Prepared by *Coderistic*

Chapter 2: Data Structures

Lesson 3: Sets

1. **Definition**: A set is an unordered, mutable collection of unique items, defined with `{}` or `set()`.
    - Example: `{1, 2, 3}` or `set([1, 2, 2])` → `{1, 2}` (duplicates removed).
2. **Creating Sets**:
    - Empty: `set()` (not `{}`—that's a dictionary).
    - With items: `{1, "Python", 3.14}` or `set([1, 2, 3])`.
3. **Properties**:
    - Unordered (no indexing).
    - Unique (no duplicates).
    - Mutable (can add/remove items).
4. **Accessing Elements**:
    - Use loops (`for item in set`) or membership test (`item in set`).
5. **Modifying Sets**:
    - Add: `add(item)` (single), `update(iterable)` (multiple).
    - Remove: `remove(item)` (errors if not found), `discard(item)` (no error), `pop()` (random), `clear()` (empties).
6. **Mathematical Operations**:
    - Union: `|` or `union()` (all unique items).
    - Intersection: `&` or `intersection()` (common items).
    - Difference: `-` or `difference()` (items in one, not other).
    - Symmetric Difference: `^` or `symmetric_difference()` (items in either, not both).
    - Subset/Superset: `<=` (`issubset()`), `>=` (`issuperset()`).
7. **Use Cases**:
    - Remove duplicates from a list.
    - Fast membership testing.
    - Mathematical operations (e.g., comparing datasets).
8. **Sets vs Lists vs Tuples**:

Feature	Set	List	Tuple
Syntax	`{}`	`[]`	`()` or none

Feature	Set	List	Tuple
Order	Unordered	Ordered	Ordered
Mutability	Mutable	Mutable	Immutable
Duplicates	No	Yes	Yes
Indexing	No	Yes	Yes
Use Case	Unique items, operations	Ordered, dynamic data	Fixed, ordered data

9. **Practical Example**:

```
friend1 = {"reading", "gaming"}
friend2 = {"gaming", "cooking"}
common = friend1 & friend2  # {'gaming'}
all_activities = friend1 | friend2  # {'reading', 'gaming', 'cooking'}
```

Exercises for Practice

1. Create a set of 5 numbers and add a duplicate; check the result.
2. Find the intersection of `{1, 2, 3, 4}` and `{3, 4, 5, 6}`.
3. Take a sentence as input and print all the unique words in it. Ignore uppercase/lowercase differences.
4. You are given two sets. Combine them into one set containing all unique elements and print the result.

```
set1 = {1, 2}
set2 = {2, 3}
```

   Hint: Use the union() method or the | operator.
5. You are given two sets. Find the elements that are common to both sets and print the result.

```
set1 = {1, 2, 3}
set2 = {2, 3, 4}
```

   Hint: Use the intersection() method or the & operator.
6. Count how many unique letters are in a word. Ignore case sensitive

```
word = "Hello"
```

7. You are given two sets. Check if all elements of the first set are also in the second set and print the result (`True` or `False`).

```
small_set = {1, 2}
big_set = {1, 2, 3}
```

Hint: Use the issubset() method or the <= operator.

8. Start with an empty set. Add three items to it and print the result.
9. You are given a set. Remove a specific item from it and print the updated set.

```
my_set = {1, 2, 3, 4}
```

10. You are given two sets. Find the items that are in the first set but not in the second set and print the result.

```
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}
```

Hint: Use the difference() method or the - operator.

11. You are given a set. Clear all the items from it and print the result.

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