Q.1: What are data structures, and why are they important?

Ans: A data structure is a way to organize and store data in a computer. Data structures are important because they make it easier for computers to process large, complex sets of information. They also help programmers organize and manage data in a structured way.

Q.2: Explain the difference between mutable and immutable data types with examples?

Ans: Mutable data types are those whose values can be modified after they are created, and Immutable data types are those whose values cannot be modified once created
Python has a Dictionary, set, and list as mutable data types, and numbers, strings, and tuples are immutables data types.

	Mutable	Immutable
Definition	Data type whose values can be changed after creation.	Data types whose values can't be changed or altered.
Memory Location	Retains the same memory location even after the content is modified.	Any modification results in a new object and new memory location
Example	List, Dictionaries, Set	Strings, Types, Integer

Q3: What are the main differences between lists and tuples in Python?

Ans: Main Difference Between list And Tuple in Python.

Sno	LIST	TUPLE
1	Lists are mutable	Tuples are immutable

2	The implication of iterations is Time-consuming	The implication of iterations is comparatively Faster
3	The list is better for performing operations, such as insertion and deletion.	A Tuple data type is appropriate for accessing the elements

Q4: Describe how dictionaries store data?

Ans: Dictionaries are used to store data values in Key Value pairs.

Q5: Why might you use a set instead of a list in Python?

Ans: Because sets cannot have multiple occurrences of the same element, it makes sets highly useful to efficiently remove duplicate values from a list or tuple and to perform common math operations like unions and intersections.

Q6: What is a string in Python, and how is it different from a list?

Ans: A string is a sequence of characters between single or double quotes. A list is a sequence of items, where each item could be anything (an integer, a float, a string, etc).

Q7: How do tuples ensure data integrity in Python?

Ans: Tuples are immutable to ensure that their contents remain constant throughout their lifecycle, guaranteeing data integrity and reliability. This immutability allows tuples to be used as keys in dictionaries and elements in sets, as they can be hashed.

Q8: What is a hash table, and how does it relate to dictionaries in Python?

Ans: Hash tables store key-value pairs and the key is generated using a hash function. Hash tables or maps in Python are implemented through the built-in dictionary data type. The keys of a dictionary in Python are generated by a hashing function.

Q9: Can lists contain different data types in Python?

Ans: A list can also contain a mix of Python types including strings, floats, and boolean But its Not used.

Q10: Explain why strings are immutable in Python?

Ans: Strings in Python are “immutable” which means they can not be changed after they are created. Some other immutable data types are integers, float, boolean, etc.
The immutability of Python string is very useful as it helps in hashing, performance optimization, safety, ease of use, etc.

Q11: What advantages do dictionaries offer over lists for certain tasks?

Ans: Getting the elements from the list data structure is more complex when compared to dictionaries. So it is more efficient to use dictionaries for the searching of elements as it can be carried out in a much faster manner.

Q12: Describe a scenario where using a tuple would be preferable over a list?

Ans: Use a tuple if you need an immutable collection where the elements won't change after creation. Tuples are generally faster and more memory-efficient than lists, making them better for fixed collections, especially as dictionary keys or when iteration speed is crucial.

Q13: How do sets handle duplicate values in Python?

Ans: Set is not allowed to store duplicated values by definition. If you need duplicated values, use a List. As specified on the documentation of the interface, when you try to add a duplicated value, the method add returns false, not an Exception.

Q14: How does the “in” keyword work differently for lists and dictionaries?

Ans: The 'in' keyword in Python returns True if a certain element is present in a Python object, else it will return False. It has two purposes: To check if a value is present in a list, tuple, range, string, etc. To iterate through a sequence in a for loop.
If you use the in or not in operators directly on a dictionary, then it’ll check whether the dictionary has a given key or not.

Q15: Can you modify the elements of a tuple? Explain why or why not?

Ans: No, you cannot modify the elements of a tuple in Python because tuples are "immutable," meaning once created, their elements cannot be changed, added, or removed; if you need to modify data, you should use a list instead which is mutable.

Q16: What is a nested dictionary, and give an example of its use case?

Ans: A basic nested dictionary might look like this: nested_dict = {'key1': {'subkey1': 'value1', 'subkey2': 'value2'}, 'key2': {'subkey1': 'value3', 'subkey2': 'value4'}}.

Use case

Nesting dictionaries can be used to create hierarchies of dictionaries that model complex

information. For example, you could create a nested dictionary to store information about a family, with each child represented by a dictionary that contains their name and year of birth.

To access items in a nested dictionary, you can use the names of the dictionaries, starting with the outer dictionary. For example, to print the name of child 2, you would use print(myfamily["child2"]["name"])

Q17: Describe the time complexity of accessing elements in a dictionary?

Ans: The dictionary try to access a value by calling the key value of that data. When it tries to access the value, it already knows the encryption process and algorithm of Key encryption, and simply finds it with Bigo of O(1) time. That's why it only takes O(1) time to access any element in the dictionary.

Q18: In what situations are lists preferred over dictionaries?

Ans: when we do not required data in key value pair then we go for list.

Q19: Why are dictionaries considered unordered, and how does that affect data retrieval?

Ans: A dictionary is termed an unordered collection of objects because dictionaries do not maintain any inherent order of the items based on when they were added. In older versions of Python dictionaries did not preserve insertion order at all. This meant that when you accessed or printed the items, the order could very, as dictionaries were optimized for fast lookups rather than maintaining order.

Q20: Explain the difference between a list and a dictionary in terms of data retrieval?

Ans: A list is an ordered collection of items, whereas a dictionary is an unordered data collection in a key: value pair. Elements from the list can be accessed using the index, while the elements of the dictionary can be accessed using keys.