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

a)

Lists

A list is an ordered sequence of elements. It is a non-scalar data structure and mutable in nature. A list can contain distinct data types in contrast to arrays storing elements belonging to the same data types.

Lists can be accessed by the help of the index by enclosing index within square brackets.

Tuples

It is also a non-scalar type defined in Python. Just like lists, it is also an ordered sequence of characters but tuples are immutable by nature. This means any modification is not allowed with this data structure.

Elements can be heterogeneous or homogenous nature separated by commas within the parenthesis.

Sets

It is an unordered collection of objects without any duplicates. This can be done by enclosing all the elements within curly braces. We can also form sets by using type casting via a keyword “set”.

Elements of a set must of immutable data types. Set doesn’t support indexing, slicing, concatenation & replication. We can iterate over elements using the index.

Dictionary

A dictionary is an unordered sequence of key-value pairs. Indices can be of any immutable type and are called keys. This is also specified within curly braces.

We can access the values by the help of unique keys associated with them.

2. Describe the Python user data structure?

a)

Linked list

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers.

Stack

A stack is a linear structure that allows data to be inserted and removed from the same end thus follows a last in first out(LIFO) system. Insertion and deletion is known as push() and pop() respectively.

Queue

A queue is a linear structure that allows insertion of elements from one end and deletion from the other. Thus it follows, First In First Out(FIFO) methodology. The end which allows deletion is known as the front of the queue and the other end is known as the rear end of the queue.

Tree

A tree is a non-linear but hierarchical data structure. The topmost element is known as the root of the tree since the tree is believed to start from the root. The elements at the end of the tree are known as its leaves. Trees are appropriate for storing data that aren't linearly connected to each other but form a hierarchy.

Graph

A Graph is a non-linear data structure consisting of nodes and edges. The nodes are sometimes also referred to as vertices and the edges are lines or arcs that connect any two nodes in the graph. A Graph consists of a finite set of vertices (or nodes) and set of Edges which connect a pair of nodes.

Hashmap

Hash maps are indexed data structures. A hash map makes use of a hash function to compute an index with a key into an array of buckets or slots. Its value is mapped to the bucket with the corresponding index. The key is unique and immutable. In Python, dictionaries are examples of hash maps.

3. Describe the stages involved in writing an algorithm?

a)

The development of an algorithm (a plan) is a key step in solving a problem. Once we have an algorithm, we can translate it into a computer program in some programming language. Our algorithm development process consists of five major steps.

Step 1: Obtain a description of the problem.

Step 2: Analyze the problem.

Step 3: Develop a high-level algorithm.

Step 4: Refine the algorithm by adding more detail.

Step 5: Review the algorithm.

4. Outline the components of a good algorithm?

a)

1.Clear and Unambiguous: Algorithm should be clear and unambiguous. Each of its steps should be clear in all aspects and must lead to only one meaning.

2.Well-Defined Inputs: If an algorithm says to take inputs, it should be well-defined inputs.

3.Well-Defined Outputs: The algorithm must clearly define what output will be yielded and it should be well-defined as well.

4.Finiteness: The algorithm must be finite, i.e. it should not end up in an infinite loops or similar.

5.Feasible: The algorithm must be simple, generic and practical, such that it can be executed upon with the available resources. It must not contain some future technology, or anything.

6.Language Independent: The Algorithm designed must be language-independent, i.e. it must be just plain instructions that can be implemented in any language, and yet the output will be same, as expected.

5. Describe the Tree traversal method?

a)

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 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

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

a)

Inorder traversal: To traverse a binary tree in Inorder, following operations are carried-out

(i) Traverse the left most subtree starting at the left external node,

(ii) Visit the root, and

(iii) Traverse the right subtree starting at the left external node.

In Postorder, following operations are carried-out

(i) Traverse all the left external nodes starting with the left most subtree which is then followed by bubble-up all the internal nodes,

(ii) Traverse the right subtree starting at the left external node which is then followed by bubble-up all the internal nodes, and

(iii) Visit the root.