1. INTRODUCTION TO DATA STRUCTURES..

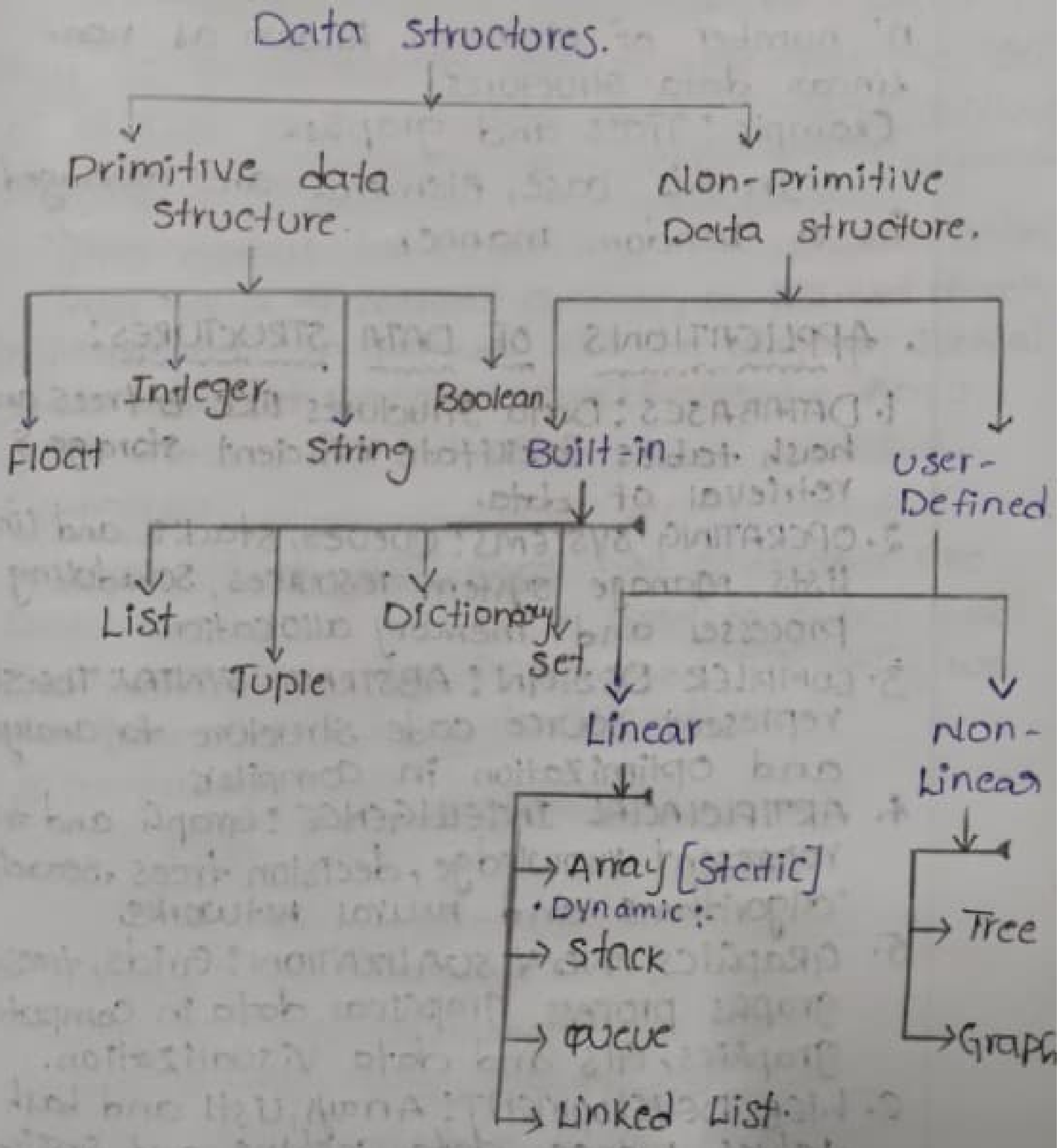
· What is Data Structure :

Data Structure is a way to store and organize data, so that it can used efficiently.

AS per name indicates itself that organizing the data in memory.

The data structure is not any programming Language like C, C++, Java, python etc,

It is a set of ALGORITHMS that we can use in any programming language to Structure data in memory.



· CLASSIFICATION OF DATA STRUCTURES.

· LINEAR DATA STRUCTURE:

The arrangement of data in the sequenti manner is known as Linear data Structure.

The data structure used for this purpose are: Arrays, stacks, queues, and Linked Lists.

In this data structures, one element is connected to only one another element in a linear form.

· NON-LINEAR DATA STRUCTURE:

When one element is connected to the 'n' number of elements known as non-linear data structures.

Example: Trees and graphs.

In this case, elements are arranged in a random manner.

· APPLICATIONS OF DATA STRUCTURES:

1. DATABASES: Data structures like B-trees and hash tables facilitate efficient storage & retrieval of data.
2. OPERATING SYSTEMS: Queues, stacks and linke lists manage system resources, scheduling processes and memory allocation.
3. COMPILER DESIGN: ABSTRACT SYNTAX TREES(A represent source code structure for analysis and optimization in compilers.
4. ARTIFICIACIAL INTELLIGENCE: Graph and tres represent knowledge, decision trees, search algorithms and neural networks.
5. GRAPHICS AND VISUALIZATION: Grids, trees an graphs process graphical data in computer graphics, GIS and data visualization.
6. WEB DEVELOPMENT: Arrays, lists and hash tables manage data, caching, and session management in web applications.

These are just a few examples, and data structures find applications in almost every area of computer science & beyond.

• ALGORITHMS AND ABSTRACT DATA TYPE [ADT].

$$\begin{tikzcd}
\boxed{\text{ALGORITHMS.}} \arrow[d] \\
\boxed{\text{ABSTRACT DATA TYPES.}} \arrow[d] \\
\boxed{\text{SET OF RULES}}
\end{tikzcd}$$

An Abstract data type [ADT] is a data type that is organized in such a way that the "set of ~~et~~ data and operations on that data without specifying how those operations are implemente

This allows for flexibility and Encapsulation.
ADT is a theoretical concept, so it's not directly implemented in python like in languages such as Java. However, we ~~can't~~ can implement ADT's using classes in PYTHON.

EXAMPLE:

A stack is a ADT that has operations like push, pop & peek. you don't need to know how those operations are implemented internally, we can just know how to use them.

program:

```
class Stack:
        def __init__(self):
                self.items = [ ].
        def is_empty(self):
                return len(self.items) == 0.
        def push(self, item):
                self.items.append (item).
        def pop(self):
                if not self.is_empty():
                return self.items.pop()
        else:
            raise ~~IndexError~~ ("pop ~~from empty stack~~

    def peek(self):
        if not self.is_empty():
            return self.items [-1]
        else:
            raise IndexError("peek from empty
                                    stack").

    def size (self):
        return len(self.items).

Stack = stack()
stack.push(1)
stack.push(2)
stack.push(3).
print("stack size:", stack.size())
print(" Top of the stack:", stack.peek()).
print ("popping:", stack.pop()).
print("stack size after popping:",
                 stack.size()).
```

OUTPUT:

```
Stack size        : 3
Top of the stack : 3
popping           : 3
Stack size after popping : 2
```

- ADT Gives the blueprint of data.
- ADT tells what is to be done and data structure tells how is to be done.

What is need of DATA STRUCTURES:

AS applications are getting complexed and amoun of data is increasing day by day, there may arise following problems:

1. PROCESSOR SPEED: AS data is growing day by day to the billions of files per entity, processer may fail to deal with that amount of data.
2. DATA STRUCTURE: consider an inventory size of 106 items to store, if our application needs to transverse 106 items every time, resolts in slowing down process.
3. MULTIPLE REQUESTS: If thousands of users are searching data simultaneously on a web server, then there are chances that to be failed to search during that process.

To solve this problems, data structures are used. Data is organized to form a data structure in a way such that all items are not required to be searched and require data can be searched instantly.

ADVANTAGES OF DATA STRUCTURES:

1. EFFICIENCY:

If the choice of a data structure for implementing a particular ADT is proper, it makes program very efficient in terms of time & space.

2. REUSABILITY:

The data structure provides reusability means that multiple client programs can use the data structure.

3. ABSTRACTION:

The data structure specified by the ADT also provides level of abstraction. The client cannot see interval working of data structure, so it does not have to worry about implementation.

4. ENCAPSULATION:

Data structures encapsulate data and related operations, promoting encapsulation and inf. hiding principles, which enhance code security & maintainabity

·Operations on Data Structures :

Data structures are building blocks in computer science that allow efficient organization, storage and manipulation of data. common operations performed are:

1. INSERTION :
   ADding new elements to the data structure.
2. DELETION :
   Removing elements from the data structure.
3. TRAVERSAL :
   Visiting each element in the data structure, usialy in a specified order (e.g, in-order, pre-order, post-order for trees).
4. SEARCH :
   Finding a specific element within the data structure.
5. SORTING :
   Arranging the elements in a specific order, such as ascending or descending.
6. MERGING :
   Combining two data structure into two or more smaller ones.
7. UPDATE :
   Modifying existing elements withen the data structure.
8. CONCATENATION :
   combining multiple data structures into one larger structure.
9. INTERSECTION :
   finding common elements between two data Structures.
10. UNION :
    Combining elements from two data Structures without duplicates.