Welcome everyone!

- You can find these slides on GitHub here: https://github.com/WomenWhoCode/WWCodePython
- Please make sure your chat is set to “All panelists and attendees”.
- Some housekeeping rules:
  - Everyone will be muted throughout the webinar, but there will be opportunities for participation!
  - Please share your thoughts on the chat and/or ask questions in the Q&A.
  - The entire team is here today. Please reach out to us with any technical questions!
-

WELCOME

WOMEN WHO CODE

Women Who Code Python

Intro to Data Structures with Python: Ace the Technical Interview

Session #4: Linked Lists

WOMEN WHO CODE®

MEET YOUR TEAM

Rishika Singh
Track Lead

Jasmeen Rajpal
Evangelist

WOMEN WHO CODE

OUR MISSION

Inspiring women to excel in technology careers.

WOMEN WHO CODE

OUR VISION

A world where women are representative as technical executives, founders, VCs, board members and software engineers.

WOMEN WHO CODE

OUR TARGET

Engineers with two or more years of experience looking for support and resources to strengthen their influence and levelup in their careers.

WOMEN WHO CODE

CODE OF CONDUCT

WWCode is an inclusive community, dedicated to providing an empowering experience for everyone who participates in or supports our community, regardless of gender, gender identity and expression, sexual orientation, ability, physical appearance, body size, race, ethnicity, age, religion, socioeconomic status, caste, creed, political affiliation, or preferred programming language(s).

Our events are intended to inspire women to excel in technology careers, and anyone who is there for this purpose is welcome. We do not tolerate harassment of members in any form. Our **Code of Conduct** applies to all WWCode events and online communities.

Read the full version and access our incident report form at **womenwhocode.com/codeofconduct**

WOMEN WHO CODE

230,000
Members

70 networks in 20 countries
Members in 97+ countries
10K+ events
$1025 daily Conference tickets
$2M Scholarships
Access to jobs + resources
Infinite connections

WOMEN WHO CODE

OUR MOVEMENT

As the world changes, we can be a connecting force that creates a sense of belonging while the world is being asked to isolate.

WOMEN WHO CODE

Upcoming Events

SAT 01 MAY

🔥 **Introduction to Deep Learning for Edge Devices Session 6: Early Exits in Neural networks** 🔥 *Featured*

Online | Data Science | 5:00 PM – 6:30 PM PDT (UTC-0700)

Register

THU 06 MAY

✨ **Intro to Data Structures with Python: Ace the Technical Interview (Live Q&A / Review Session)** ✨ *Featured*

Online | Python | 5:00 PM – 6:30 PM PDT (UTC-0700)

Register

SAT 15 MAY

🔥 **Introduction to Deep Learning for Edge Devices: Session 7: Edge TPU and Accelerators**🔥 *Featured*

Online | Data Science | 5:00 PM – 6:30 PM PDT (UTC-0700)

Register

THU 20 MAY

✨ **Intro to Data Structures with Python: Ace the Technical Interview (Session #6: Trees)**✨ *Featured*

Online | Python | 5:00 PM – 6:30 PM PDT (UTC-0700)

Register

SAT 29 MAY

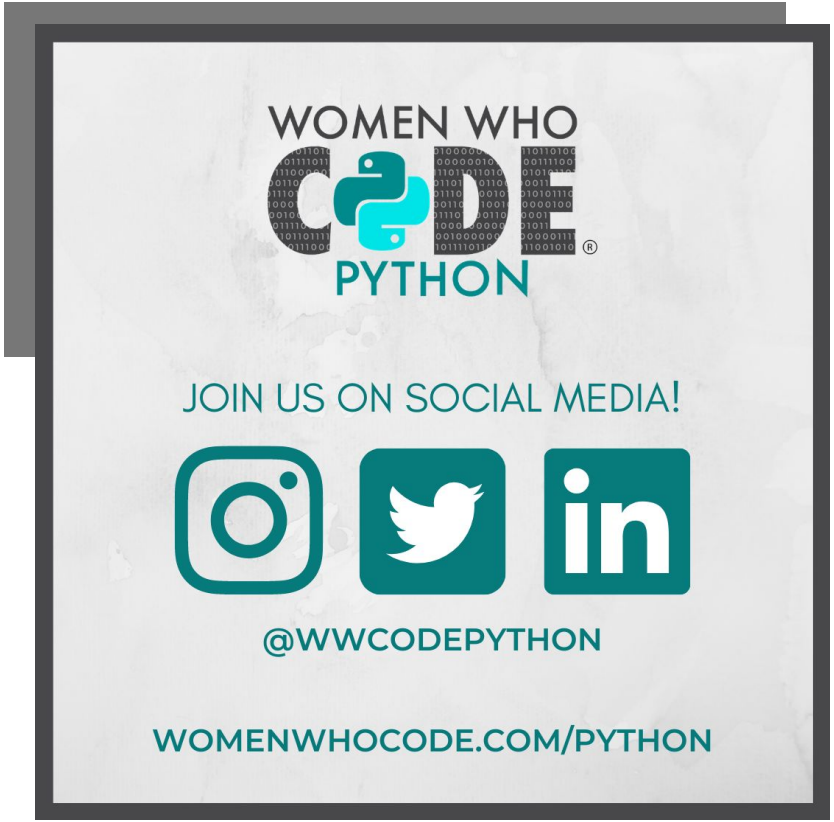
🔥 **Introduction to Deep Learning for Edge Devices: Session 8: Mini Project**🔥 *Featured*

Online | Data Science | 5:00 PM – 6:30 PM PDT (UTC-0700)

Register

WOMEN WHO CODE PYTHON

Stay Connected



Soumya Vemuri

CS Undergrad Student | WWCode Python Volunteer

Today's Agenda

1. What are Linked Lists?
2. Operations on Linked Lists
    a. Insertion
    b. Deletion
3. Why do we use Linked Lists?
4. Arrays vs Linked Lists
5. Types of Linked Lists
    a. Singly Linked Lists
    b. Doubly Linked Lists
    c. Circular Linked Lists
6. Live Coding
7. Resources & Preparation

What are Linked Lists?

Linked Lists

- Linked Lists are a type of Linear Data Structure.
- Linked Lists consist of a sequence of elements *(nodes)*, connected by **links**.
- Each **node** of the linked list consists of
  - A value (data)
  - A pointer

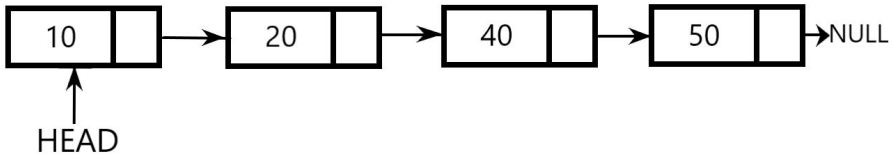
Linked Lists (contd.)

- Each linked list has a Head.
  - We access the rest of the elements through the head
- The last element of a Linked List points to NULL
- Unlike Arrays, these elements aren't stored in contiguous memory locations.

Operations on Linked Lists

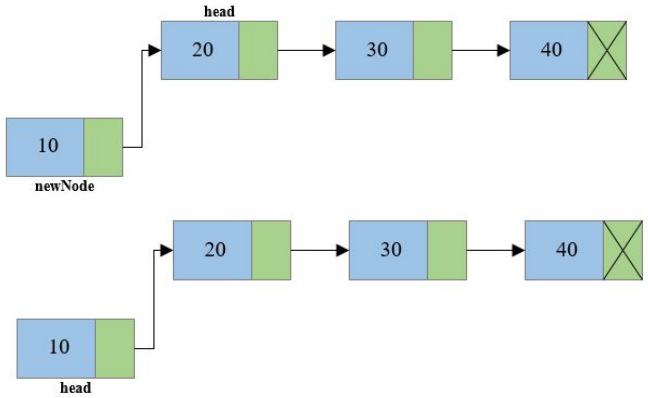
Operations on Linked Lists

- **Traversing** the linked list
- **Insert:** Inserting an element into a linked list.
- **Delete:** Deleting an element from a linked list.
- Inserting and Deleting a node from a linked list can be done at
  - the head
  - a given position (nth position, or after a value)
  - the end of a linked list

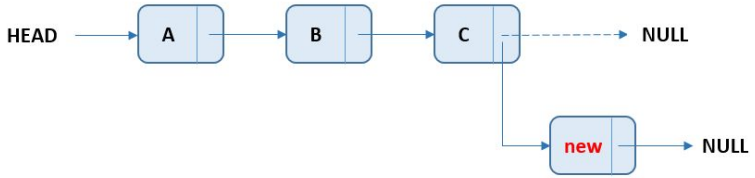


Operations on Linked Lists: Insertion

- Insertion at the Beginning of the linked list
  - Make the NEXT of the new node to point at HEAD
  - Make the New Node the HEAD

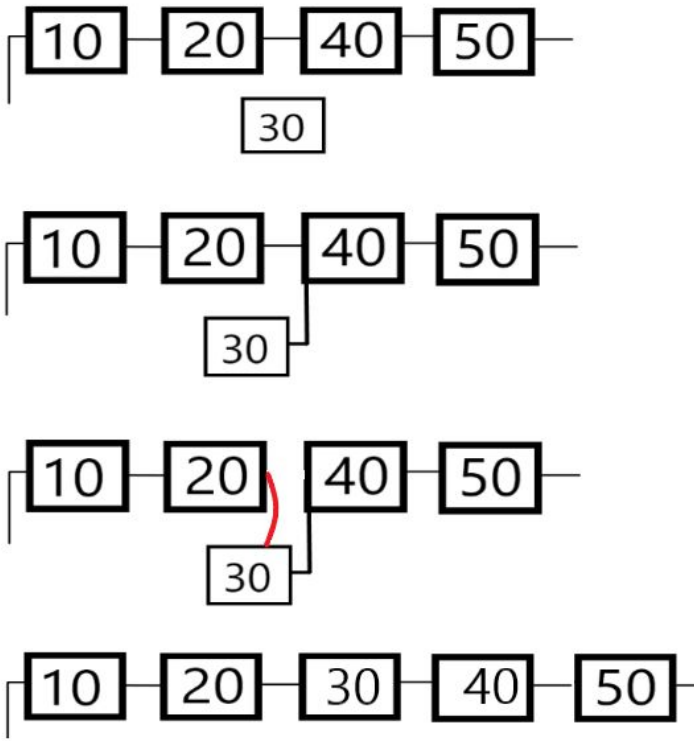


- Insertion at the End of the linked list



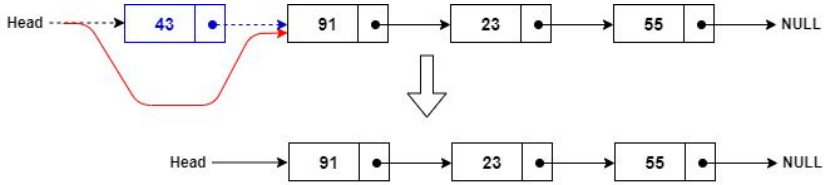
Operations on Linked Lists: Insertion

- Insertion in the Middle of a linked list
  - Traverse the list till we reach the ‘2nd’ node, or the node with ‘20’
  - Point ‘30’s next to ‘20’s next
  - Change ‘20’s next to point to ‘30’

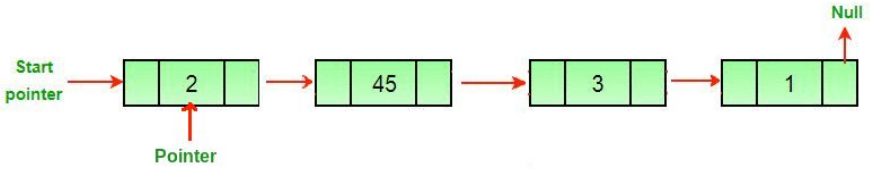


Operations on Linked Lists: Deletion

- Deletion of a node at the Beginning of the linked list
  - Traverse to the second Node of the List
  - Make the second Node the HEAD

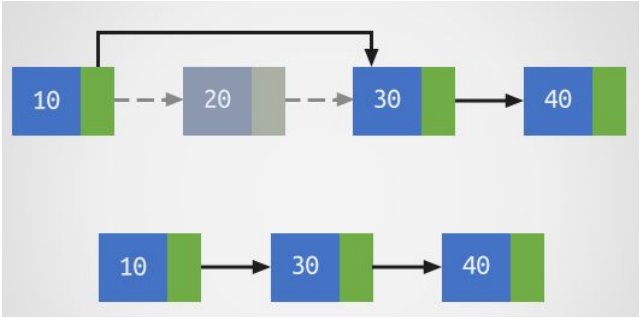


- Deletion at the End of the linked list
  - Traverse till the second-to last node
  - Point next of this node to NULL



Operations on Linked Lists: Deletion

- Deleting a node from the Middle of a linked list
  - Traverse till you reach the node that's before the one you want to delete
  - Change the next of this node to the 'next' of the next node



Why use linked lists?

Advantages of Linked Lists

- The main advantage of linked lists is that it is a Dynamic Data Structure.
  - Its size can grow or shrink during run time.

- As we don't need to declare its size before running the code, no additional memory space is wasted.

Disadvantages of Linked Lists

- Linked lists require an extra memory location for each value, to store a pointer to the next element.

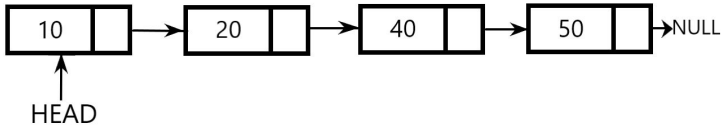
- To access an element in a linked list, we have to traverse all the elements that come before it.

- In most linked lists, travelling to the previous element isn't possible.

Arrays vs Linked Lists

Arrays vs Linked Lists

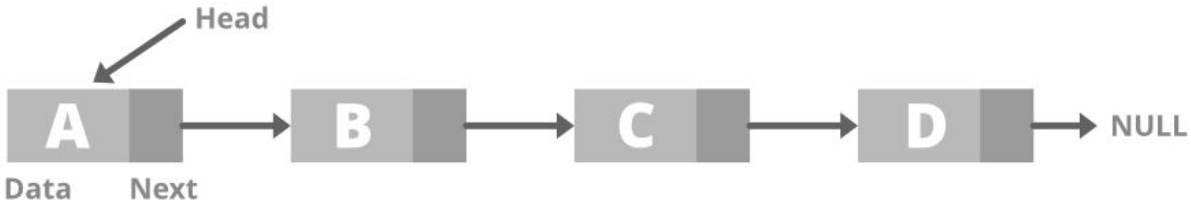
Action	Arrays	Linked Lists
Accessing an element	Constant Time. (Random Access)	We need to traverse all the elements that come before it. (Sequential Access)
Inserting/Deleting an element at the beginning	We need to shift all the elements in the array (Linear Time).	Constant Time
Inserting/Deleting an element at the end	Constant Time	We need to traverse all the elements of the linked list to access the last element (Linear Time).
Memory location of the elements	Elements are present in contiguous memory locations. The memory is allocated at compile time	The elements are linked via pointers, and may be placed in different memory locations. The memory is allocated during run time.



Types of Linked Lists

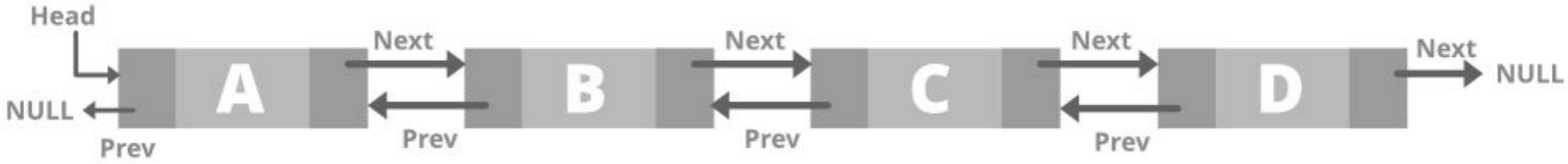
Singly (Simple) Linked Lists

- Each **node** in this type of linked list consists of a data value, and a pointer to the next node.
- In this type of Linked List, traversal is possible only in one direction (from head to the tail).



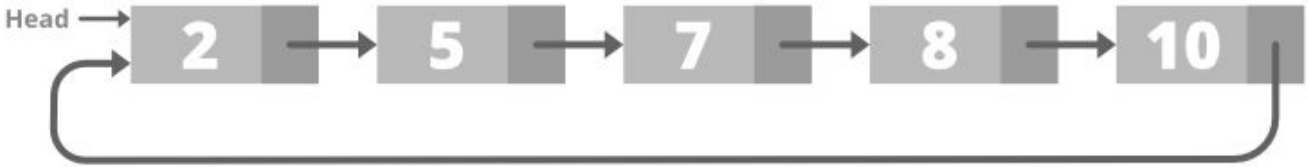
Doubly (Simple) Linked Lists

- Each **node** in this type of linked list consists of a data value, and two pointers - one to the ***next*** node, and one to the ***previous*** node
- In this type of Linked List, we can travel to the next node as well as the previous.
- Both the Head and the 'Tail' of the linked list point to NULL

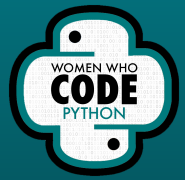


Circular Linked Lists

- Each **node** in this type of linked list consists of a data value, and a pointer to the next node.
- In this type of Linked List, the last node **points to the Head**.



Q&A Time!



Time for Live Coding!

- https://colab.research.google.com/drive/1L7Oj0JcukuizjhirC35vT4Uz0YSbP5B-?usp=sharing

Next Session!

tinyurl.com/ds-series-form

INTRO TO

DATA STRUCTURES WITH PYTHON

ACE THE TECHNICAL INTERVIEW

THU. MAY 6TH
@ 8:00PM EDT

WOMEN WHO CODE PYTHON

Questions?

Join our Slack channel:
#intro-data-structures-stdy-grp

Thank You!