tPython :

Python is a high-level, interpreted, general-purpose programming language. Being a general-purpose language, it can be used to build almost any type of application with the right tools/libraries. Additionally, python supports objects, modules, threads, exception-handling, and automatic memory management which help in modeling real-world problems and building applications to solve these problems.

Benefits of using Python:

- Python is a general-purpose programming language that has a simple, easy-to-learn syntax that emphasizes readability and therefore reduces the cost of program maintenance. Moreover, the language is capable of scripting, is completely open-source, and supports third-party packages encouraging modularity and code reuse.
- Its high-level data structures, combined with dynamic typing and dynamic binding, attract a huge community of developers for Rapid Application Development and deployment.

Python is a Dynamically Typed Language. It interprets the code line by line and therefore checks for any errors during execution of the program.

What is Scope in Python?

Every object in Python functions within a scope. A scope is a block of code where an object in Python remains relevant. Namespaces uniquely identify all the objects inside a program. However, these namespaces also have a scope defined for them where you could use their objects without any prefix. A few examples of scope created during code execution in Python are as follows:

- A local scope refers to the local objects available in the current function.
- A global scope refers to the objects available throughout the code execution since their inception.

What are lists and tuples? What is the key difference between the two?

Lists and Tuples are both sequence data types that can store a collection of objects in Python. The objects stored in both sequences can have different data types. Lists are represented with square brackets ['sara', 6, 0.19], while tuples are represented with parantheses ('ansh', 5, 0.97).
But what is the real difference between the two? The key difference between the two is that while lists are mutable, tuples on the other hand are immutable objects. This means that lists can be modified, appended or sliced on the go but tuples remain constant and cannot be modified in any manner. You can run the following example on Python IDLE to confirm the difference:

What are modules and packages in Python?

Python packages and Python modules are two mechanisms that allow for modular programming in Python. Modularizing has several advantages -

- Simplicity: Working on a single module helps you focus on a relatively small portion of the problem at hand. This makes development easier and less error-prone.
- Maintainability: Modules are designed to enforce logical boundaries between different problem domains. If they are written in a manner that reduces interdependency, it is less likely that modifications in a module might impact other parts of the program.
- Reusability: Functions defined in a module can be easily reused by other parts of the application.

What are global, protected and private attributes in Python?

- Global variables are public variables that are defined in the global scope. To use the variable in the global scope inside a function, we use the global keyword.
- Protected attributes are attributes defined with an underscore prefixed to their identifier eg. _sara. They can still be accessed and modified from outside the class they are defined in but a responsible developer should refrain from doing so.
- Private attributes are attributes with double underscore prefixed to their identifier eg. __ansh. They cannot be accessed or modified from the outside directly and will result in an AttributeError if such an attempt is made

Self is used to represent the instance of the class. With this keyword, you can access the attributes and methods of the class in python. It binds the attributes with the given arguments. self is used in different places and often thought to be a keyword. But unlike in C++, self is not a keyword in Python

__init__ is a contructor method in Python and is automatically called to allocate memory when a new object/instance is created. All classes have a __init__ method associated with them. It helps in distinguishing methods and attributes of a class from local variables.

- Arrays in python can only contain elements of same data types i.e., data type of array should be homogeneous. It is a thin wrapper around C language arrays and consumes far less memory than lists.
- Lists in python can contain elements of different data types i.e., data type of lists can be heterogeneous. It has the disadvantage of consuming large memory.

How are arguments passed by value or by reference in python?

- Pass by value: Copy of the actual object is passed. Changing the value of the copy of the object will not change the value of the original object.
- Pass by reference: Reference to the actual object is passed. Changing the value of the new object will change the value of the original object.

9. What is init method in python?

The init method works similarly to the constructors in Java. The method is run as soon as an object is instantiated. It is useful for initializing any attributes or default behaviour of the object at the time of instantiation.

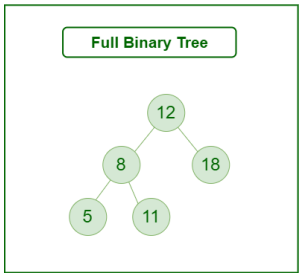
TREES:

Types of Binary Tree based on the number of children:
Following are the types of Binary Tree based on the number of children:

Full Binary Tree
Degenerate Binary Tree
Skewed Binary Trees

1. **Full Binary Tree :**

   A Binary Tree is a full binary tree if every node has 0 or 2 children. The following are examples of a full binary tree. We can also say a full binary tree is a binary tree in which all nodes except leaf nodes have two children.
   A full Binary tree is a special type of binary tree in which every parent node/internal node has either two or no children. It is also known as a proper binary tree.



2. **Degenerate Tree :**

   A tree in which every node has only one child, either left or right

3. Skewed Tree :
   A tree which is dominated either by left nodes or right nodes

   These are all the types based on child nodes

Types based on completion of levels :

Types of Binary Tree On the basis of the **completion** of levels:
Complete Binary Tree
Perfect Binary Tree
Balanced Binary Tree

1. **Complete Binary Tree :**

A Binary Tree is a Complete Binary Tree if all the levels are completely filled except possibly the last level and the last level has all keys as left as possible.

A complete binary tree is just like a full binary tree, but with two major differences:

Every level except the last level must be completely filled.
All the leaf elements must lean towards the left.
The last leaf element might not have a right sibling i.e. a complete binary tree doesn't have to be a full binary tree

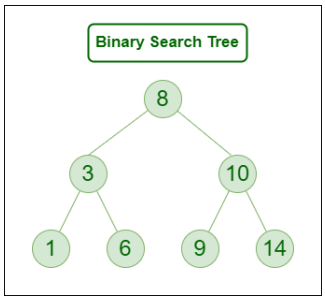
2. Perfect Binary Tree

A Binary tree is a Perfect Binary Tree in which all the internal nodes have two children and all leaf nodes are at the same level.

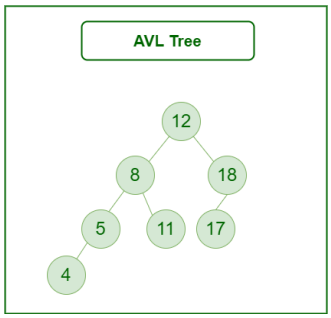
3. Balanced Binary Tree
A binary tree is balanced if the height of the tree is O(Log n) where n is the number of nodes. For Example, the AVL tree maintains O(Log n) height by making sure that the difference between the heights of the left and right subtrees is at most

1. Binary Search Tree
Binary Search Tree is a node-based binary tree data structure that has the following properties:

The left subtree of a node contains only nodes with keys lesser than the node's key.
The right subtree of a node contains only nodes with keys greater than the node's key.
The left and right subtree each must also be a binary search tree.



2. AVL Tree
AVL tree is a self-balancing Binary Search Tree (BST) where the difference between heights of left and right subtrees cannot be more than one for all nodes.



ARRAY :

An array is a collection of items of same data type stored at contiguous memory locations. Or Array is a linear data structure that is a collection of similar data types. Arrays are stored in contiguous memory locations. It is a static data structure with a fixed size. It combines data of similar types.

Operations : Insert , delete through traversal

Applications :

Financial Analysis: Arrays are used in financial analysis to store historical stock prices and other financial data. This allows for efficient data access and analysis, which is important in real-time trading systems

Signal Processing: Arrays are used in signal processing to represent a set of samples that are collected over time. This can be used in applications such as speech recognition, image processing, and radar systems.

Multimedia Applications: Arrays are used in multimedia applications such as video and audio processing, where they are used to store the pixel or audio samples. For example, an array can be used to store the RGB values of an image.

Data Mining: Arrays are used in data mining applications to represent large datasets. This allows for efficient data access and processing, which is important in real-time applications.

Arrays are used as the base of all sorting algorithms.
Arrays are used to implement other DS like a stack, queue, etc.

Advantages :
Arrays provide direct and efficient access to any element in the collection. Accessing an element in an array is an O(1) operation, meaning that the time required to access an element is constant and does not depend on the size of the array.

Disadvantage :
Fixed size: Arrays have a fixed size that is determined at the time of creation. This means that if the size of the array needs to be increased, a new array must be created and the data must be copied from the old array to the new array, which can be time-consuming and memory-intensive.

Insertion and deletion issues: Inserting or deleting an element from an array can be inefficient and time-consuming because all the elements after the insertion or deletion point must be shifted to accommodate the change.
Wasted space: If an array is not fully populated, there can be wasted space in the memory allocated for the array. This can be a concern if memory is limited.

Linked List

Linked List is a linear data structure, in which elements are not stored at a contiguous location, rather they are linked using pointers. Linked List forms a series of connected nodes, where each node stores the data and the address of the next node.

The size of memory can be allocated or de-allocated at run time based on the operation that is being performed such as insertion or deletion.

Ease of Insertion/Deletion: The insertion and deletion of elements are simpler than arrays since no elements need to be shifted after insertion and deletion, Just the address needed to be updated.

As we know Linked List is a dynamic data structure the size increases or decreases as per the requirement so this avoids the wastage of memory.

There are mainly three types of linked lists:

Single-linked list
Double linked list
Circular linked list

Operations on Linked Lists
Insertion: Adding a new node to a linked list involves adjusting the pointers of the existing nodes to maintain the proper sequence. Insertion can be performed at the beginning, end, or any position within the list
Deletion: Removing a node from a linked list requires adjusting the pointers of the neighboring nodes to bridge the gap left by the deleted node. Deletion can be performed at the beginning, end, or any position within the list.
Searching: Searching for a specific value in a linked list involves traversing the list from the head node until the value is found or the end of the list is reached.

Advantages of Linked Lists
Dynamic Size: Linked lists can grow or shrink dynamically, as memory allocation is done at runtime.
Insertion and Deletion: Adding or removing elements from a linked list is efficient, especially for large lists.
Flexibility: Linked lists can be easily reorganized and modified without requiring a contiguous block of memory.

Disadvantages of Linked Lists
Random Access: Unlike arrays, linked lists do not allow direct access to elements by index. Traversal is required to reach a specific node.
Extra Memory: Linked lists require additional memory for storing the pointers, compared to arrays.

Linked lists are versatile data structures that provide dynamic memory allocation and efficient insertion and deletion operations. Understanding the basics of linked lists is essential for any programmer or computer science enthusiast. With this knowledge, you can implement linked lists to solve various problems and expand your understanding of data structures and algorithms.

1. Singly Linked List

A singly linked list is a special type of linked list in which each node has only one link that points to the next node in the linked list.

Each node holds a single value and a reference to the next node in the list.

The list has a head, which is a reference to the first node in the list, and a tail, which is a reference to the last node in the list.

The nodes are not stored in a contiguous block of memory, but instead, each node holds the address of the next node in the list.

Accessing elements in a singly linked list requires traversing the list from the head to the desired node, as there is no direct access to a specific node in memory.

App : Operating systems: Singly linked lists are used in operating systems for tasks such as scheduling processes and managing system resources.

2. **Doubly Linked List**

A doubly linked list is a special type of linked list in which each node contains a pointer to the previous node as well as the next node in the structure.\

Dynamic size: The size of a doubly linked list can change dynamically, meaning that nodes can be added or removed as needed.
Two-way navigation: In a doubly linked list, each node contains pointers to both the previous and next elements, allowing for navigation in both forward and backward directions.
Memory overhead: Each node in a doubly linked list requires memory for two pointers (previous and next), in addition to the memory required for the data stored in the node.

App :
Implementing a Hash Table: Doubly linked lists can be used to implement hash tables, which are used to store and retrieve data efficiently based on a key.

Disadv : Requires more memory due to 2 pointers apart from data stored
More code coz more manipulation of pointers may give rise to bugs when compared to arrays

3. **Circular Linked List**

A circular linked list is a special type of linked list in which the last node is connected to the first node, creating a continuous loop. In a circular linked list, each node has a reference to the next node in the sequence, just like in a regular linked list, but the last node's reference points back to the first node.

The characteristics of a circular linked list are:

The last node in the list points back to the first node.

Unlike a regular linked list, which ends with a null reference, a circular linked list has no end, as the last node points back to the first node.

Circular linked lists can grow or shrink dynamically as elements are added or removed.

App :

Resource allocation: In operating systems, circular linked lists are used to implement round-robin scheduling, where processes are given equal time slices in a cyclic fashion.

Music and video playback: Circular linked lists can be used to implement continuous playback of songs or videos.

ADV
Applications in various fields: Circular linked lists have various applications in fields such as operating systems, game development, music and video playback, buffer management, and LRU Cache, among others.

TOR :

A Linked List is a linear data structure that is used to store a collection of data with the help of nodes. A linked list is made up of two items that are data and a reference to the next node. A reference to the next node is given with the help of pointers and data is the value of a node. Each node contains data and links to the other nodes. It is an ordered collection of data elements called a node and the linear order is maintained by pointers. It has an upper hand over the array as the number of nodes i.e. the size of the linked list is not fixed and can grow and shrink as and when required, unlike arrays. Some of the features of the linked list are as follows:
The consecutive elements are connected by pointers.
The size of a linked list is not fixed.
The last node of the linked list points to null.
Memory is not wasted but extra memory is consumed as it also uses pointers to keep track of the next successive node.
The entry point of a linked list is known as the head.

3 type : singly , double and circular

Mostly used for insertion and deletion of data as memory can dynamically grow or shrink and time comp less than that of an array ie Efficient Insertion and Deletion: Inserting or deleting elements in a linked list is fast and efficient, as you only need to modify the reference of the next node, which is an O(1) operation.

Graph :

A Graph is a non-linear data structure consisting of vertices and edges

Graph data structures are a powerful tool for representing and analyzing complex relationships between objects or entities.

They are particularly useful in fields such as social network analysis, recommendation systems, and computer networks.

Eg :

Imagine a game of football as a web of connections, where players are the nodes and their interactions on the field are the edges. This web of connections is exactly what a graph data structure represents, and it's the key to unlocking insights into team performance and player dynamics in sports

Vertices are the fundamental units of the graph
Edges are drawn or used to connect two nodes of the graph.

1. Null Graph :

   A graph is known as a null graph if the nodes are not connected by edges.

2. Trivial Graph :

   A graph which has only one vertex or a node is known as a trivial graph.

3. Undirected graph :
   a. A graph where the nodes are connected by edges but the edges do not have any direction or is undirected towards a node.

4. Directed graph : Nodes are considered to be in pairs since edges connecting them are in one direction.

5. Connected graph : The graph in which we can traverse to any node from any given node. There exists atleast one connection between all the nodes
6. Disconnected graph : The graph in which atleast one node is not reachable from any one node is known as a disc graph.

7. Complete graph : When there is an edge present between all the nodes, the graph is said to be complete.

8. Cycle graph : The graph itself is a cycle
9. Cyclic graph : There exists atleast one cycle in the graph

10. Directed Acyclic graph : A directed graph in which cycles don't exist.

Basic Operations on Graphs

Insertion of Nodes/Edges in the graph – Insert a node into the graph.
Deletion of Nodes/Edges in the graph – Delete a node from the graph.
Searching on Graphs – Search an entity in the graph.
Traversal of Graphs – Traversing all the nodes in the graph

Social media network friends

Neural Networks: Vertices represent neurons and edges represent the synapses between them. Neural networks are used to understand how our brain works and how connections change when we learn.

When to use Graphs:
When you need to represent and analyze the relationships between different objects or entities.
When you need to perform network analysis.
When you need to identify key players, influencers or bottlenecks in a system.
When you need to make predictions or recommendations.

Disadv:

1. Creating and manipulating graphs can be computationally expensive, especially for very large or complex graphs.
2. Graph algorithms can be difficult to design and implement correctly, and can be prone to bugs and errors.