Q.1- Discuss string slicing and provide examples

Ans.- String Slicing in Python: String slicing allows accessing a subset of a string using the syntax string [start:end:step].

- **start**: The index to start slicing (inclusive).
- **end**: The index to stop slicing (exclusive).
- **step**: The step value for slicing.

Examples:

```
text = "Hello, World!"

print(text[0:5])

print(text[7:])

print(text[::2])

print(text[::-1])
```

Q.2- Explain the key features of lists in Python

Ans.- Key Features of Lists in Python

- **Dynamic:** Lists can grow or shrink in size.
- **Heterogeneous:** Can store elements of different data types.
- **Mutable:** Supports modification (add, update, delete).
- **Indexed:** Elements are accessed using zero-based indexing.
- **Ordered:** Maintains the order of elements.

Example:

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

Q.3- Describe how to access, modify, and delete elements in a list with examples

Ans.- Access:

```
my_list = [10, 20, 30, 40]

print(my_list[1])
```

Modify:

```
my_list[1] = 25

print(my_list)
```

Delete:

```
del my_list[2]
print(my_list)
my_list.remove(25)
print(my_list)
removed_item = my_list.pop(0)
print(removed_item)
print(my_list)
```

Q.4- Compare and contrast tuples and lists with examples

Ans.- Comparing Tuples and Lists

Feature	Tuple	List
Mutability	Immutable	Mutable
Performance	Faster for read-only tasks	Slower due to mutability
Syntax	(1, 2, 3)	[1, 2, 3]
Use Cases	Data that shouldn't change	Data needing modification

---

Example:

my_tuple = (1, 2, 3)

my_list = [1, 2, 3]

Q.5- Describe the key features of sets and provide examples of their use

Ans.- Key Features of Sets

- Unordered: No indexing or slicing.
- Unique Elements: Duplicate values are not allowed.
- Mutable: Elements can be added or removed.
- Efficient Operations: Fast membership testing and set operations like union, intersection, etc.

Example:

```
my_set = {1, 2, 3}
```

```
my_set.add(4)
print(my_set)
```

Q.6- Discuss the use cases of tuples and sets in Python programming

Ans.- Tuples:

- Represent immutable groups of data (e.g., coordinates, configurations).
- Useful as dictionary keys or elements of a set.

Sets:

- Eliminate duplicate elements.
- Perform mathematical set operations (union, intersection, etc.).
- Efficiently test membership.

Q.7- Describe how to add, modify, and delete items in a dictionary with examples

Ans.- Adding:

```
my_dict = {"name": "Alice"}
my_dict["age"] = 25
print(my_dict)
```

Modifying:

```
my_dict["age"] = 26
print(my_dict)
```

Deleting:

```
del my_dict["age"]
print(my_dict)
my_dict.pop("name")
print(my_dict)
```

Q.7- Discuss the importance of dictionary keys being immutable and provide examples

Ans.- Dictionary keys must be immutable to ensure they remain hashable (their hash values don't change). Mutable keys like lists could cause inconsistencies.

Example: my_dict = {("x", "y"): "point", 42: "answer"} 4o