Data Structures and Algorithms Assignment 1

1. **Describe Python's built-in data structure?**

   Python has four basic inbuilt data structures namely Lists, Dictionary, Tuple and Set. These almost cover 80% of the our real world data structures. This article will cover the above mentioned topics.

   Above mentioned topics are divided into four sections below.

   Lists: Lists in Python are one of the most versatile collection object types available. The other two types are dictionaries and tuples, but they are really more like variations of lists.

   Python lists do the work of most of the collection data structures found in other languages and since they are built-in, you don't have to worry about manually creating them.
   Lists can be used for any type of object, from numbers and strings to more lists.
   They are accessed just like strings (e.g. slicing and concatenation) so they are simple to use and they're variable length, i.e. they grow and shrink automatically as they're used.
   In reality, Python lists are C arrays inside the Python interpreter and act just like an array of pointers.
   Dictionary: In python, dictionary is similar to hash or maps in other languages. It consists of key value pairs. The value can be accessed by unique key in the dictionary.

   Keys are unique & immutable objects.
   Syntax:

   dictionary = {"key name": value}
   Tuple : Python tuples work exactly like Python lists except they are immutable, i.e. they can't be
   changed in place. They are normally written inside parentheses to distinguish them from lists (which use square brackets), but as you'll see, parentheses aren't always necessary. Since tuples are immutable, their length is fixed. To grow or shrink a tuple, a new tuple must be created.
   Sets: Unordered collection of unique objects.

   Set operations such as union (|) , intersection(&), difference(-) can be applied on a set.
   Frozen Sets are immutable i.e once created further data can't be added to them
   {} are used to represent a set. Objects placed inside these brackets would be treated as a set.

2. **Describe the Python user data structure?**

   A data structure allows data to be added, removed, stored and maintained in a structured manner. Python supports two types of data structures:

   Non-primitive data types: Python has list, set, and dictionary as its non-primitive data types which can also be considered its in-built data structures.

User-defined data structures: Data structures that aren't supported by python but can be programmed to reflect the same functionality using concepts supported by python are user-defined data structures. There are many data structure that can be implemented this way:
Linked list
Stack
Queue
Tree
Graph
Hashmap

3. Describe the stages involved in writing an algorithm?

Step – 1 : Obtain detailed information on the problem.
It is a very important step in writing an algorithm. Before starting with an algorithm the programmer must obtain maximum information about the problem that needs to be solved.

This step will help the programmer to get a better understanding of the problem which will surely prove to be useful while solving the problem.

Step – 2 : Analyze the problem.
Proper Analysis of the problem must be done including the data that must be gained, processed and retrieved for generating a valid output.

This step helps the programmer with various processes that must be attained while generating the output along with structure and type of data.

Step – 3 : Think of a problem solving approach.
This is a very important and the most difficult step in writing an algorithm. The programmer has to come up with a problem solving approach that will help us to build the model to solve the given problem.

Experience and practice is an important factor in thinking the problem solving approach. So make sure you try writing different algorithm and reading algorithms that will assist you for better understanding.

Step – 4 : Review the problem solving approach and try to think of a better Alternative.
A high quality Algorithm must contain the best approach to solve a problem which will help in reducing the effort in coding as well as decrease time complexity and size of the program.

Therefore, when you think of a problem solving approach try thinking of a better alternative with better results, this will help you generate a finer programming logic.

Also review the problem solving approach, and make sure the problem will be solved through it.

Step – 5 : Develop a basic structure of the Algorithm.
Develop a basic structure of the problem solving approach; explain the approach step-by-step with short and effective description.

Step – 6 : Optimize, Improve and refine.
After developing an Algorithm try optimizing the explanation to increase readability and accessibility of it.

Also try Improving and refining the algorithm for better understanding and efficient for use.

4. Outline the components of a good algorithm?

Input and Output must be specified
An Input is the data transferred by the user to the program to produce certain output.

An Algorithm can have 0 to more inputs from the user. If Inputs must be taken from user the details of the data must be specified in the algorithm.

An Output is the data transferred by the program to the user results the computations.

An Algorithm must have at least 1 well-defined and desired output.

All important steps must be mentioned
An Algorithm comprises of all short step-by-step processes that is performed in a program, thus every important step must be present in the Algorithm in considerate details.

The steps mentioned in an algorithm must lack grammatical mistakes to avoid misunderstanding and confusion.

Instructions must be perfectly ordered
An Algorithm is a very important step in programming thus, it must be perfectly ordered to lack severe errors and confusion while coding.

An Algorithm helps and supports a programmer while coding to avoid errors, therefore a well ordered algorithm will provide better assistance.

Short and effective descriptions
An Algorithm must contain short but effective description of the process meant to be conducted, to increase reliability and efficiency of it.

The Algorithm must contain finite number of steps
An Algorithm must conclude after performing certain operation and generating an output. The loops mentioned in the algorithm must terminate after performing the operations.

The Algorithm must contain finite amount of steps to generate a valid output.

5. Describe the Tree traversal method?

Traversal is a process to visit all the nodes of a tree and may print their values too. Because, all nodes are connected via edges (links) we always start from the root (head) node. That is, we cannot randomly access a node in a tree. There are three ways which we use to traverse a tree −
In-order Traversal
Pre-order Traversal
Post-order Traversal

Generally, we traverse a tree to search or locate a given item or key in the tree or to print all the values it contains.

In-order Traversal
In this traversal method, the left subtree is visited first, then the root and later the right sub-tree. We should always remember that every node may represent a subtree itself.

If a binary tree is traversed in-order, the output will produce sorted key values in an ascending order.

Pre-order Traversal
In this traversal method, the root node is visited first, then the left subtree and finally the right subtree.

Post-order Traversal
In this traversal method, the root node is visited last, hence the name. First we traverse the left subtree, then the right subtree and finally the root node.

6. Explain the difference between inorder and postorder tree traversal?

Inorder Traversal :

In a binary tree, inorder traversal the order of traversing the nodes is :

Left node → Root node → Right node

Postorder Traversal :

Postorder traversal the order of traversing the nodes is :

Left node → Right node → Root node.