**Data Types and Structures**

---

1. **What are data structures, and why are they important?**
   **Data structures help in storing, organizing, and accessing data efficiently.**
   **They are important for solving problems quickly and using memory effectively.**
   **Examples include lists, dictionaries, tuples, and sets.**

---

2. **Difference between mutable and immutable data types:**
   **Mutable types can be changed after creation (e.g., lists, dictionaries).**
   **Immutable types cannot be changed once created (e.g., strings, tuples).**
   **Changing mutable objects affects the original data, unlike immutable ones.**

---

3. **Main differences between lists and tuples:**
   **Lists are mutable (can change), while tuples are immutable (cannot change).**
   **Lists use [], tuples use ().**
   **Tuples are faster and use less memory.**

---

4. **How dictionaries store data:**
   **Dictionaries store data as key-value pairs.**
   **They use a hash table to map keys to values.**
   **Accessing values using keys is fast and efficient.**

---

5. **Why use a set instead of a list:**
   **Sets automatically remove duplicate values.**
   **They offer faster lookup for checking membership.**
   **Good for storing unique elements only.**

---

6. **What is a string in Python and how it differs from a list:**
   **A string is a sequence of characters, immutable in nature.**
   **A list is a collection of items, mutable and can have mixed types.**
   **Strings can't be changed; lists can.**

---

7. **How tuples ensure data integrity:**
   **Tuples are immutable, so data cannot be modified.**
   **This protects the original values from accidental changes.**
   **They're useful for fixed data like coordinates.**

---

8. **What is a hash table and how it relates to dictionaries:**
   **A hash table stores data using key-value pairs.**
   **Dictionaries in Python are built using hash tables.**
   **Keys are hashed for fast data retrieval.**

---

9. **Can lists contain different data types?**
   **Yes, Python lists can store multiple data types.**
   **Example: [1, "hello", 3.14, True].**
   **They are flexible and dynamic.**

---

10. **Why are strings immutable in Python?**
    **Immutability ensures safety, consistency, and memory efficiency.**
    **It allows strings to be used as dictionary keys.**
    **It avoids accidental changes to data.**

---

11. **Advantages of dictionaries over lists:**
    **Dictionaries provide key-based fast access to data.**
    **They are better for storing labeled or structured data.**
    **Lists are good for simple ordered data, dictionaries for mapped data.**

---

12. **When to use a tuple instead of a list:**
    **Use tuples when you don't want the data to change.**
    **They are faster and consume less memory than lists.**
    **Great for fixed or constant data.**

---

13. **How sets handle duplicate values:**
    **Sets only store unique values; duplicates are removed automatically.**
    **They are ideal for filtering and checking unique elements.**
    **Example: {1, 2, 2, 3} → {1, 2, 3}.**

---

14. **How the “in” keyword works in lists vs dictionaries:**
    **In lists, in checks if a value exists.**
    **In dictionaries, it checks if a key exists.**
    **For dictionaries, in doesn’t check values unless specified.**

---

15. **Can you modify elements of a tuple?**
    **No, tuples are immutable.**
    **Once created, their elements can't be changed or replaced.**
    **Trying to modify causes an error.**

---

16. **What is a nested dictionary and use case:**
    **A nested dictionary has dictionaries inside another dictionary.**
    **It's used to store complex, structured data.**
    **Example: storing student details with names as keys.**

---

17. **Time complexity of accessing dictionary elements:**
    **Accessing an element by key in a dictionary is O(1) on average.**
    **It’s very fast due to the use of hash tables.**
    **Worst case is O(n) but rare.**

---

18. **When are lists preferred over dictionaries?**
    **Lists are better when data has no labels or keys.**
    **Use them when order and position matter.**
    **They’re simple and efficient for sequences.**

---

19. **Why were dictionaries considered unordered?**
    **Before Python 3.7, dictionaries didn't preserve insertion order.**
    **Now they maintain the order of items added.**
    **Still, access is always by key, not position.**

---

20. **Difference between a list and a dictionary (data retrieval):**
    **Lists use index numbers to access values (e.g., list[0]).**
    **Dictionaries use keys to access values (e.g., dict["name"]).**
    **Dictionaries are better for labeled data.**