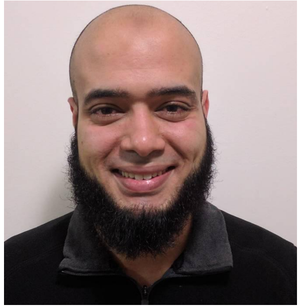
Data Structures

The Node

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

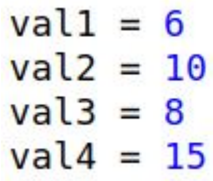


Recall: Array and List

- Arrays has fixed size. You can’t delete/insert/expand
- List was our way to get a dynamic array
    - append: create new memory, copy the data, add the element and remove old memory
- List **pros and cons**
    - Pros: Now more dynamic + still O(1) access to any position
    - Cons: Memory block **reallocation** and data copies during expansions = O(n)
    - Cons: Array is **contiguous** in memory, what if new requested memory is not available!?
- Can we avoid these **memory issues**?
    - E.g. **expanding** the content with a single element is always **O(1)**

Intuition

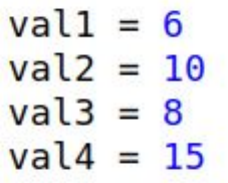
- We can create a single integer such as val1
    - Or any other data type or mix of types.
    - *Most of content is integers for simplicity*
- We can create several **separate** integers the same way
    - Here we show 4 separate **variables**
- We can expand with more separate values
- But this is not useful so far!
- We need them to be **linked,** not separate!



```
val1 = 6
val2 = 10
val3 = 8
val4 = 15
```

Intuition

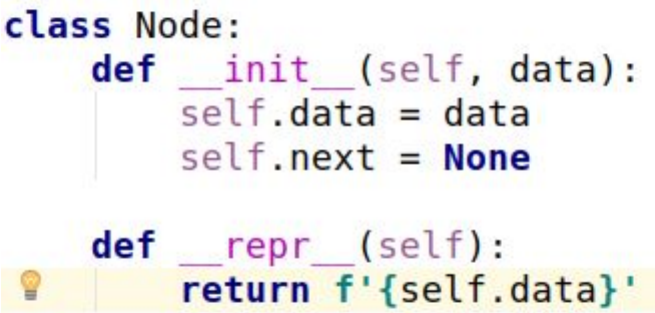
- Can we group them together?
- What if we also create other 4 objects to **link** them together?
    - Link from 1 to 2
    - Link from 2 to 3
    - Link from 3 to 4
    - Link from 4 to ?   Flag to stop!
    - I will call this kind of linkers as **pointers** (terminology from C/C++)
- What about a class with 2 variables?
    - Variable to hold the data
    - Variable to link to the other variable (the pointer)
    - Let’s call it a **Node**



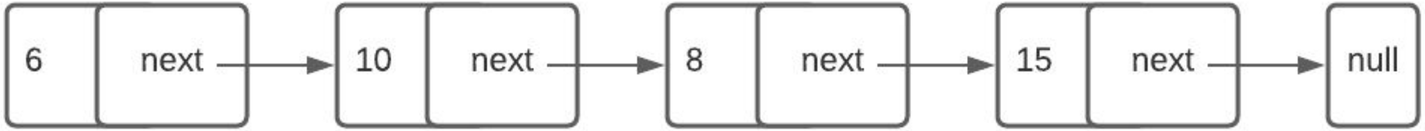
```
val1 = 6
val2 = 10
val3 = 8
val4 = 15
```

Node Data Structure

- If we create this class, we can easily include both things
  - The new data
  - And its link (next) to the next node
- Can you you this class to create in The memory the below connections?



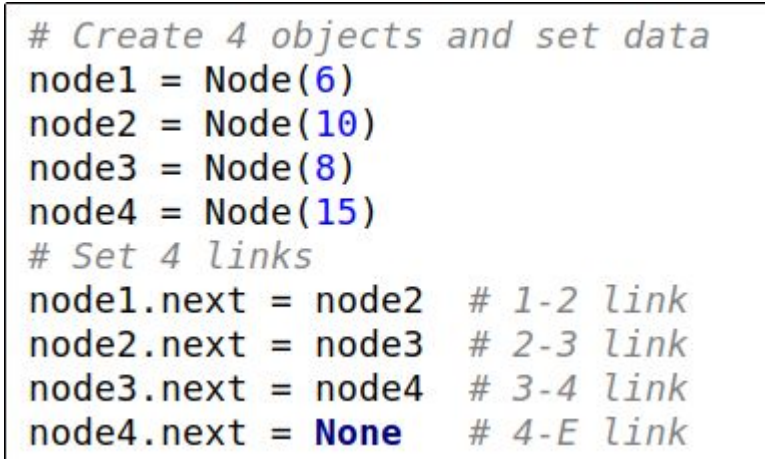
```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

    def __repr__(self):
        return f'{self.data}'
```



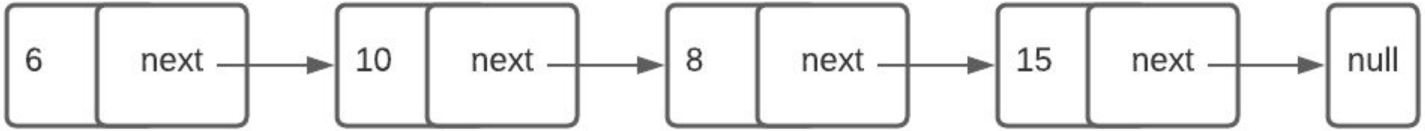
6 next → 10 next → 8 next → 15 next → null

Create and Link!

- Let’s create 4 objects, and set data
- Link each object to the next one
- To mark the last node, use None



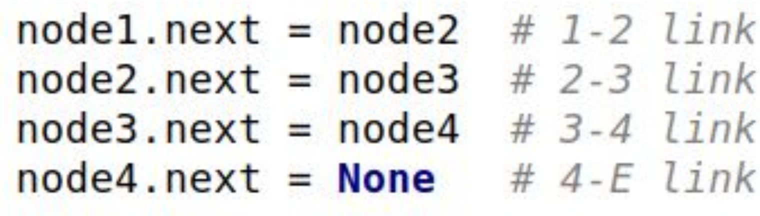
```python
# Create 4 objects and set data
node1 = Node(6)
node2 = Node(10)
node3 = Node(8)
node4 = Node(15)
# Set 4 links
node1.next = node2  # 1-2 link
node2.next = node3  # 2-3 link
node3.next = node4  # 3-4 link
node4.next = None   # 4-E link
```



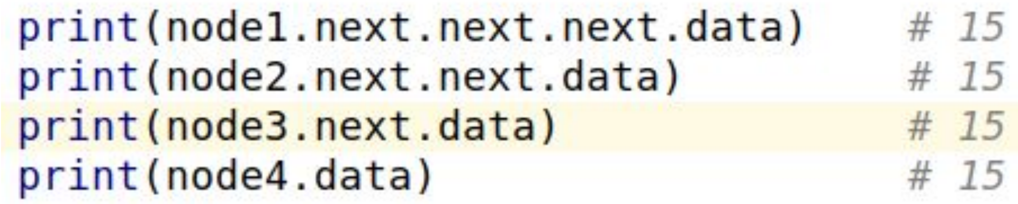
6 next → 10 next → 8 next → 15 next → null

Navigate!

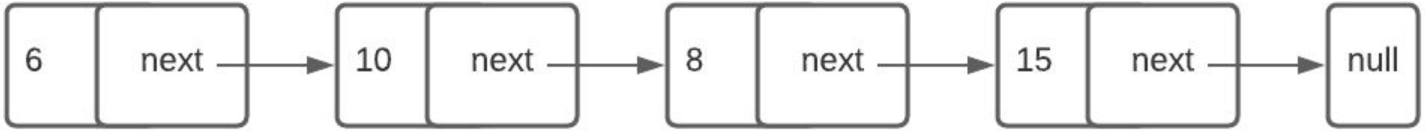
- Now, given **ONLY** the first node (**head**), we can move to any next node
  - next.next.next
- node1.next is node2
- node2.next is node 3
- Then node1.next.next is node 3



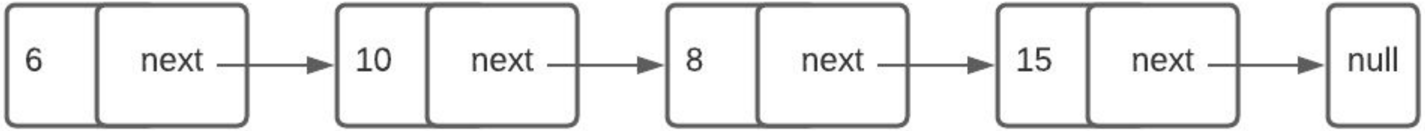
```
node1.next = node2   # 1-2 link
node2.next = node3   # 2-3 link
node3.next = node4   # 3-4 link
node4.next = None    # 4-E link
```

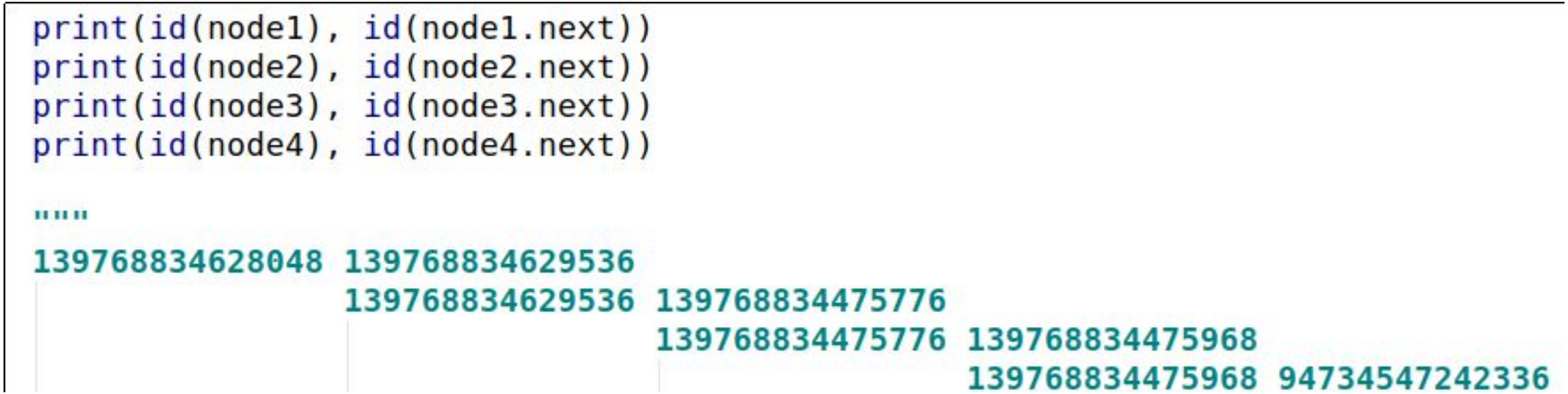


```
print(node1.next.next.next.data)     # 15
print(node2.next.next.data)          # 15
print(node3.next.data)               # 15
print(node4.data)                    # 15
```



Memory details

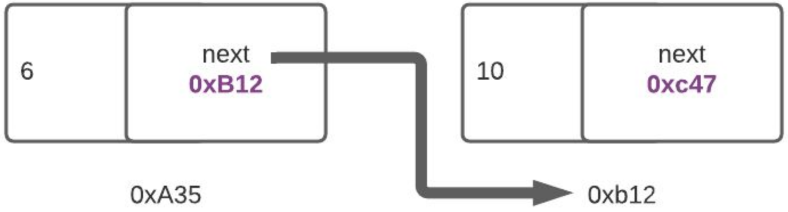




```
print(id(node1), id(node1.next))
print(id(node2), id(node2.next))
print(id(node3), id(node3.next))
print(id(node4), id(node4.next))

"""
139768834628048 139768834629536
                139768834629536 139768834475776
                                139768834475776 139768834475968
                                                139768834475968 94734547242336
```

Memory details

- Each node has an address
  - id(node). In image referred with 0xSomething
- Each node.next is the **linked** node
  - This is the other node and has it is ID
- You may use these 2 addresses to debug for mistakes



Your turn

- Make sure you understand this content fully
- Create and link nodes
- Play with them
- Be careful with the last node if its value is None
    - Don’t get its next as no such thing (None.next!)

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."