Python Tuples and Sets

Overview

This document covers the key concepts, exercises, and interview questions related to tuples and sets in Python.

Tuples

Key Concepts

- **Definition**: Tuples are ordered, immutable collections of items.
- **Syntax**:

  ```
  tuple_name = (item1, item2, item3)
  ```

- **Immutability**: Once created, the elements of a tuple cannot be changed.
- **Accessing Elements**: You can access tuple elements using indexing and slicing.
- **Common Methods**: `count()`, `index()`
- **Unpacking**:

  ```
  a, b, c = (1, 2, 3)
  ```

- **Concatenation and Repetition**:

  ```
  tuple1 + tuple2
  tuple1 * 3
  ```

Exercises

1. **Basic Tuple Creation**

   - Create a tuple with the elements "apple", "banana", and "cherry". Print the tuple.

     ```
     fruits = ("apple", "banana", "cherry")
     print(fruits)
     ```

2. **Tuple Unpacking**

   - Create a tuple `colors = ("red", "green", "blue")`. Unpack the tuple into three variables and print them.

     ```
     colors = ("red", "green", "blue")
     red, green, blue = colors
     print(red, green, blue)
     ```

3. **Tuple Indexing**

   - Given a tuple `numbers = (10, 20, 30, 40, 50)`, print the second and last element.

```
numbers = (10, 20, 30, 40, 50)
print(numbers[1], numbers[-1])
```

Interview Questions

1. **Why are tuples considered immutable in Python?**
2. **How can you convert a list to a tuple? Provide an example.**
3. **What are some advantages of using tuples over lists?**
4. **How can you return multiple values from a function using a tuple?**
5. **Explain tuple packing and unpacking with examples.**

Sets

Key Concepts

- **Definition**: Sets are unordered collections of unique items.
- **Syntax**:

  ```
  set_name = {item1, item2, item3}
  ```

- **Uniqueness**: Sets automatically remove duplicate elements.
- **Mutability**: Sets can be modified by adding or removing elements.
- **Common Methods**: `add()`, `remove()`, `discard()`, `clear()`
- **Set Operations**:
  - Union: `|` or `set1.union(set2)`
  - Intersection: `&` or `set1.intersection(set2)`
  - Difference: `-` or `set1.difference(set2)`
  - Symmetric Difference: `^` or `set1.symmetric_difference(set2)`
- **Membership Testing**: Use `in` to check if an item exists in a set.

Exercises

1. **Basic Set Creation**

   - Create a set with the elements "apple", "banana", and "cherry". Add "orange" to the set and print it.

   ```
   fruits = {"apple", "banana", "cherry"}
   fruits.add("orange")
   print(fruits)
   ```

2. **Set Operations**

   - Given two sets `A = {1, 2, 3, 4}` and `B = {3, 4, 5, 6}`, find the union, intersection, and difference of these sets.

   ```
   A = {1, 2, 3, 4}
   B = {3, 4, 5, 6}
   union = A | B
   intersection = A & B
   difference = A - B
   print(union, intersection, difference)
   ```

3. **Membership Testing**
    - Create a set `numbers = {10, 20, 30, 40, 50}`. Check if 30 is in the set and if 60 is not in the set.

      ```python
      numbers = {10, 20, 30, 40, 50}
      print(30 in numbers)
      print(60 not in numbers)
      ```

Interview Questions

1. **How are sets different from lists and tuples in Python?**
2. **What is the time complexity of checking for membership in a set?**
3. **How can you remove duplicates from a list using a set? Provide an example.**
4. **Explain the difference between `remove()` and `discard()` methods in sets.**
5. **How can you find the symmetric difference between two sets?**

Additional Resources

- Python Official Documentation on Tuples
- Python Official Documentation on Sets
