2. Handout

Python Data Structures: Lists, Tuples, Dictionaries, and Sets

Introduction to Data Structures

Data structures are ways to store and organize data so that it can be accessed and worked with efficiently. In Python, the fundamental data structures include Lists, Tuples, Dictionaries, and Sets. Understanding these will help you manage and organize data effectively.

---

1. Lists

A **list** is a collection of items that are ordered and mutable. Lists allow duplicate elements and can store different types of data.

Key Characteristics:

- **Ordered**: Items have a defined order.
- **Mutable**: Items can be changed, added, or removed.
- **Allows Duplicates**: Multiple items with the same value are allowed.

Examples:

1. Creating a list with different data types:

   ```
   my_list = [1, 2, 3, "apple", "banana", 3]
   ```

2. Accessing elements by index:

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

3. Adding and removing elements:

   ```
   my_list.append("orange")
   my_list.remove("apple")
   ```

2. Tuples

A **tuple** is similar to a list, but it is immutable. Once created, you cannot change, add, or remove items from a tuple.

Key Characteristics:

- **Ordered**: Items have a defined order.
- **Immutable**: Cannot be modified after creation.
- **Allows Duplicates**: Duplicate values are allowed.

Examples:

1. Creating a tuple:

   ```
   my_tuple = (1, 2, 3, "apple", "banana", 3)
   ```

2. Accessing elements by index:

   ```
   print(my_tuple[1])  # Output: 2
   ```

3. Finding the length of a tuple:

   ```
   print(len(my_tuple))  # Output: 6
   ```

3. Dictionaries

A **dictionary** is a collection of key-value pairs. Each key is unique and used to access the corresponding value.

Key Characteristics:

- **Unordered**: Items do not have a specific order.
- **Mutable**: Items can be changed, added, or removed.
- **Unique Keys**: Keys must be unique, but values can be duplicated.

Examples:

1. Creating a dictionary:

```
my_dict = {"name": "John", "age": 25, "city": "Mumbai"}
```

2. Accessing values by keys:

```
print(my_dict["name"])  # Output: John
```

3. Adding and updating items:

```
my_dict["email"] = "john@example.com"
my_dict["age"] = 26
```

---

4. Sets

A **set** is a collection of unique items. Sets are unordered and unindexed, and they do not allow duplicate values.

Key Characteristics:

- **Unordered**: Items do not have a specific order.
- **Mutable**: Items can be added or removed.
- **No Duplicates**: Duplicate values are not allowed.

Examples:

1. Creating a set:

```
my_set = {1, 2, 3, "apple", "banana", 3}
```

2. Adding elements:

```
my_set.add("orange")
```

3. Removing elements:

```
my_set.remove("banana")
```

Activity: Write a Python Script Using Lists and Dictionaries

Task:

1. Create a list of student names.
2. Create a dictionary where the keys are student names and the values are their scores.
3. Add a new student and their score to the dictionary.
4. Print the updated dictionary.

Solution:

```
# Step 1: Create a list of student names
students = ["Rahul", "Priya", "Anjali"]

# Step 2: Create a dictionary with student names as keys and scores as values
scores = {"Rahul": 85, "Priya": 90, "Anjali": 78}

# Step 3: Add a new student and their score
scores["Vikram"] = 88

# Step 4: Print the updated dictionary
print(scores)
```

Expected Output:

```
{'Rahul': 85, 'Priya': 90, 'Anjali': 78, 'Vikram': 88}
```

Conclusion

In this session, we explored four essential Python data structures: **Lists, Tuples, Dictionaries, and Sets**. Each structure has unique characteristics and use cases. By understanding how to use them, you can efficiently manage and manipulate data in your

Python programs. Keep practicing to enhance your skills in using these data structures effectively!