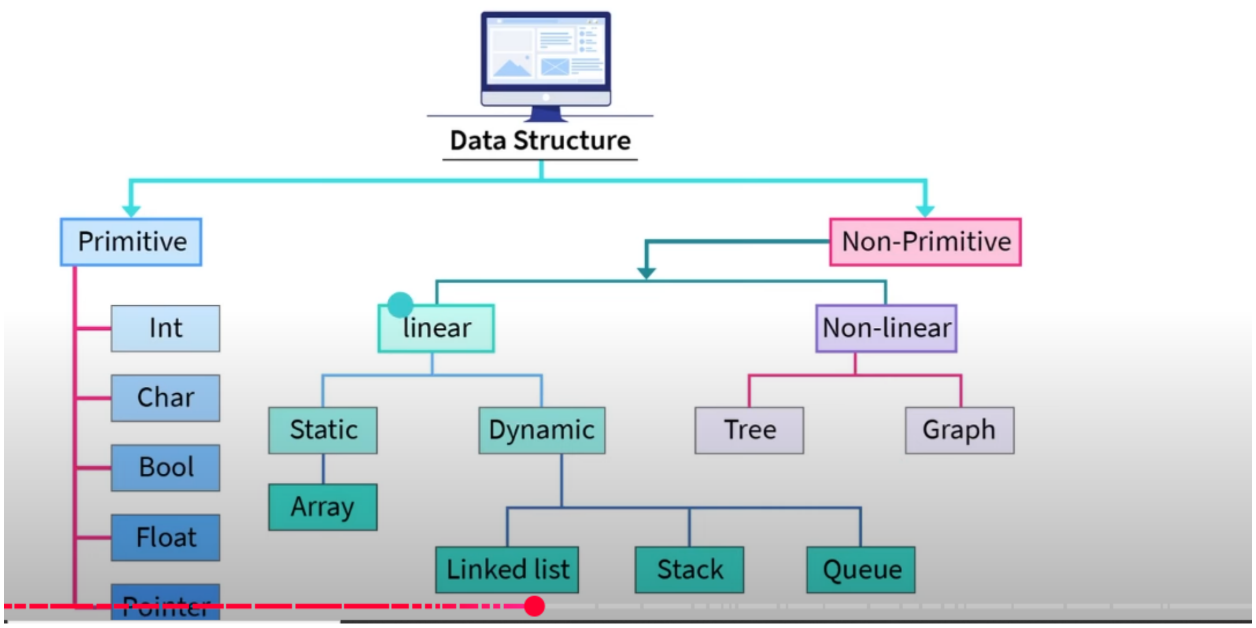
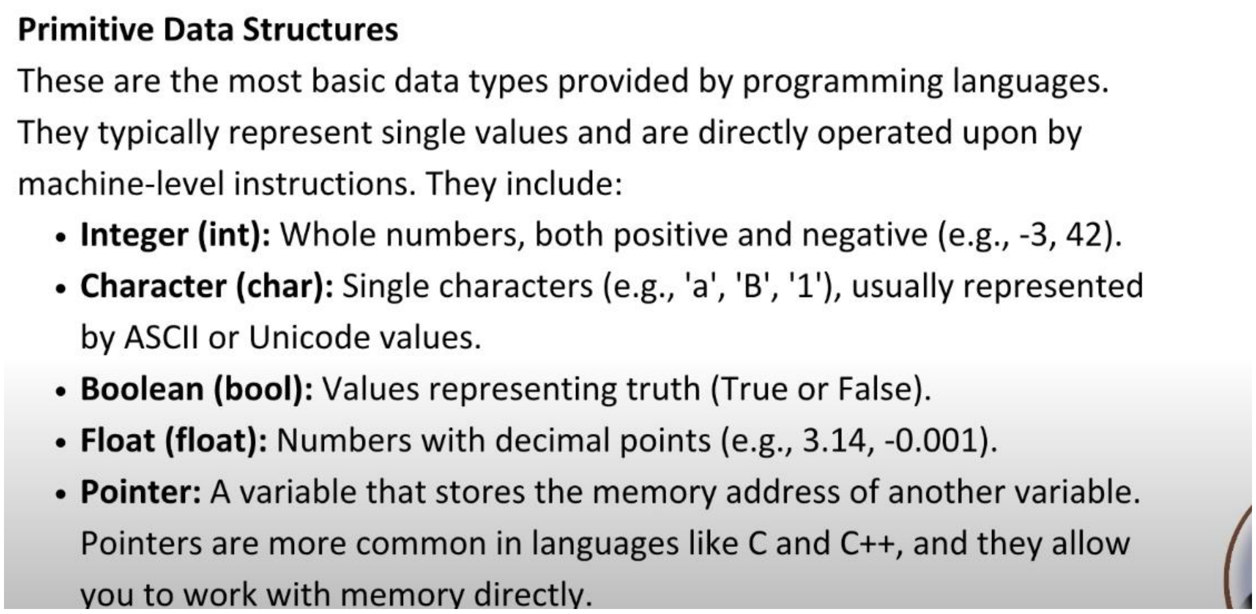
Data Structures in Python

In Python, **data structures** are used to store and organize data efficiently, making it easy to perform operations like accessing, modifying, and manipulating the data.

- Python data structures are the **building blocks of any robust and efficient programming solution**.
- They play a crucial role in organizing and managing data, making our code more organized, optimized, and scalable.

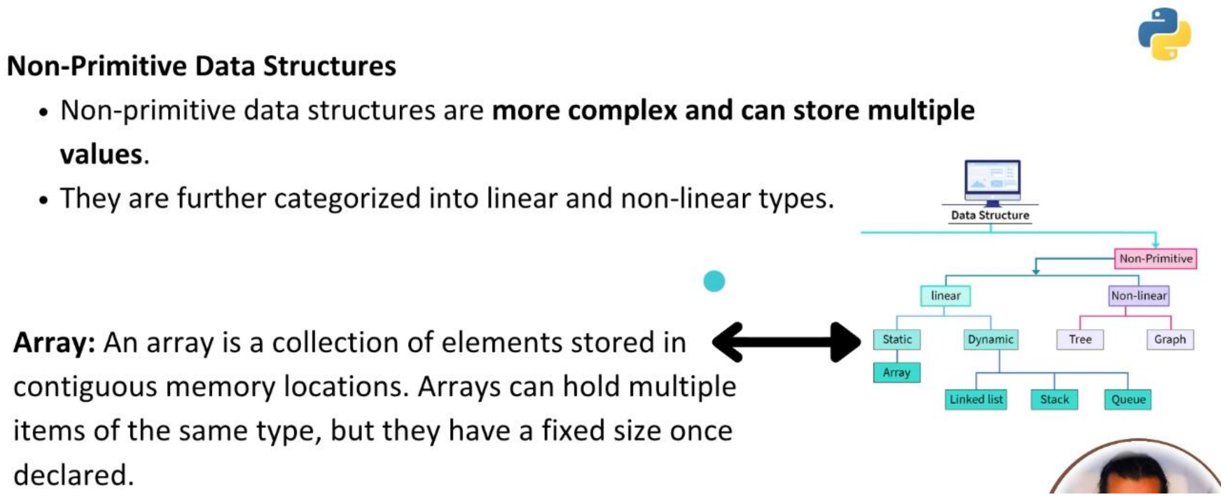




Primitive Data Structures

These are the most basic data types provided by programming languages. They typically represent single values and are directly operated upon by machine-level instructions. They include:

- **Integer (int):** Whole numbers, both positive and negative (e.g., -3, 42).
- **Character (char):** Single characters (e.g., 'a', 'B', '1'), usually represented by ASCII or Unicode values.
- **Boolean (bool):** Values representing truth (True or False).
- **Float (float):** Numbers with decimal points (e.g., 3.14, -0.001).
- **Pointer:** A variable that stores the memory address of another variable. Pointers are more common in languages like C and C++, and they allow you to work with memory directly.

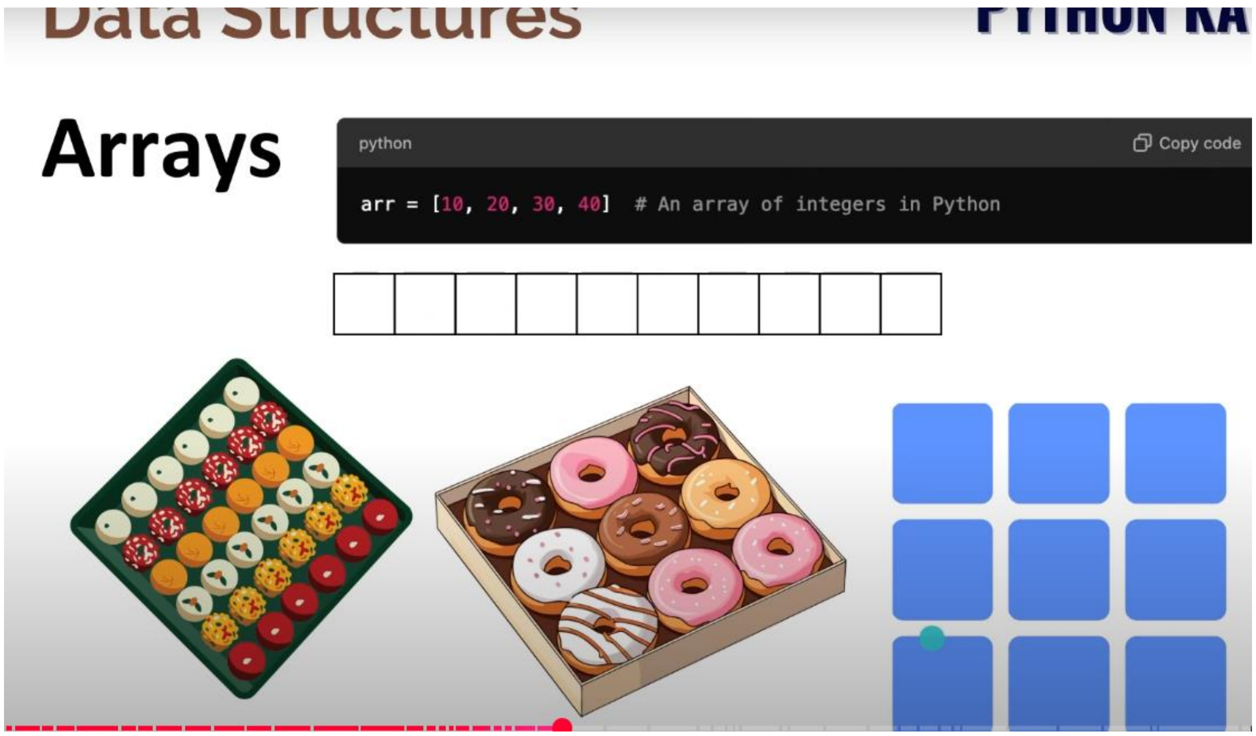


Non-Primitive Data Structures

- Non-primitive data structures are **more complex and can store multiple values**.
- They are further categorized into linear and non-linear types.

Array: An array is a collection of elements stored in contiguous memory locations. Arrays can hold multiple items of the same type, but they have a fixed size once declared.

Arrany can be 1D 2D 3D



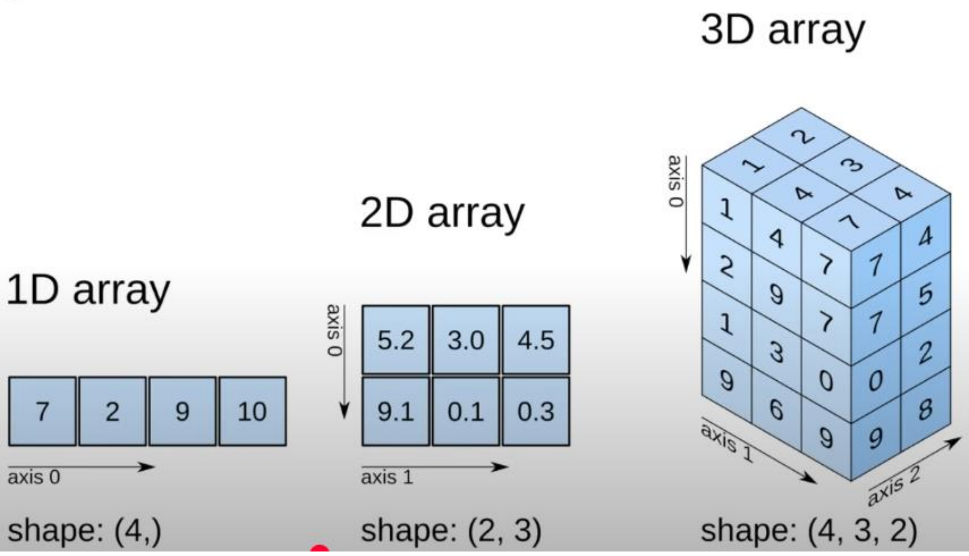
Data Structures PYTHON KA

Arrays

```python
arr = [10, 20, 30, 40]  # An array of integers in Python
```

Arrays

Example



Lists:
Lists are like containers that hold multiple items in Python. You can put anything you want inside a list, such as numbers, words, or even other lists.
Lists are ordered, so you can keep related data together and access it easily.

python

```python
names = ["Ali", "Ayesha", "Sara"]
print(names)
```

Dictionaries:
Dictionaries work like real-life dictionaries, where words are connected to their meanings.
In Python, a dictionary associates keys with values. This allows you to quickly find a value using its corresponding key.

```
dictionary = {
    "Apple": "A red fruit",
    "Car": "A vehicle with wheels",
    "Book": "Something you read"
}
```

Here:

- `"Apple"` is the **key**
- `"A red fruit"` is the **value**

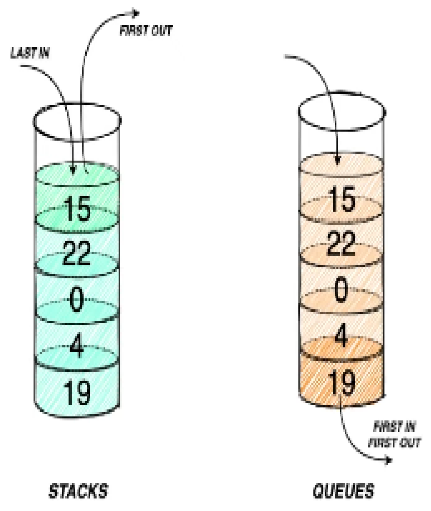
Stacks:

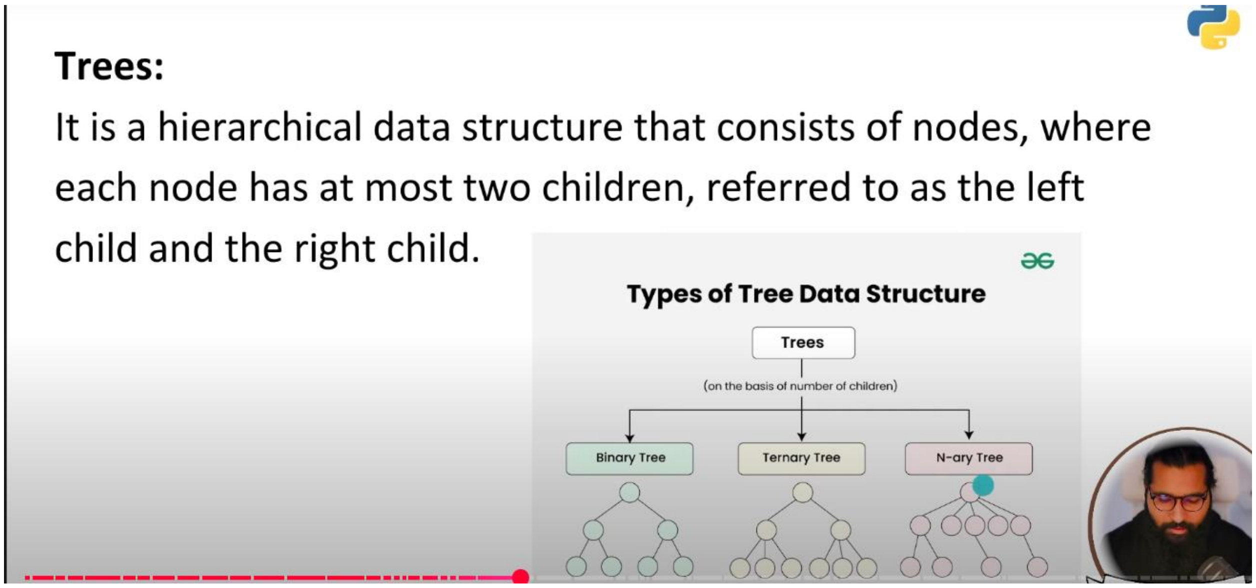
Imagine a stack of plates; you can only add or remove plates from the top. In programming, a stack follows the Last-In-First-Out (LIFO) principle.

The last item added is the first one to be removed.

QUEUES:

A queue is a linear data structure that follows the **First In, First Out (FIFO)** principle. The first item added is the first one to be removed. Queues are used in scenarios like managing tasks in a print queue or scheduling processes in an operating system.

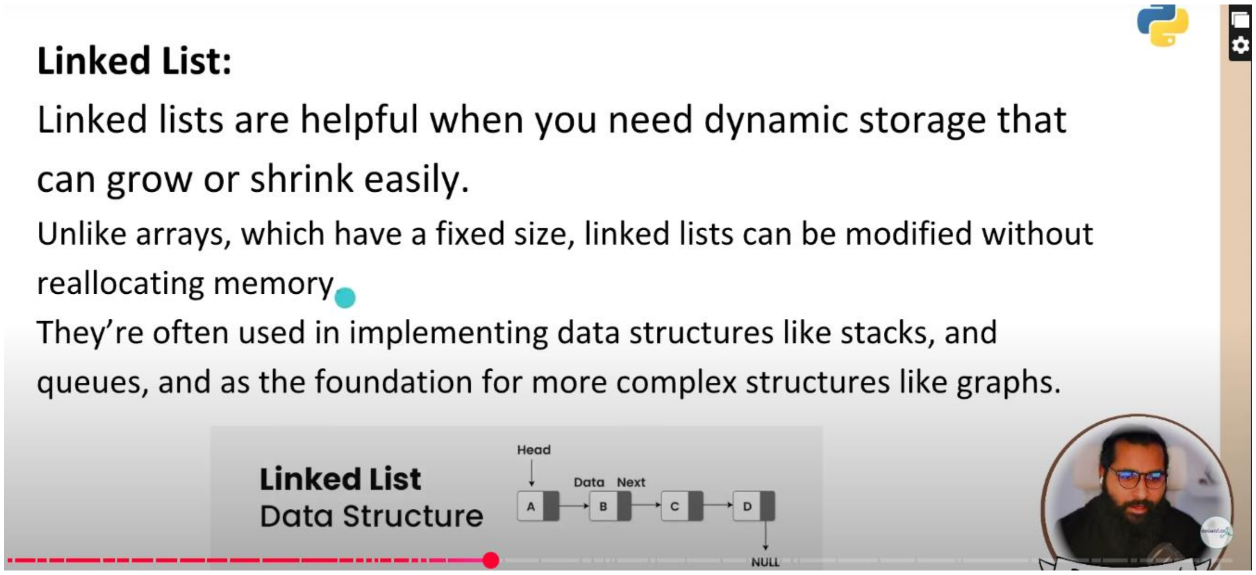




Trees:

It is a hierarchical data structure that consists of nodes, where each node has at most two children, referred to as the left child and the right child.

Types of Tree Data Structure
Trees
(on the basis of number of children)
Binary Tree
Ternary Tree
N-ary Tree



Linked List:

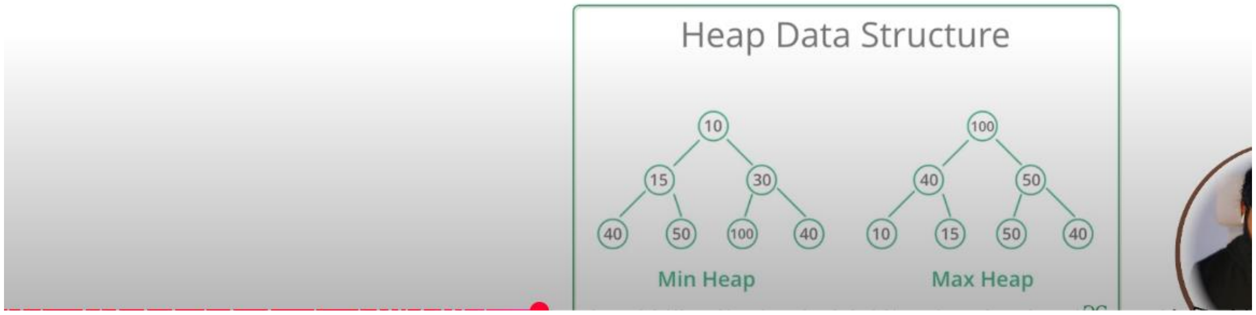
Linked lists are helpful when you need dynamic storage that can grow or shrink easily.

Unlike arrays, which have a fixed size, linked lists can be modified without reallocating memory.

They're often used in implementing data structures like stacks, and queues, and as the foundation for more complex structures like graphs.

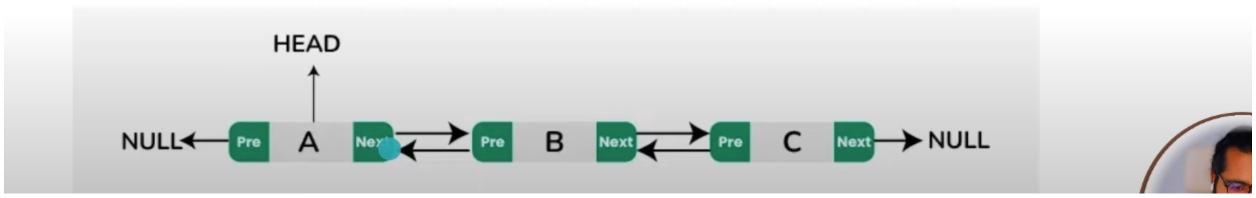
Heapq:

Heapq is like a special type of list where the elements are arranged in a particular order that makes it easy to find the smallest (or largest) element quickly.



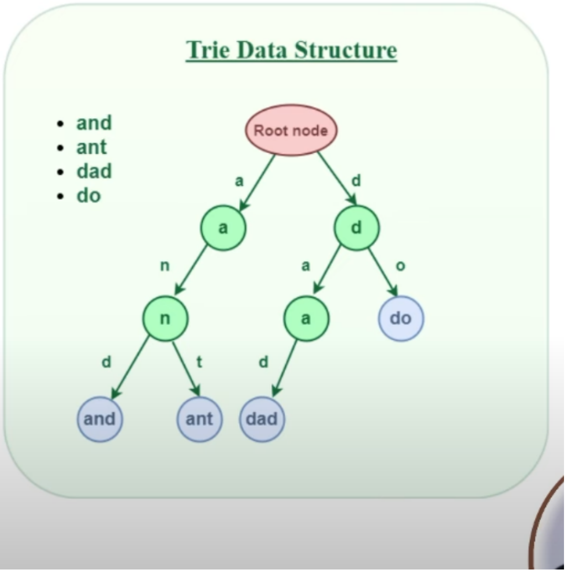
Doubly Linked List:

A doubly linked list is similar to a linked list, but each node has a reference to both the next and the previous node.



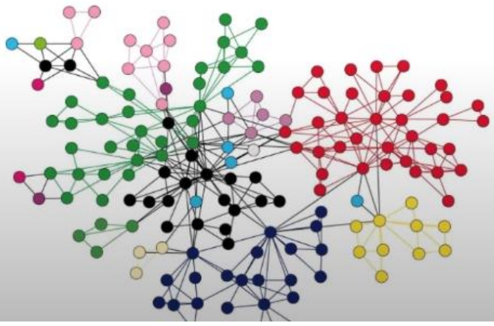
Trie (Prefix Tree):

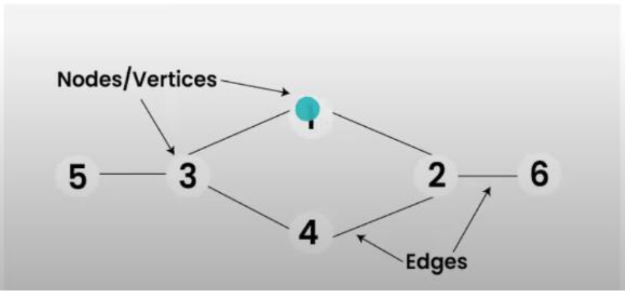
A trie is a tree-like data structure that is used to efficiently store and search a large collection of strings. It is often used for autocomplete and dictionary implementations.



Graph:

A graph is a collection of nodes (vertices) and edges that connect pairs of nodes. Graphs are widely used to represent networks or relationships between objects.




Nodes/Vertices
1
5
3
2
6
4
Edges