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

A: Data structures are ways of organizing and storing data so it can be accessed and modified efficiently. They are important because they determine how quickly and easily operations like searching, inserting, or deleting data can be performed.

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

A: Mutable data types can be changed after creation (lists, dictionaries, sets). Immutable data types cannot be changed after creation (strings, tuples). Example: a list can have elements added or removed, but a string's content cannot be altered without creating a new string.

3. What are the main differences between lists and tuples in Python?

A: Lists are mutable, can be modified, and usually used when the data might change. Tuples are immutable, generally faster, and are used for fixed collections of items.

4. Describe how dictionaries store data.

A: Dictionaries store data as key-value pairs. Internally, they use a hash table where each key's hash determines where the value is stored in memory.

5. Why might you use a set instead of a list in Python?

A: Sets automatically remove duplicate elements and allow fast membership checks. They are useful when uniqueness matters.

6. What is a string in Python, and how is it different from a list?

A: A string is an immutable sequence of characters, while a list is a mutable sequence that can hold elements of any data type.

7. How do tuples ensure data integrity in Python?

A: Since tuples are immutable, their content cannot be altered after creation. This ensures that the data stays unchanged, preserving its integrity.

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

A: A hash table is a data structure that maps keys to values using a hash function. Python dictionaries are implemented using hash tables.

9. Can lists contain different data types in Python?

A: Yes, Python lists can hold elements of different data types, such as integers, strings, and objects, in the same list.

10. Explain why strings are immutable in Python.

A: Strings are immutable to make them hashable and to ensure safety when used as keys in dictionaries, and for performance optimization.

11. What advantages do dictionaries offer over lists for certain tasks?

A: Dictionaries provide faster lookups for data when you know the key, while lists require searching through elements.

12. Describe a scenario where using a tuple would be preferable over a list.

A: When you need to store fixed data that should not change, such as geographic coordinates (latitude, longitude).

13. How do sets handle duplicate values in Python?

A: Sets automatically discard duplicate values, storing only unique elements.

14. How does the 'in' keyword work differently for lists and dictionaries?

A: For lists, 'in' checks if a value exists among the list's elements. For dictionaries, 'in' checks if a key exists.

15. Can you modify the elements of a tuple? Explain why or why not.

A: No, tuples are immutable, so their elements cannot be modified once created.

16. What is a nested dictionary, and give an example of its use case.

A: A nested dictionary is a dictionary that contains other dictionaries as values. Example: student_data = {'John': {'age': 20, 'grade': 'A'}}

17. Describe the time complexity of accessing elements in a dictionary.

A: Accessing elements by key in a dictionary has an average time complexity of O(1) due to hash table implementation.

18. In what situations are lists preferred over dictionaries?

A: Lists are preferred when you need to store items in a specific order or when order matters more than fast key-based lookup.

19. Why are dictionaries considered unordered, and how does that affect data retrieval?

A: Before Python 3.7, dictionaries did not preserve insertion order. Even now, order is not meant for data retrieval; keys are looked up by their hash values, not position.

20. Explain the difference between a list and a dictionary in terms of data retrieval.

A: Lists retrieve elements by their index position, while dictionaries retrieve elements by their keys.