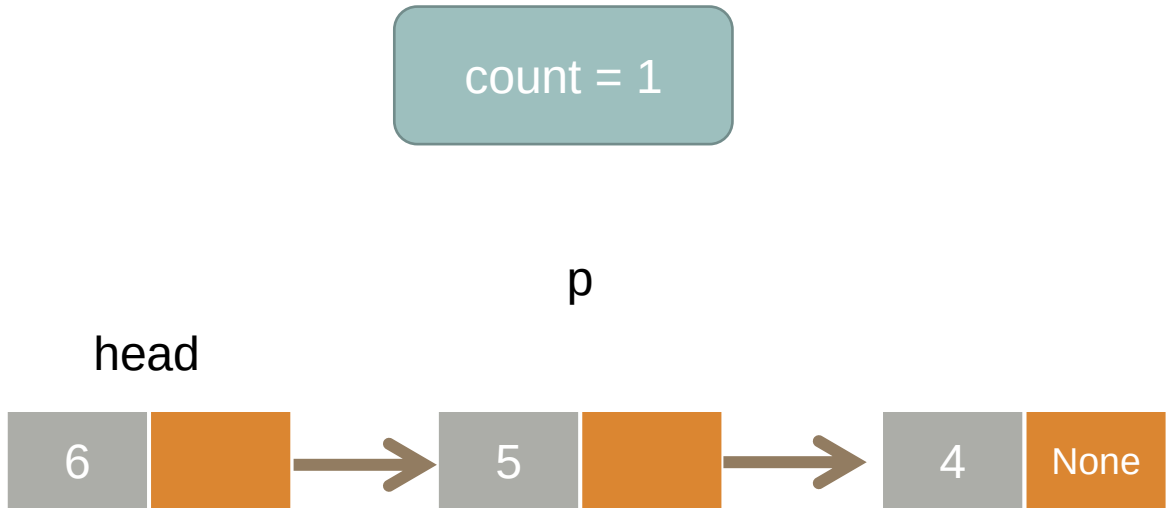
compute the length of the linked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def length(self):
        p=self.head
        count=0
        while p!= None:
            p=p.next
            count +=1
        return count

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
print(LL.length())
```



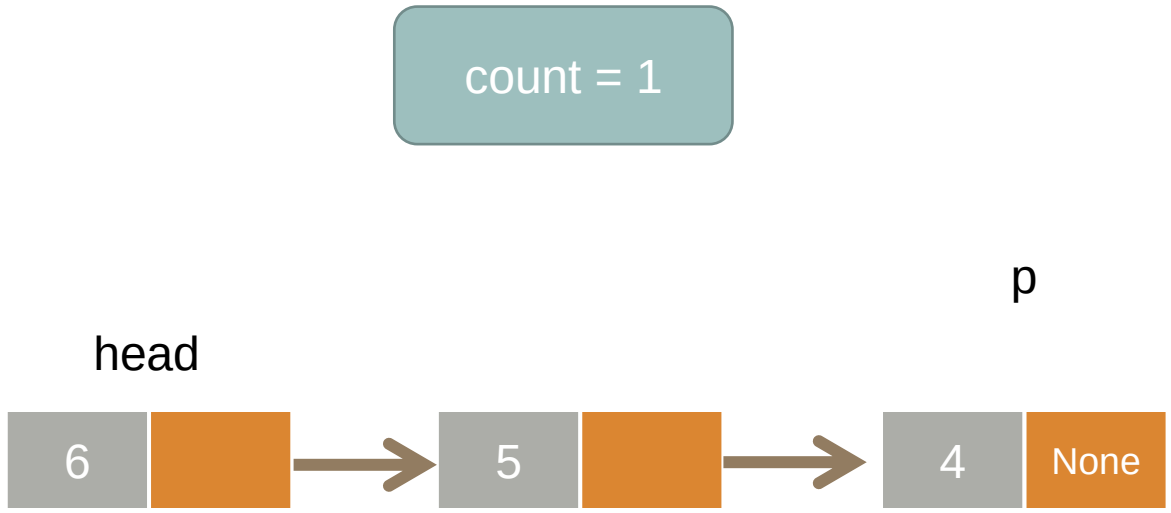
compute the length of the linked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def length(self):
        p=self.head
        count=0
        while p!= None:
            p=p.next
            count +=1
        return count

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
print(LL.length())
```



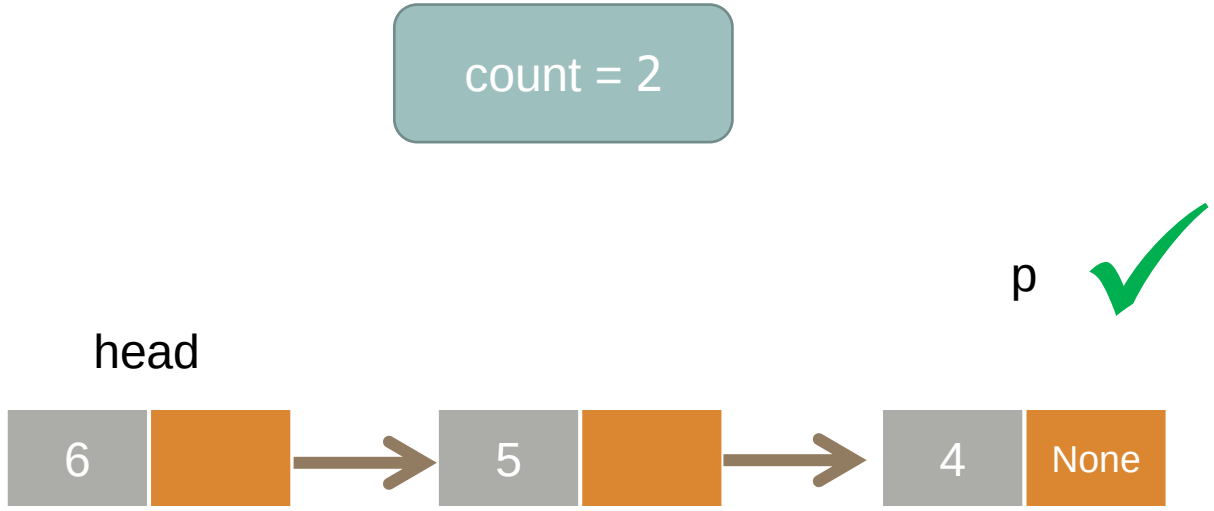
compute the length of the linked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def length(self):
        p=self.head
        count=0
        while p!= None:
            p=p.next
            count +=1
        return count

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
print(LL.length())
```



compute the length of the linked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def length(self):
        p=self.head
        count=0
        while p!= None:
            p=p.next
            count +=1
        return count

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
print(LL.length())
```



compute the length of the linked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def length(self):
        p=self.head
        count=0
        while p!= None:
            p=p.next
            count +=1
        return count

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
print(LL.length())
```

count = 2

p p = None

compute the length of the linked list

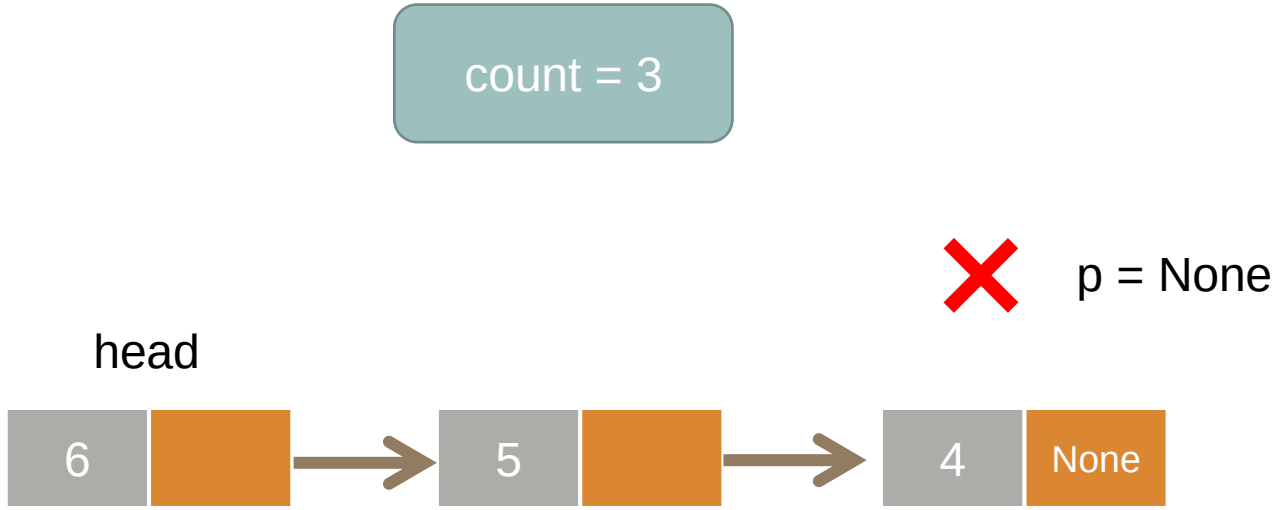
```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def length(self):
        p=self.head
        count=0
        while p!= None:
            p=p.next
            count +=1
        return count

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
print(LL.length())
```

count = 2

p = None

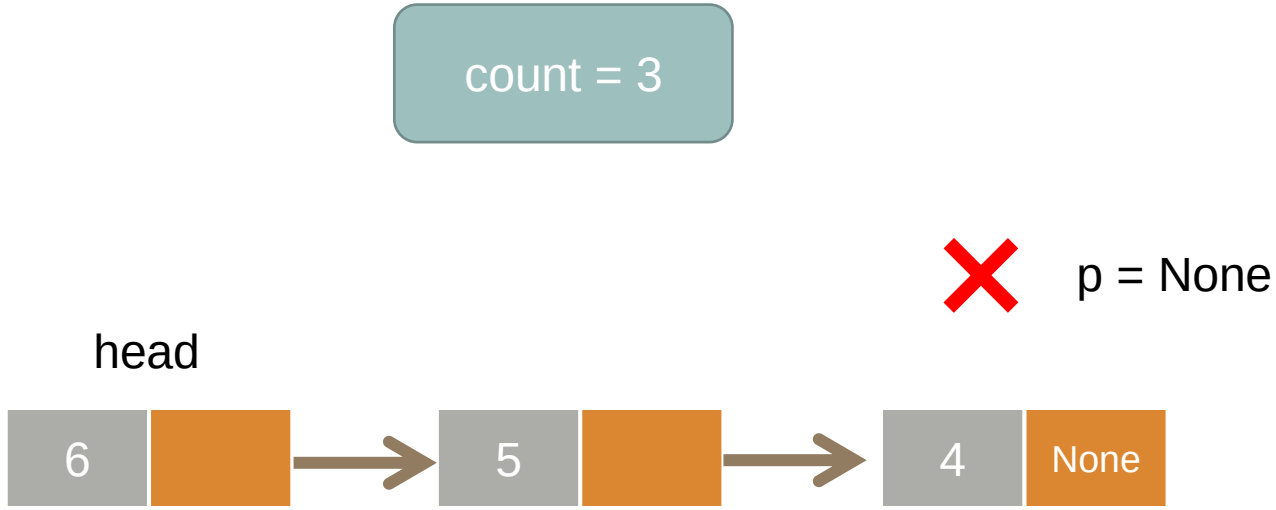
compute the length of the linked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def length(self):
        p=self.head
        count=0
        while p!= None:
            p=p.next
            count +=1
        return count

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
print(LL.length())
```



compute the length of the linked list

```
class Node:
    def __init__(self,data,next=None):
        self.data = data
        self.next = next
class linklist:
    def __init__(self,head=None):
        self.head=head
    def length(self):
        p=self.head
        count=0
        while p!= None:
            p=p.next
            count +=1
        return count

LL=linklist()
LL.insert_begin(4)
LL.insert_begin(5)
LL.insert_begin(6)
print(LL.length())
```



Thank you