Data Structure Assignment.

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

Data structures are organized formats for storing and managing data. They are important because they enable efficient data access, manipulation, and storage, which are essential for writing optimized programs.

---

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

- **Mutable**: Can be changed after creation.
  Example: `list = [1, 2, 3]` → `list[0] = 5`

- **Immutable**: Cannot be changed after creation.
  Example: `tuple = (1, 2, 3)` → Cannot modify any element.

---

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

- Lists are mutable, while tuples are immutable.
- Lists use square brackets `[]`, while tuples use parentheses `()`.
- Lists are slower, tuples are faster due to fixed size.

---

4. Describe how dictionaries store data.

Dictionaries store data as key-value pairs. Each key maps to a specific value, allowing for fast data lookup using the key.

---

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

Sets automatically remove duplicates and offer faster membership testing due to their underlying hash-based structure.

---

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

A string is a sequence of characters (immutable), whereas a list is a sequence of elements that can be of any type (mutable).

---

7. How do tuples ensure data integrity in Python?

Tuples are immutable, meaning once created, their content cannot change. This immutability makes them ideal for storing fixed data.

---

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

A hash table stores data using a hash function that maps keys to values. Python dictionaries use hash tables internally for efficient key-based access.

---

9. Can lists contain different data types in Python?

Yes, Python lists can contain elements of different data types.
Example: `["hello", 123, True]`

---

10. Explain why strings are immutable in Python.

Strings are immutable to ensure data integrity, efficient memory usage, and safe sharing across different parts of a program.

---

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

Dictionaries offer faster lookups through keys, while lists require a search through all elements to find a value.

---

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

Tuples are preferable when storing fixed data like coordinates `(x, y)` or constant values that should not be modified.

---

13. How do sets handle duplicate values in Python?

Sets automatically discard duplicate elements. Only unique values are stored.

---

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

- In a list, it checks if an element exists.
- In a dictionary, it checks if a key exists (not the value).

---

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

No, tuples are immutable. Once created, their elements cannot be changed. This behavior preserves data integrity.

---

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

A nested dictionary is a dictionary within another dictionary.
Example:

python
CopyEdit

```
student = {
  "name": "Alex",
  "grades": {"math": 90, "science": 95}
}
```

Use case: storing structured data like student profiles.

---

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

Accessing elements in a dictionary takes constant time: **O(1)**, on average, due to its hash-based structure.

---

18. In what situations are lists preferred over dictionaries?

Lists are preferred when maintaining order and accessing elements by index is required, such as ordered data or sequences.

---

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

Before Python 3.7, dictionaries were unordered, meaning the order of key-value pairs was not guaranteed. Since 3.7, they preserve insertion order, but access is still key-based rather than by position.

---

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

- **List**: Elements are accessed by index.
  Example: `list[0]`

- **Dictionary**: Elements are accessed by key.
  Example: `dict["name"]`