Python Data Structures

In Python, ***List, Tuple, Set, and Dictionary*** are four different data structures used to store collections of elements. Each of them has distinct characteristics and is used for specific purposes. Below, I'll explain each one in detail along with examples:

1. **List**:

- A list is an ordered collection of elements that can be of any data type (e.g., integers, strings, other lists, etc.).
- Lists are mutable, meaning you can modify their elements after they are created.
- They are defined using square brackets `[ ]`.

Example:

```
# Creating a list of integers
my_list = [1, 2, 3, 4, 5]

# Adding an element to the end of the list
my_list.append(6)

# Modifying an element
my_list[2] = 7

# Accessing elements
print(my_list[0])  # Output: 1
```

```
# List can contain different data types
mixed_list = [1, 'hello', True, 3.14]
```

2. **Tuple**:

- A tuple is similar to a list, but it is immutable, meaning you cannot change the elements after creation.
- Tuples are defined using parentheses `( )`.

Example:

```
# Creating a tuple
my_tuple = (1, 2, 3, 4, 5)

# Accessing elements
print(my_tuple[0])  # Output: 1

# Attempting to modify will result in an error
my_tuple[2] = 7  # This will raise a TypeError
```

3. **Set:**

- A set is an unordered collection of unique elements. It is useful when you want to perform mathematical operations like union, intersection, etc., on collections of elements.
- Sets are defined using curly braces `{ }`.

Example:

```
# Creating a set
my_set = {1, 2, 3, 4, 5}

# Adding an element to the set
my_set.add(6)

# Trying to add a duplicate element, it won't raise an error but won't change the set
my_set.add(2)

# Performing set operations
set1 = {1, 2, 3}
set2 = {3, 4, 5}
union = set1.union(set2)  # Output: {1, 2, 3, 4, 5}
intersection = set1.intersection(set2)  # Output: {3}
```

4. **Dictionary:**

- A dictionary is an unordered collection of key-value pairs. It allows you to store and retrieve data based on a unique key.
- Dictionaries are defined using curly braces `{ }`, and each key-value pair is separated by a colon `:`.

Example:

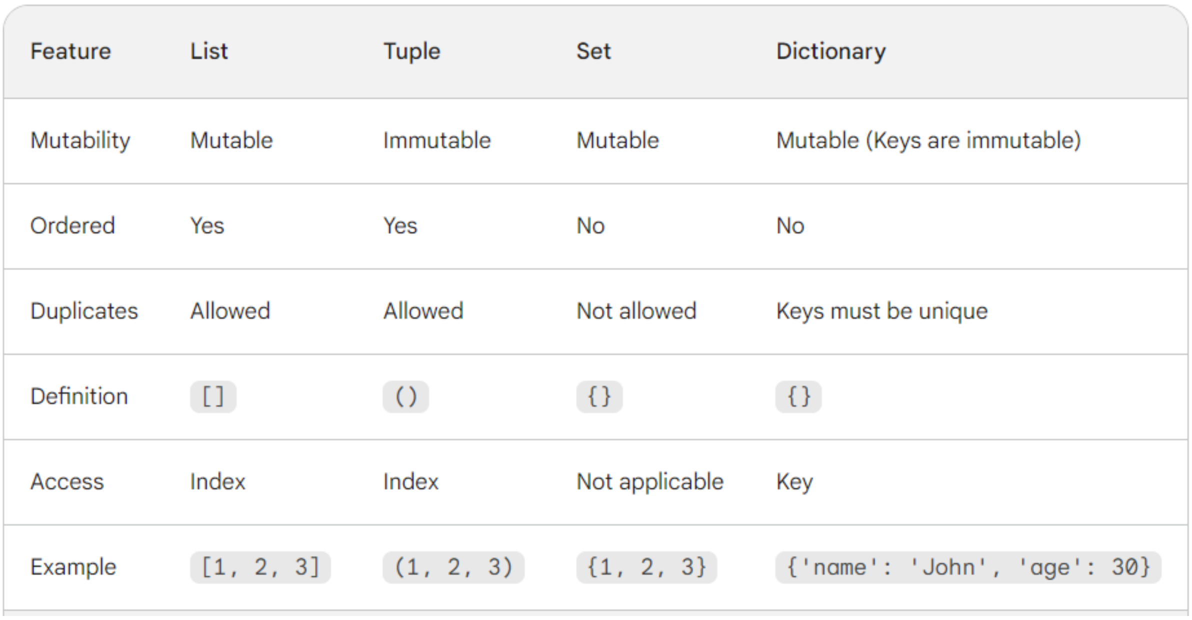
```
# Creating a dictionary
my_dict = {'name': 'John', 'age': 30, 'city': 'New York'}
```

```
  # Accessing values using keys
  print(my_dict['name'])  # Output: John

  # Modifying a value
  my_dict['age'] = 31

  # Adding a new key-value pair
  my_dict['occupation'] = 'Engineer'

  # Dictionary keys must be unique, so if you add another value with the same key, it will overwrite the existing value
  my_dict['age'] = 32
```

To summarize:

- Lists and Tuples are ordered collections, but Lists are mutable while Tuples are immutable.
- Sets are unordered collections of unique elements, useful for mathematical operations.
- Dictionaries are collections of key-value pairs, allowing efficient lookup based on keys.

A tabular format summarizing the differences between Lists, Tuples, Sets, and Dictionaries in Python:



Feature	List	Tuple	Set	Dictionary
Mutability	Mutable	Immutable	Mutable	Mutable (Keys are immutable)
Ordered	Yes	Yes	No	No
Duplicates	Allowed	Allowed	Not allowed	Keys must be unique
Definition	`[]`	`()`	`{}`	`{}`
Access	Index	Index	Not applicable	Key
Example	`[1, 2, 3]`	`(1, 2, 3)`	`{1, 2, 3}`	`{'name': 'John', 'age': 30}`

Please note that in the "Access" column, "Index" for Lists and Tuples means accessing elements using their position in the collection, while for Dictionaries it means using the key to retrieve the corresponding value. For Sets, there is no specific position-based access.