Data Structures & Algorithms in Python

Instructor: Aung Hein

Kaung Sein Technology

HELLO!

I am Aung Hein

Bachelor of Computer Science (UCSY)

Master of Science in IT & Data Science (EIU,Paris)

Cyber Security & Quantum Computing Enthusiasts

Kaung Sein Technology

Data Structures and Algorithms in Python

Data Structures

0. Intro to Data Structures
1. Stack, Queues & Deques
2. Link List
3. Trees
4. Dictionaries
5. Sets and Frozen Sets
6. Recursion, …

Algorithms

0. Algorithmic Thinking
1. Depth-First Search
2. Breadth-First Search
3. A* Search
4. Quicksort
5. Selection Sort
6. Greedy Algorithms, …

Kaung Sein Technology

2

Linked Lists

Kaung Sein
Technology

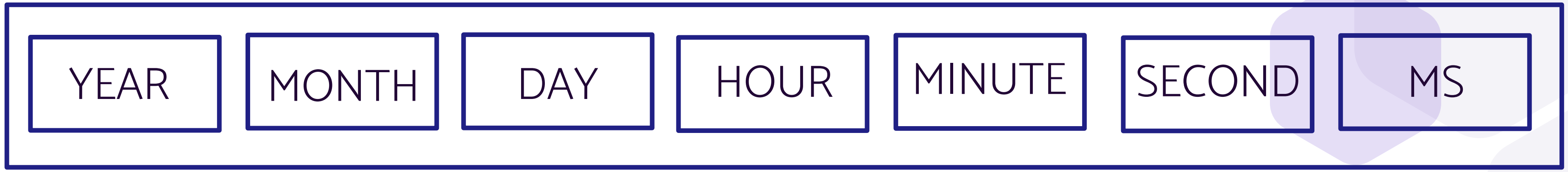
Lists

class list

- The Python list, despite its lowercase spelling in Python, this is actually a class, the list class

Lists (Cont’d)

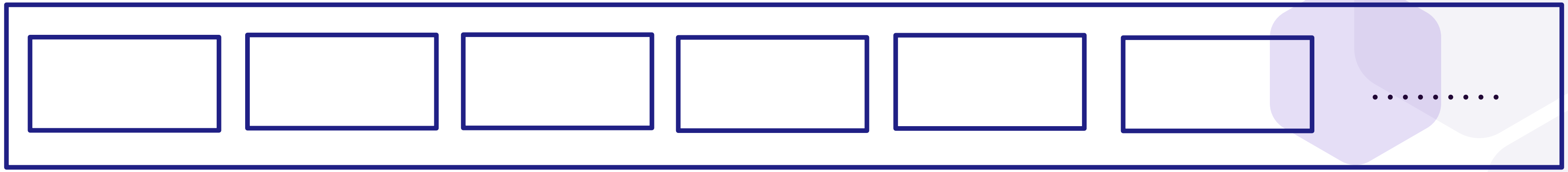
class datetime



- A datetime class, have a limited amount of data they will ever hold in them
- Python knows exactly how much memory it needs to store it

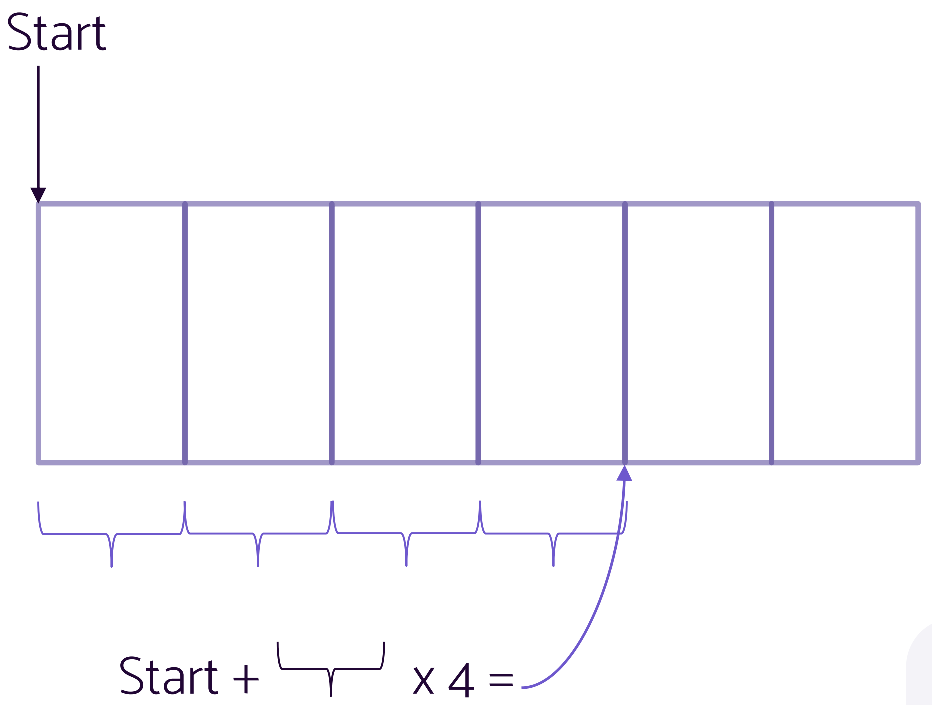
Lists (Cont'd)

class list

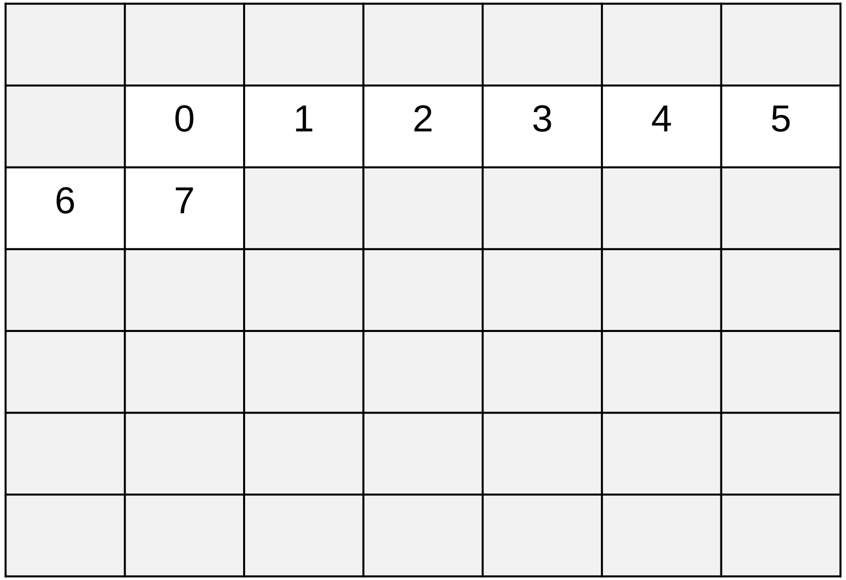


- What about a list?
- How big is a list?
- Well, it can be any size. It can be zero elements long
- it can be 500 million elements long, there's just no telling

Linked Lists (Cont’d)



Linked Lists (Cont’d)



- So when we create a list, Python needs to find an empty chunk of memory that can comfortably accommodate a list that will potentially grow.
- Let’s say we assume our list will hold no more than eight items, so we find a chunk of memory that’s big enough to hold those eight items.
- it’s super fast

Linked Lists (Cont'd)

- But what happens when we keep adding to that list and it gets too big for its current space?
- Well, the list needs to be moved
- This requires moving every single value from the old list to the new and bigger list, and this operation takes a lot of time
- The eighth item added to the list is fast
- ninth item takes a very, very, very long time, longer than if we had just allocated the right amount of space for our list in the first place.

Array Declaration in Java

```
myNumberArray = new int[20];
```

- declare the maximum number of things your list will hold upfront
- For example, this array declaration in Java
- This is declaring a new integer array, the Java version of a list, that is 20 integer long
- That way, it never has to reallocate memory or move it around. You can tell upfront how big the list is going to be

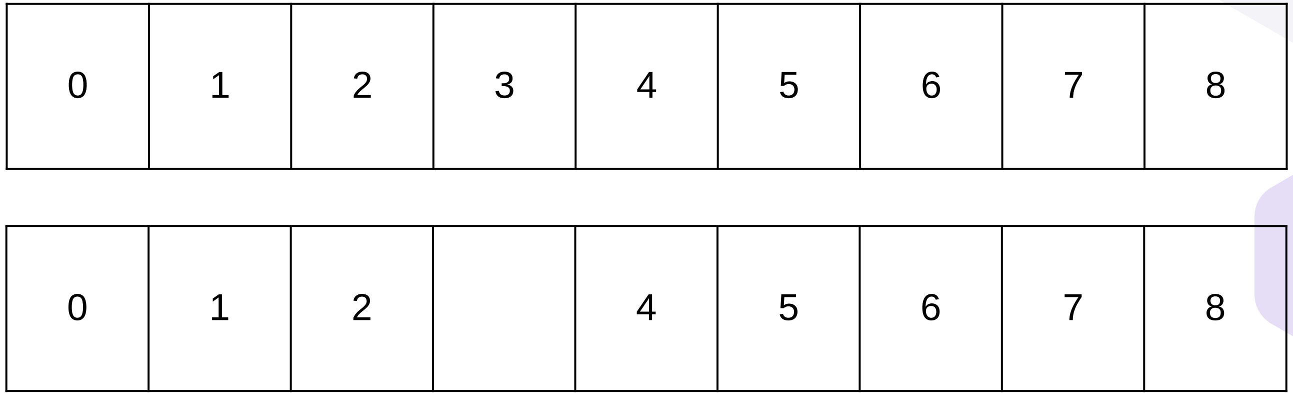
Kaung Sein Technology

Linked List (Cont’d)

0	1	2	3	4	5	6	7	8

- we run into issues if you delete items from the middle of a list. If an item is deleted, you have to take all of the following items and move them one by one. And this is a problem that the Python tuple has solved very nicely

Linked List (Cont'd)



- we run into issues if you delete items from the middle of a list.

Linked List (Cont'd)

0	1	2		4	5	6	7	8

0	1	2	4	5	6	7	8	

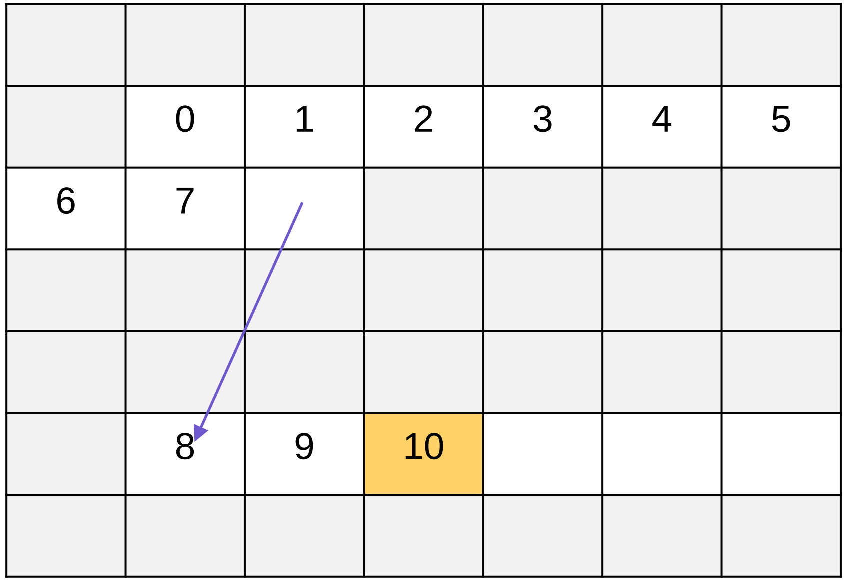
- If an item is deleted, you have to take all of the following items and move them one by one. And this is a problem that the Python tuple has solved very nicely

Tuple Declaration in Python

```
myTuple = (1, 2, 3)
```

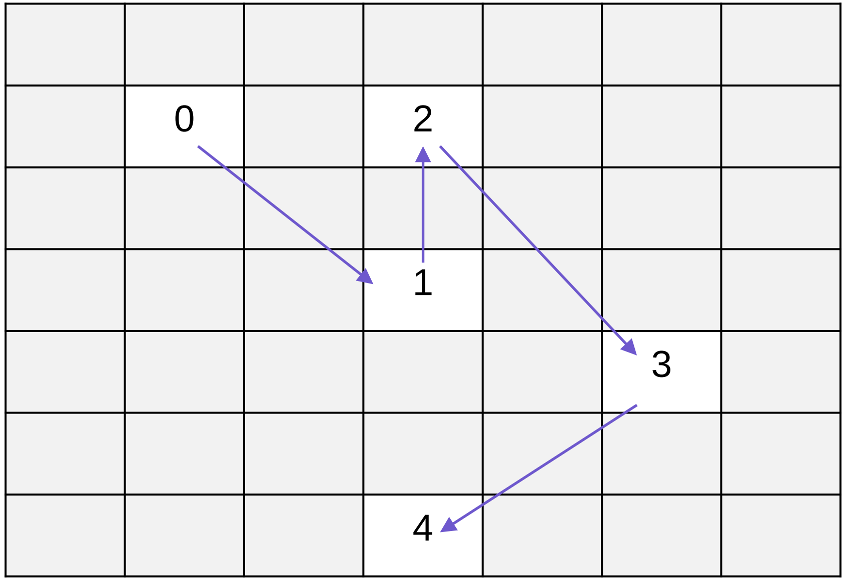
- Simply never let people add or remove items. You declare exactly how big it will ever be, but you can't add or remove them

Lists (Cont'd)



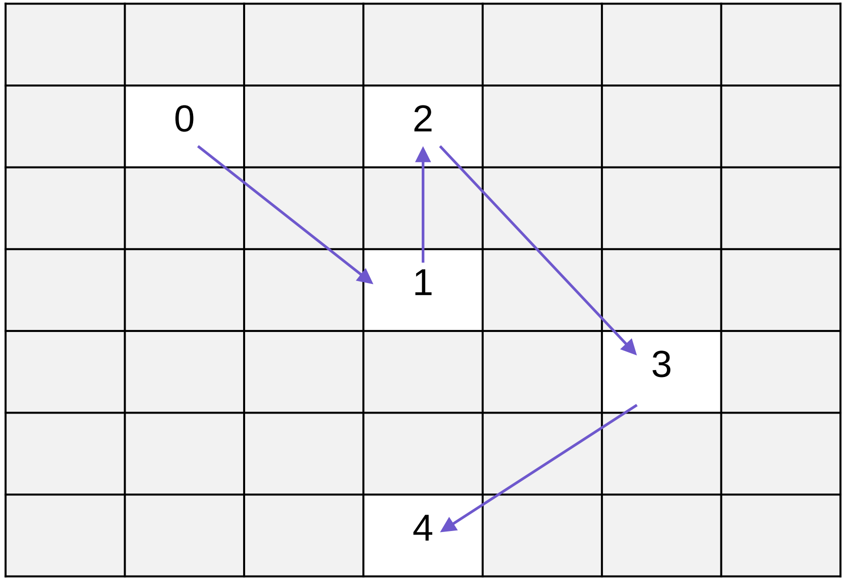
- what if when you ran out of memory, you just added a little note that said hey, the list continues over here in this block, and then you continue on with your list
- Now, one problem is that if you want to jump to the the last element of the list, you have to jump to the end of this one first, read the address where it continues, and then jump to the next list
- You can't just jump straight to the segment of memory that you want.

Lists (Cont’d)



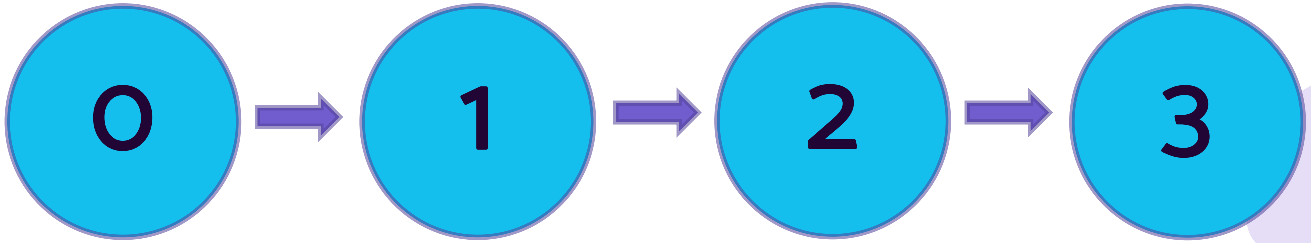
- What if every single element simply had a pointer to where the next element was?
- When you declare a new list, then you never need to allocate more than one chunk of memory because the next item could be stored here and there

Lists (Cont'd)



- You can trivially, add and delete items from the list without having to rearrange anything else
- You just skip over the node you want to remove, and everything else can stay in place
- The only thing you give up is your ability to jump to any random index in the list

Linked List


$$\begin{tikzcd} 0 \arrow[r] & 1 \arrow[r] & 2 \arrow[r] & 3 \end{tikzcd}$$

Applications of Linked Lists

- Rendering animation frames
- “Undo” functionality in a document editor

- What about “Redo” functionality?

- Any situation in which you never need to jump directly to an arbitrary index and where you really only care about the current and the next element, a linked list may be ideal

Let's write the linked list program

- Go to exercise 1

Quiz

- 1. What code leaves the value “current” pointing to the last node of a non-empty list?

```
current = current.head
while current.next:
        current = current.next
```

```
current = current.head
while current:
        current = current.next
```

```
current = current.head
while current.next:
        current = current.next.next
```

Kaung Sein
Technology

Quiz Answers

1.
```
current = current.head
while current.next:
        current = current.next
```

Let’s write the linked list searching program

- Go to exercise 2

Kaung Sein
Technology

Increasing Recursion Limit

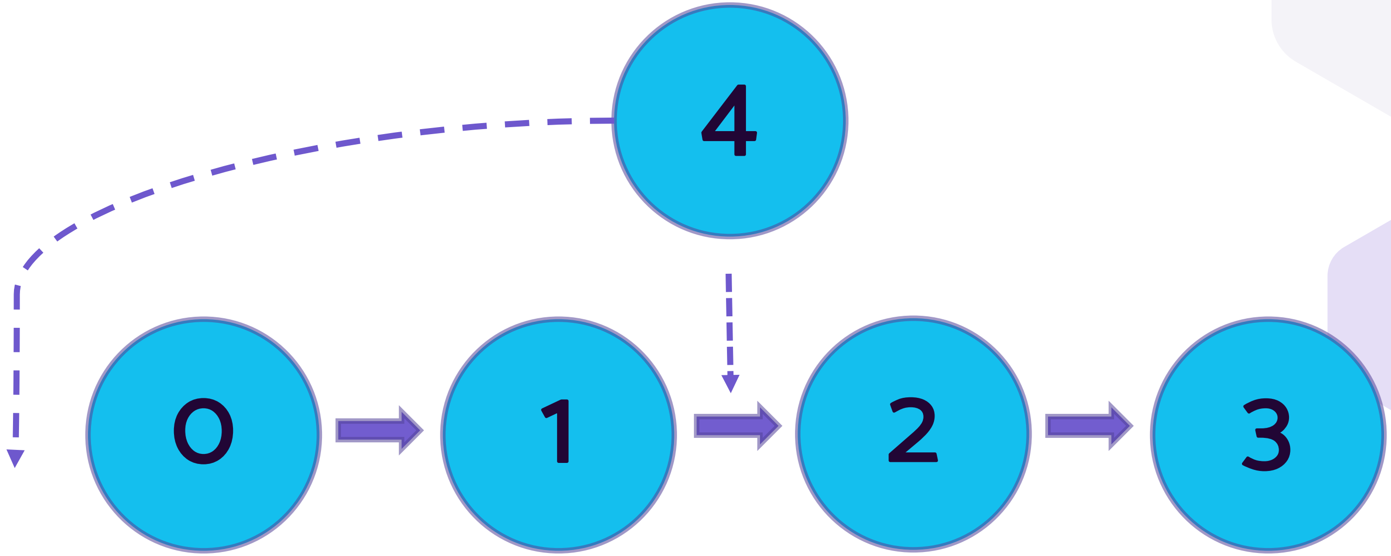
- Not a robust solution
- Stacks take time and memory to create

Kaung Sein Technology

Let's write the linked list deleting program

- Go to exercise 3

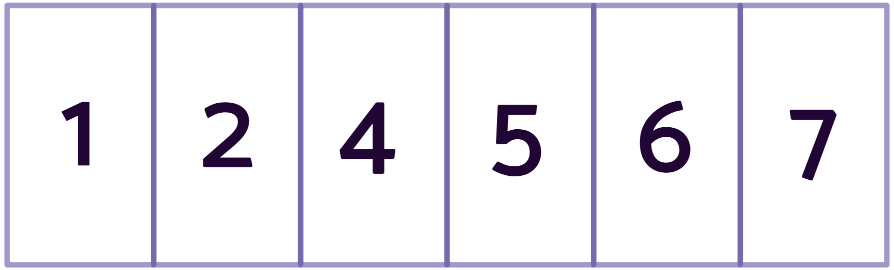
Insert a Node



Kaung Sein Technology

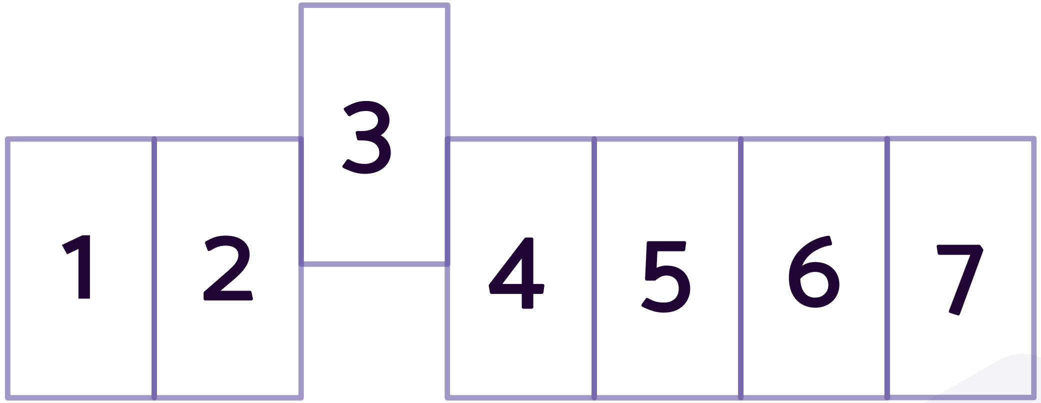
Insert a Node (Cont'd)

- In a traditional list, you have to shuffle all the subsequent elements around to make room for a new one.

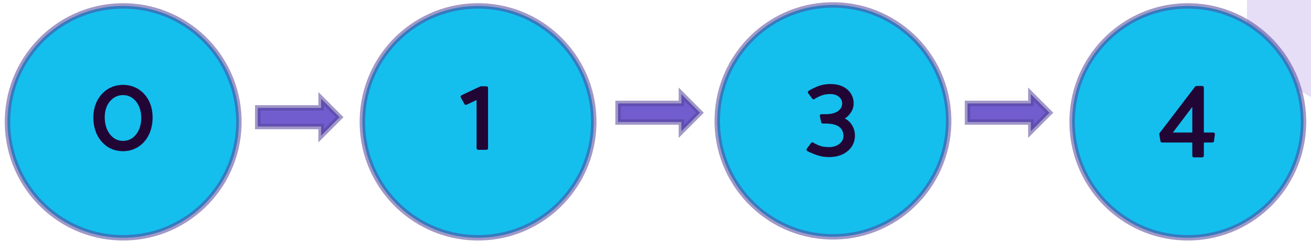


Insert a Node (Cont'd)

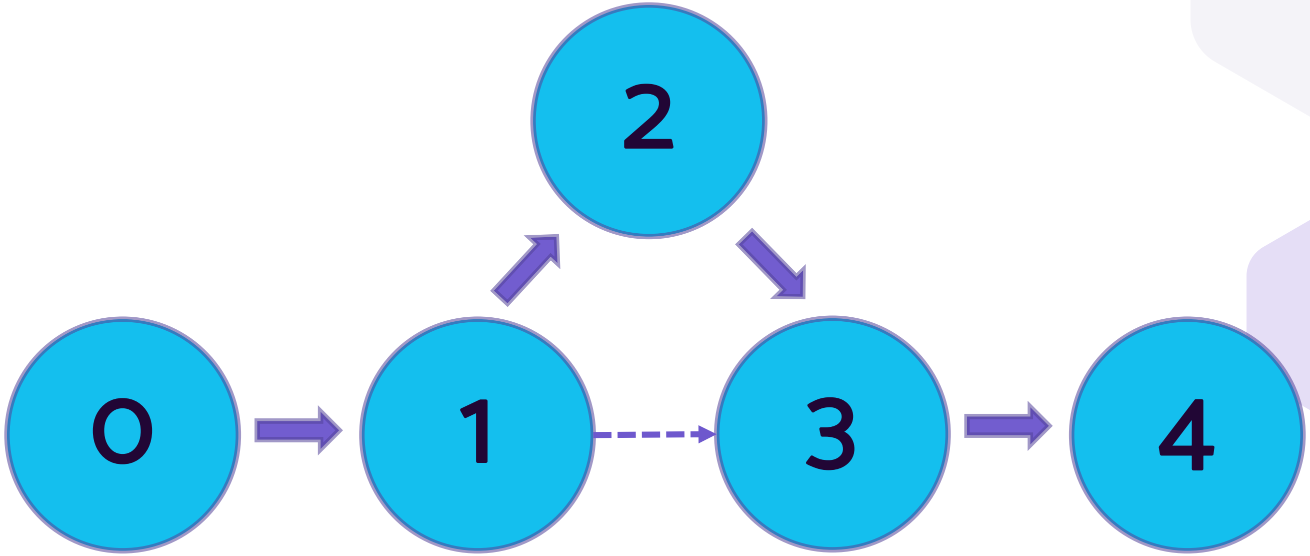
- In a traditional list, you have to shuffle all the subsequent elements around to make room for a new one.



Insert a Node



Insert a Node



Let's write the linked list inserting program

- Go to exercise 4

Remove Duplicate Node Challenge

- Complete the ‘remove_duplicates() so that it removes any duplicates nodes

Input: 2 -> 2 -> 3 -> 3 -> 3 -> 4 -> 5

Result: 2 -> 3 -> 4 -> 5

Kaung Sein Technology

Remove Duplicate Node Challenge

Go to exercise challenge1_solution.py for solution

Quiz

- 2. Which cases do you need to consider for node insertion?
  - Just a single case
  - Empty list, replacing head node, any subsequent position
  - Empty list, any subsequent position
  - Replacing the head node, any subsequent position

Quiz

- 3. What happens to a Node that is no longer referenced by any other Node?
  - It will stay in memory until you restart your computer
  - It stays in memory waiting for a node to link to it again
  - The Python garbage collector removes it from memory
  - It stays indefinitely

Quiz Answers

2. Empty list, replacing head node, any subsequent position
3. The Python garbage collector removes it from memory

Kaung Sein Technology

THANKS!

You can find me at:

- https://www.facebook.com/mgsein123
- aunghein116688@gmail.com
- 095120779

Kaung Sein Technology