Lists

Lists

- จากที่เราได้เขียน stack queue และ deque มาแล้วโดยใช้ list ของ Python เราพบว่า list มีประสิทธิภาพมากใช้งานง่าย กระบวนการการเก็บข้อมูล

- อย่างไรก็ตามไม่ใช่ว่าทุกภาษาโปรแกรมจะมี list ให้ใช้แบบ Python ซึ่งหากเป็นเช่นนี้ก็ต้องสร้างขึ้นมาเอง

List

- list เป็นกลุ่มของข้อมูลที่แต่ละสมาชิกในข้อมูลจะมีตำแหน่งสัมพันธ์กับตัวอื่นๆ
- เริ่มต้นเราสนใจ list แบบไม่เรียงลำดับ
- เราจะพิจารณา list ตามสมาชิกตัวที่ 1 สมาชิกตัวที่ 2 ... โดยเราสามารถอ้างถึงตัวแรกและตัวสุดท้ายของ list ได้ เพื่อความง่ายเราจะสมมติว่า list ไม่เก็บข้อมูลซ้ำกัน
- ตัวอย่างเช่นกลุ่มของข้อมูลจำนวนเต็ม 54,87,26,48,30 จะเป็น unordered list ของคะแนนสอบ

Unordered list abstract data type

- โครงสร้างของ unordered list เป็นกลุ่มของข้อมูลที่แต่ละข้อมูลมีความสัมพันธ์ของตำแหน่งกับข้อมูลอื่น มี operation ดังนี้
- List() สร้าง list เปล่า ไม่ต้องการ parameter และ return list ว่าง
- add(item) เพิ่มข้อมูลเข้า list ต้องส่งข้อมูลให้แต่ไม่ต้อง return สมมติว่าข้อมูลที่เพิ่มไม่มีใน list
- remove(item) ลบข้อมูลออกจาก list ต้องการข้อมูลที่จะลบ สมมติว่ามีข้อมูลนั้นใน list

Unordered list abstract data type

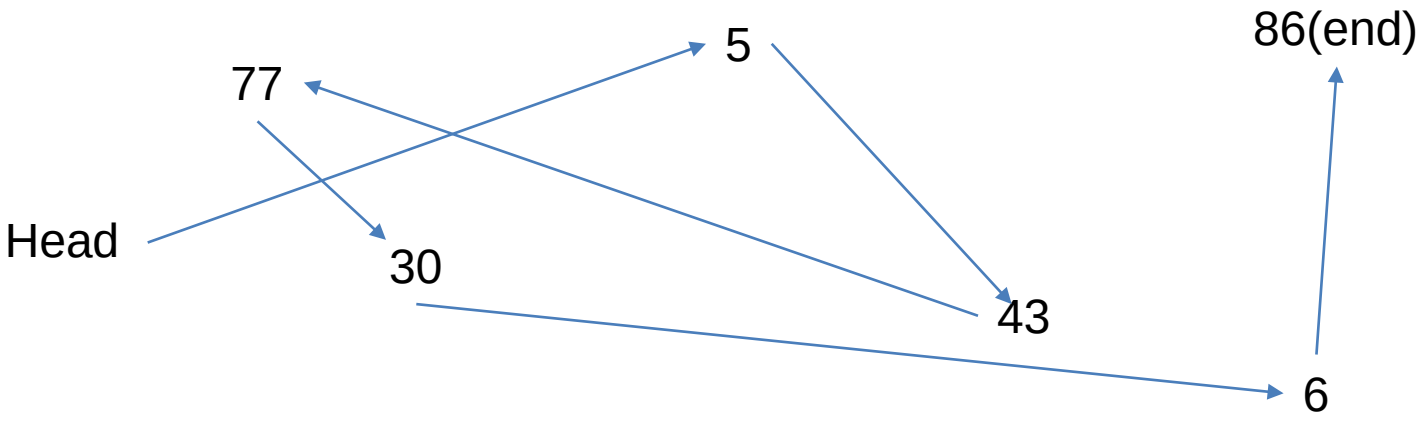
- search(item) ค้นหา item ใน list ต้องการข้อมูลที่จะค้นหาและคืนค่า boolean
- is_empty() ตรวจสอบว่า list ว่างไหม ไม่ต้องการ parameter คืนค่า boolean
- size() คืนค่าจำนวนของข้อมูลใน list ไม่ต้องการ parameter คืนค่าเป็นจำนวนเต็ม
- append(item) เพิ่มข้อมูลลงท้าย list ต้องการข้อมูลที่จะเพิ่ม ไม่คืนค่า สมมติว่าข้อมูลที่จะเพิ่มไม่มีใน list

Unordered list abstract data type

- index(item) คืนค่าตำแหน่งของข้อมูลใน list ต้องการข้อมูล คืนค่าเป็นตำแหน่ง สมมติว่าข้อมูลอยู่ใน list
- insert(pos,item) เพิ่มข้อมูลใหม่ลงในตำแหน่ง pos ต้องการข้อมูล ไม่คืนค่า สมมติว่าข้อมูลไม่มีใน list และมีตำแหน่งว่างพอที่จะใส่ข้อมูล
- pop() เอาข้อมูลออกและคืนค่าข้อมูลสุดท้ายใน list ไม่ต้องการ parameter และคืนค่า สมมติว่ามีข้อมูลอย่างน้อย 1 ตัว
- pop(pos) เหมือน pop แต่เอาตัวที่ตำแหน่ง pos ออก

Implement an unordered list: linked list

- ในการสร้าง unordered list โดยทั่วไปเราจะสร้างเป็น linked list
- สิ่งที่เราต้องการคือ ต้องแน่ใจว่าเราเก็บตำแหน่งที่สัมพันธ์กันของข้อมูล อย่างไรก็ตามไม่จำเป็นที่ต้องเก็บในตำแหน่งที่ต่อเนื่องกันในหน่วยความจำ



Implement an unordered list: linked list

- ทำให้เห็นว่าค่าต่างๆ สามารถถูกเก็บได้อย่างอิสระในหน่วยความจำ ถ้าเราสามารถเก็บตำแหน่งที่เก็บของแต่ละข้อมูลได้ ในรูปเราจึงมองว่ามีลูกศรชี้ไปว่าข้อมูลต่อไปเป็นตัวไหน
- ดังนั้นมันสำคัญมากที่จะเก็บตำแหน่งของจุดเริ่มต้น
- เมื่อเรารู้ตำแหน่งของข้อมูลตัวแรกแล้วจะสามารถบอกตัวที่สองและตัวต่อๆ ไปได้
- โดยทั่วไปจะเริ่มต้นที่ head ของ list และข้อมูลสุดท้ายจะไม่ชี้ไปข้อมูลอื่น

Node class

- ในการสร้าง list เราจะสร้าง node มาเป็นโครงสร้างพื้นฐานในการเก็บข้อมูล
- แต่ละ node จะเก็บข้อมูลหลัก 2 อย่างคือ ข้อมูล(data field) และที่อยู่ของ node ต่อไป (next field)
- ในการสร้าง node จะต้องกำหนดค่าเริ่มต้นให้กับ 2 ข้อมูลข้างต้น

Node class

```
class Node:
    def __init__(self,init_data):
        self.data = init_data
        self.next = None
    def get_data(self):
        return self.data
    def get_next(self):
        return self.next
    def set_data(self, new_data):
        self.data = new_data
    def set_next(self, new_next):
        self.next = new_next
```

Node class

- หากเราต้องการสร้าง Node ใหม่จะสร้างด้วย

  temp = Node(75)

  print(temp.get_data**()) # 75**

- เมื่อ node ใหม่ถูกสร้าง node ถัดไปจะเป็น None หมายความว่าไม่มี node ตัวถัดไป
- จากที่เราบอกว่า unordered list จะถูกสร้างจากกลุ่มของ node ซึ่งแต่ละอันจะถูกเชื่อมกันด้วย next ซึ่งเป็นที่อยู่ของ node ถัดไป

The unordered list class

ดังนั้นเริ่มต้น


```
def __init__(self):
    self.head = None
```

เริ่มต้นเราจะสร้าง list ที่ไม่มีข้อมูลเลย เราจะสร้างด้วย

```
mylist = UnorderedList()
```

นั่นคือเริ่มต้น head ไม่ชี้อะไรเลย

The unordered list class

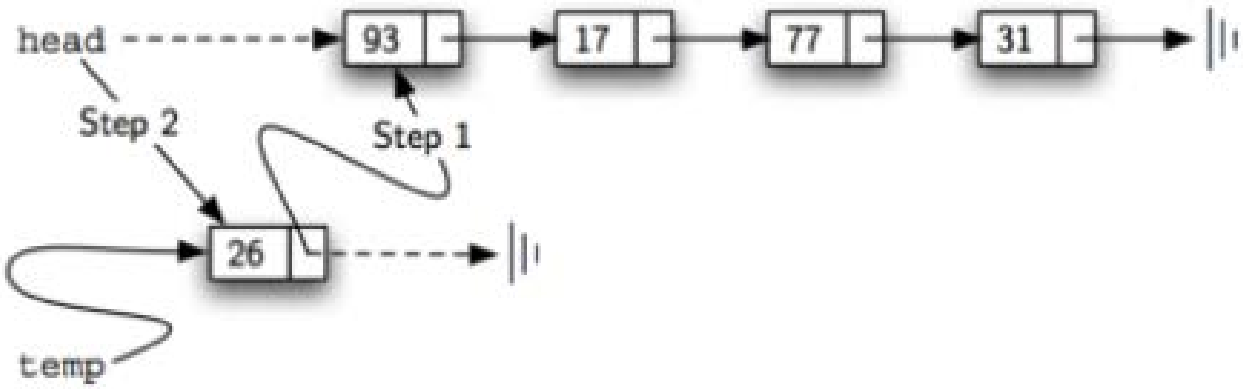
- is_empty() จะเป็นการตรวจสอบว่า head ของ list ว่างไหม ผลลัพธ์ที่ได้จะเป็น boolean จาก self.head == None
- เป็นจริงเมื่อไม่มี node ใน list
- ดังนั้น list ใหม่จะ empty

```
def is_empty(self):
        return self.head == None
```

The unordered list class

- การนำข้อมูลเข้าไปใน list
- เราต้องสร้าง function ในการเพิ่มข้อมูล อย่างไรก็ตามก่อนที่เราจะสร้าง เราต้องตอบคำถามอย่างหนึ่งก่อน ว่าเราจะเพิ่มข้อมูลใหม่ไปใน liked list ของเราตรงไหน
- เนื่องจากว่าตอนนี้เราสนใจ unordered linked list ตำแหน่งของข้อมูลใหม่จะไปอยู่ที่ใดก็ได้ ดังนั้นเราจะเพิ่มข้อมูลไปไว้ตำแหน่งที่ทำง่ายที่สุด

The unordered list class



- จากโครงสร้างของ linked list นั้นมีจุดทางเข้าโครงสร้าง 1 จุด ตรง head
- Node ทุก node จะถูกไปถึงได้ก็ต้องเริ่มที่ head จุดที่จะเพิ่มข้อมูลง่ายที่สุดคือจุดนี้โดยเราจะสร้าง node ใหม่ แล้วให้ node ใหม่ชี้ข้อมูลเก่า จากนั้นให้มันเป็น head

The unordered list class

- mylist.add(31)
- mylist.add(76)
- mylist.add(61)

หากคำสั่งเป็นเช่นนี้ node 31 จะถูกเพิ่มเข้ามา จากนั้น node 76 จะมาอยู่ก่อนหน้า จากนั้น node 61 จะมาอยู่หน้าสุด

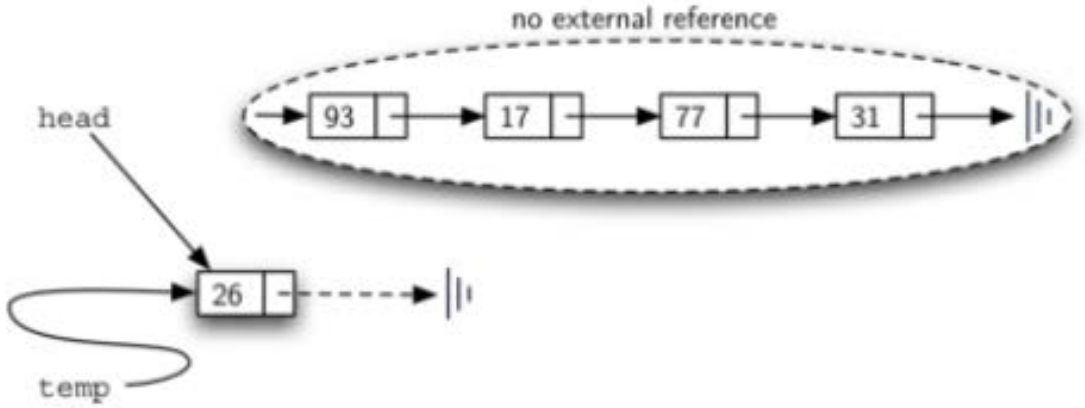
ดังนั้นฟังก์ชัน (method) add จะสร้าง node ใหม่เพื่อเก็บข้อมูล จากนั้นเชื่อม node ใหม่เข้ากับโครงสร้างเก่า (ห้ามทำให้สาย list ขาด)

The unordered list class

```
def add(self, item):
        temp = Node(item)
        temp.set_next(self.head)
        self.head = temp
```

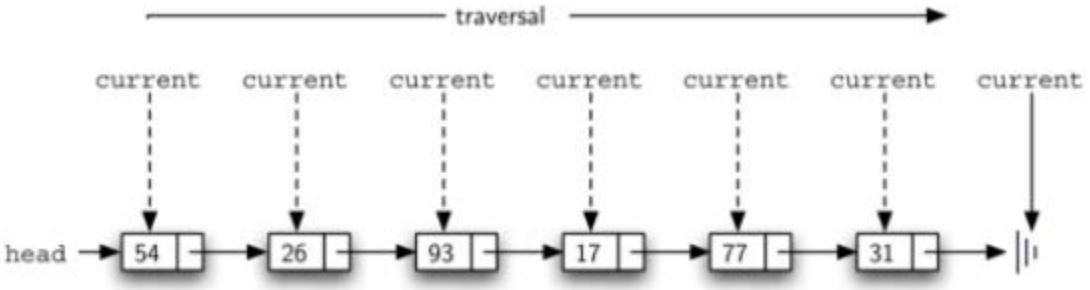
ลำดับของคำสั่งในบรรทัดที่ 3 และ 4 สำคัญมาก

หากสลับที่จะเกิดอะไรขึ้น



The unordered list class

- ต่อไปเราจะมาดู size search remove ซึ่งทั้ง 3 อันนี้จะต้องมีการท่องเข้าไปใน list เพื่อตรวจสอบ การท่องไปใน list เรียกว่า linked list traversal
- โดยการ traversal นี้จะเริ่มที่ node แรกใน list จากนั้นก็จะไปต่อทีละ node ตามที่ next บอก



The unordered list class

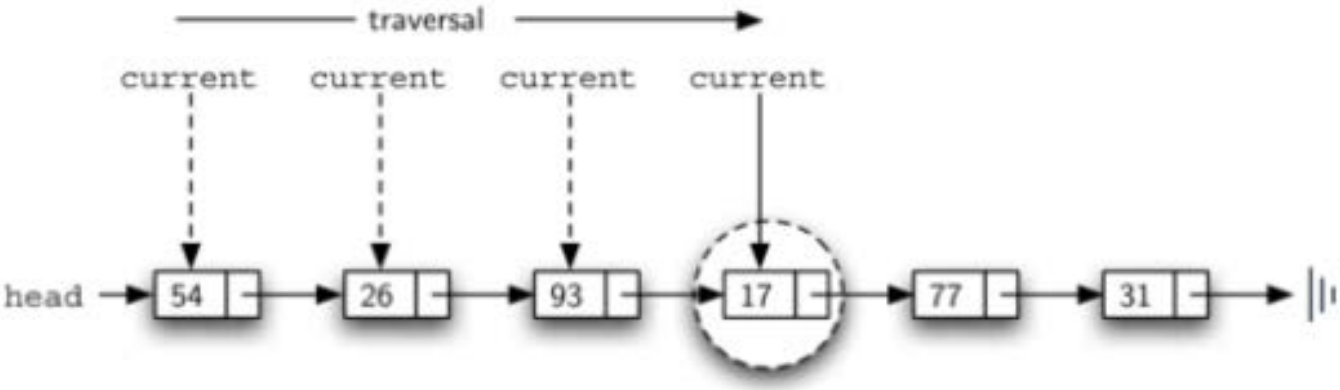
- Method size เราก็จะท่องไปใน list ตั้งแต่ตัวแรกจนตัวสุดท้าย โดยที่นับจำนวน node ที่ผ่านไปด้วย
- เราจะมีตัวแปรที่ชื่อ current เอาไว้อ้างอิงว่าตอนนี้อยู่ที่ node ไหน ดังนั้นเริ่มต้นให้ current เป็น head ของ list
- เริ่มต้น เรายังไม่ได้ท่องเข้าไปใน list จึงให้ count เป็น 0
- จากนั้นเราก็จะท่องไปใน list จนกว่าจะเจอ None คือท้าย list

The unordered list class

```
def size(self):
        current = self.head
        count = 0
        while current != None:
                count = count + 1
                current = current.get_next()
        return count
```

The unordered list class

- Method search การค้นหาค่าใน linked list ใน unordered linked list ใช้หลักการการท่องไปใน list เนื่องจากว่าข้อมูลไม่ได้เรียงลำดับ ข้อมูลสามารถอยู่ตรงไหนก็ได้ใน list
- เราก็จะแวะไปทีละ node ใน linked list ถามว่ามีข้อมูลที่ต้องการเก็บอยู่ไหม ถ้าเราถามครบทุก node แล้วไม่พบ แสดงว่าข้อมูลไม่มีอยู่ใน list ตัวอย่างการหาข้อมูล 17 (มีใน List)



The unordered list class

```
def search(self,item):
        current = self.head
        found = False
        while current != None and not found:
                if current.get_data() == item:
                        found = True
                else:
                        current = current.get_next()
        return found
```

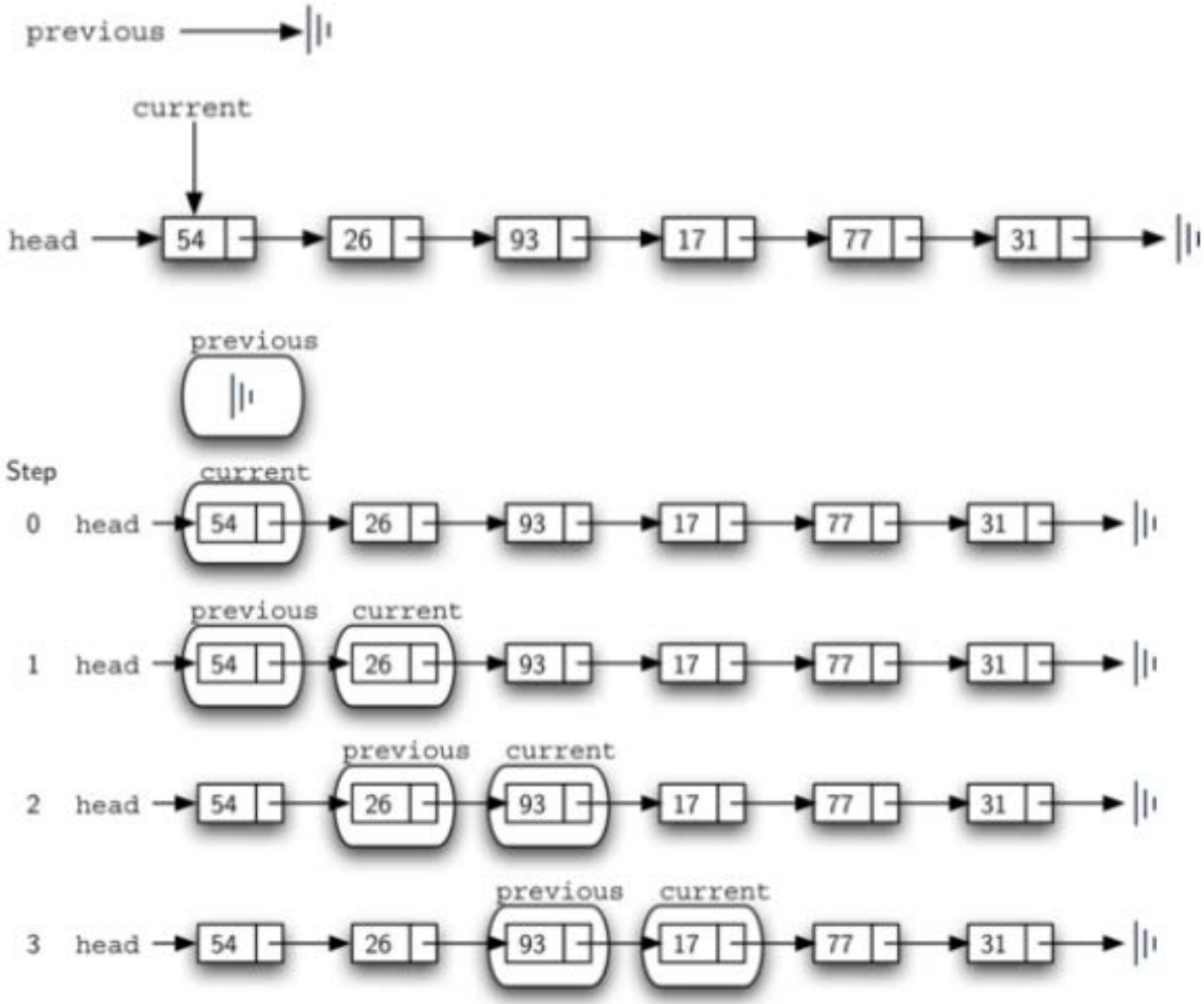
The unordered list class

- Method remove มี 2 ขั้นตอน
  - ขั้นตอนแรก เราต้องท่องไปใน list เพื่อหา item ที่เราต้องการลบ
  - ขั้นที่สองเมื่อเราพบแล้ว เราจะลบมัน
- ขั้นแรกเหมือนกับ traversal เริ่มต้นที่ head แล้วขยับไปเรื่อยๆ
- เมื่อเราพบแล้ว current จะเป็น node ที่เราจะลบ แล้วเราจะลบอย่างไรดี

The unordered list class

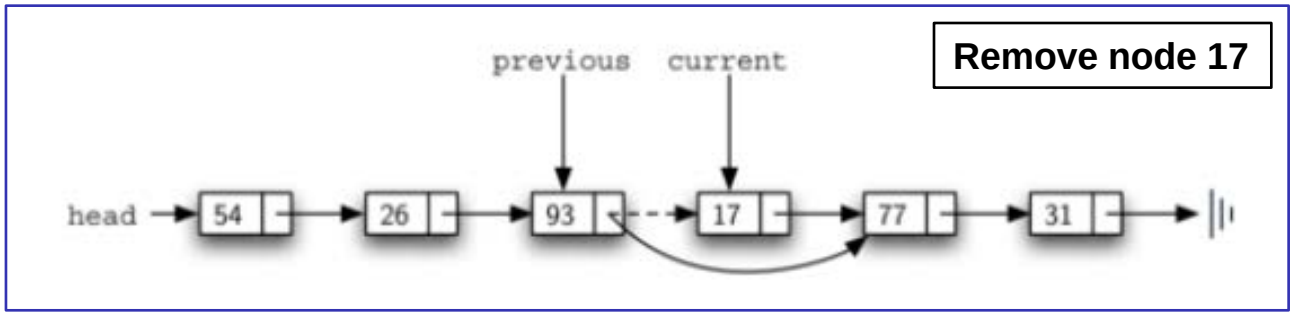
- ในการลบเราจะต้องย้าย link ของ node ก่อนหน้าข้าม current ไปชี้ node ถัดจาก current เลย
- แต่เราไม่มีฟังก์ชันชี้กลับ แล้วทำอย่างไรดี
- เราจะสร้างตัวแปรมาอีกตัว ชื่อว่า previous ที่จะเป็น node ก่อน current
- นั่นคือเมื่อ current หยุดที่ node ที่จะลบ previous จะอยู่ node ก่อนหน้า

The unordered list class



The unordered list class

```
def remove(self,item):
        current = self.head
        previous = None
        found = False
        while not found:
                if current.get_data() == item:
                        found = True
                else:
                        previous = current
                        current = current.get_next()
        if previous == None:
                self.head = current.get_next()
        else:
                previous.set_next(current.get_next())
```



The unordered list class

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

    def get_data(self):
        return self.data

    def get_next(self):
        return self.next

    def set_data(self, new_data):
        self.data = new_date

    def set_next(self, new_next):
        self.next = new_next
```

```
import Node

class UnorderedList:

    def __init__(self):
        self.head = None

    def is_empty(self):
        return self.head == None

    def add(self,item):
        temp = Node.Node(item)
        temp.set_next(self.head)
        self.head = temp

    def size(self):
        current = self.head
        count = 0
        while current != None:
            count = count + 1
            current = current.get_next()
        return count
```

```
def search(self,item):
    current = self.head
    found = False
    while current != None and not found:
        if current.get_data() == item:
            found = True
        else:
            current = current.get_next()
    return found

def remove(self,item):
    current = self.head
    if self.size() == 1 and current.get_data() == item:
        self.head = None
        found = True
    else:
        previous = None
        found = False
        while not found and current.get_next() != None:
            if current.get_data() == item:
                found = True
            else:
                previous = current
                current = current.get_next()
        if found:
            if previous == None:
                self.head = current.get_next()
            else:
                previous.set_next(current.get_next())
        else:
            print("The item", item, "not found in the list")
```

The unordered list class

- ลองใช้งาน

```
mylist = UnorderedList()

mylist.add(2)

print(mylist.search(2))

print(mylist.search(3))

print(mylist.size())

mylist.add(88)

print(mylist.size())
```

- ให้ลองเขียน method print_list() เพื่อแสดงข้อมูลใน list ทุกตัวทีละตัว ตั้งแต่ตัวแรกจนตัวสุดท้าย

Ordered list abstract data type

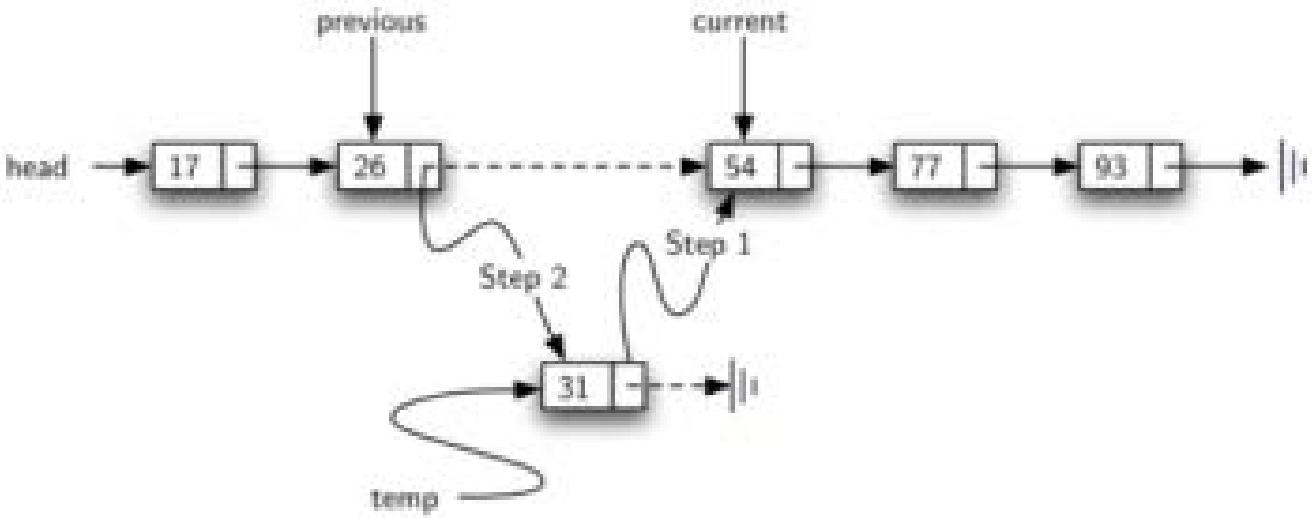
- ต่อไปเราจะพิจารณา list แบบเรียงลำดับ นั่นคือข้อมูลเรียงจากน้อยไปมาก เช่น 17, 26, 31, 54, 88, 100
- method ส่วนใหญ่จะเหมือนกับแบบ unordered list
- OrderedList() สร้าง ordered list เปล่า
- add(item) เพิ่ม item ใหม่เข้าไปใน list โดยเพิ่มให้ตำแหน่งถูกต้องตามลำดับ
- remove(item) ลบ item ใน list

Ordered list abstract data type

- is_empty() list ว่างไหม
- size() list มีขนาดเท่าไร
- index(item) คืนค่าตำแหน่งของ item ใน list
- pop() ลบและคืนค่าตัวสุดท้ายออก
- pop(pos) ลบ item ตำแหน่งที่ pos ออก

Ordered list abstract data type

- การเพิ่ม



- เราจะต้องไปหาตำแหน่งที่จะเพิ่ม นั่นคือ ตัวแรกสุดที่มากกว่าตัวที่เราจะเพิ่ม
- ทั้งนี้เราต้องเก็บข้อมูล node ก่อนหน้าไว้ด้วย เช่นเดียวกับคราวก่อนเราจะใช้ตัวแปร previous

Ordered list abstract data type

```
def add(self,item):
        current = self.head
        previous = None
        stop = False
        while current != None and not stop:
            if current.get_data()>item:
                stop = True
            else:
                previous = current
                current = current.get_next()
        temp = Node(item)
        if previous == None:
            temp.set_next(self.head)
            self.head = temp
        else:
            temp.set_next(current)
            previous.set_next(temp)
```

โจทย์: ให้จัดเก็บข้อมูลดังแสดงด้านล่างด้วย

- Unordered List
- Ordered List:
  - เรียงจากน้อยไปมาก
  - เรียงจากมากไปน้อย

Elephant, Bird, Pig, Cat, Dog, Horse, Ant, Snake