* What is the divide and conquer strategy?

Divide and conquer strategy is about dividing a big problem into subproblems and combining solutions after solving each problem. This strategy recursively breaks down big problems into subproblems. You can also divide and conquer subproblems until the problem gets quite simple and there is a direct answer to it. If the problem is about sorting, then subproblems should also become about sorting. But subproblems should answer same questions as big problem answers. That means divide and conquer has a recursive nature. And in the end, you should have a method to combine all solutions to subproblems to solve big problems.

* What is binary search and how does it work?

Binary search only works when array is sorted. Means that either array is in ascending or descending order. Application of divide and conquer strategy and its time complexity is O (logn). Let's say we are given an array which is sorted in ascending order. First, we are calculating the middle index to divide array into two parts. The middle index element of an array is also middle value since our array is sorted. We compare this value to the number we are searching for. If the middle value is less than the number, we are looking for then we will look to the right side of an array, if the middle value is bigger than number then we will do vice versa. Now the area we are searching for is half the length of array and we should find a new middle index and so, middle value to compare and find the number we are searching. To find the middle index we take the start index of the right side of array and last index, sum these two indexes, and divide by 2. If the number we are searching for was in the left side of array, we would find the sum of start index of an array and middle index of array and divide by 2. Then this process continues until we find the number.

* Explain the distinction between a list and a tuple.

Both list and tuple can store multiple data which can be either from unique data types or different data types. But there are some important distinctions between these 2 data types.

1. Lists are mutable, however tuples are immutable

We can insert, append, or delete an element from the list. However, if we try to do the same operations on the tuples, we will get an error. So, more operations can work on lists, but not on tuples.

2. Tuples are faster than lists

Because tuples are immutable it brings them an advantage of making process faster.

3. Tuples are memory efficient than lists

* Can you explain how Python manages memory?

Let's say we have created an x which has a value of 10. Then when we create a new object y which has the same value y will point to the same memory address as x. The reason is Python optimizes memory utilization by allocating same object reference to a new variable if object already exists with the same value. As we create a new variable, memory is divided into two sections. Stack memory and heap

memory. The main method is created under stack memory and the program begins its execution under stack memory. Objects are created in heap memory whereas references are created in stack memory.

If we create a new function then a new stack frame is created on invocation of a function which holds the function with its variable. After execution of function the result object of function is stored in the heap memory. And as the function's execution finishes, the stack frame is removed from stack memory.

* What is the difference between pickling and unpickling?

"Pickling" is the process whereby a Python object hierarchy is converted into a byte stream, and "unpickling" is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy.

* What are the different types of search algorithms?

There are two types of searching algorithms

1. Sequential Search: Linear Search - In this, the list or array is traversed sequentially and every element is checked
2. Interval Search: Binary Search - These algorithms are specifically designed for searching in sorted data-structures.