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

Data structures are organized ways of storing and managing data in memory. They are important because they:

- Make data storage efficient.
- Enable faster access, modification, and processing.
- Allow solving complex problems (e.g., graphs, trees, queues).

---

2. Difference between mutable and immutable data types with examples

Mutable: Can be changed after creation.
Example:

```
my_list = [1, 2, 3]
my_list[0] = 10   # Allowed
```

-

Immutable: Cannot be changed once created.
Example:

```
my_str = "hello"
# my_str[0] = "H"  ❌ Error
```

-

---

3. Main differences between lists and tuples in Python

Feature	List ([ ])	Tuple (( ))
Mutability	Mutable	Immutable
Speed	Slower	Faster
Use case	Frequent updates	Fixed data

---

4. How dictionaries store data

Dictionaries store data as **key-value pairs** using a **hash table** internally. Keys must be unique and hashable, values can be anything.

---

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

- Sets automatically remove duplicates.
- Membership testing (`in`) is faster in sets (O(1)) compared to lists (O(n)).

---

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

- String: Immutable sequence of characters.

List: Mutable sequence that can hold any data type.

```
s = "hello"     # cannot change characters
l = ["h","e"]   # can replace elements
```

-

---

7. How do tuples ensure data integrity in Python?

Because tuples are immutable, their data cannot be altered accidentally, making them reliable for fixed records (like database rows).

---

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

A **hash table** is a data structure that maps keys to values using a hash function.
In Python, dictionaries are built on hash tables for fast lookups, insertions, and deletions.

---

9. Can lists contain different data types in Python?

Yes.

```
mixed = [1, "hello", 3.14, True]
```

---

10. Why are strings immutable in Python?

- For security (strings are widely used in identifiers, paths, etc.).
- For memory efficiency (same string reused across programs).
- For consistency in hash-based structures (like dictionary keys).

---

11. Advantages of dictionaries over lists

- Fast lookups by key (O(1)).
- More meaningful data storage (key-value pairs instead of index-based).

---

12. How do sets handle duplicate values in Python?

Sets automatically remove duplicates:

```
s = {1,2,2,3}
print(s)  # {1,2,3}
```

---

13. Scenario where tuple is preferable over list

- When storing **fixed data** like coordinates `(x, y)` or months of the year.
- Useful as **dictionary keys** since they are hashable.

---

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

List: Checks if a **value** is present.

```
2 in [1,2,3]  # True
```

-

Dictionary: Checks if a **key** is present.

```
"name" in {"name":"Ishan"}  # True
```

-

---

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

No, tuples are immutable. Once created, their elements cannot be changed.
However, if a tuple contains a mutable object (like a list), that list can be modified.

---

16. What is a nested dictionary, and example use case

A dictionary inside another dictionary.
Example:

```
students = {
  "101": {"name":"Alice", "age":20},
  "102": {"name":"Bob", "age":22}
}
```

Use case: Representing structured data like records or JSON.

---

17. Time complexity of accessing elements in a dictionary

- Average case: **O(1)** (very fast, thanks to hashing).
- Worst case (rare): **O(n)** (if many collisions).

---

18. Situations where lists are preferred over dictionaries

- When order matters.
- When storing simple collections without needing key-value mapping.
- When indexing by position is required.

---

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

Before Python 3.7, dictionaries did not preserve insertion order. Retrieval is based on keys, not positions.
Since 3.7+, they maintain insertion order, but conceptually they are still key-based.

---

20. Difference between a list and a dictionary in terms of data retrieval

List: Retrieve by position (index).

```
my_list[0]
```

-

Dictionary: Retrieve by key.

```
my_dict["name"]
```

-