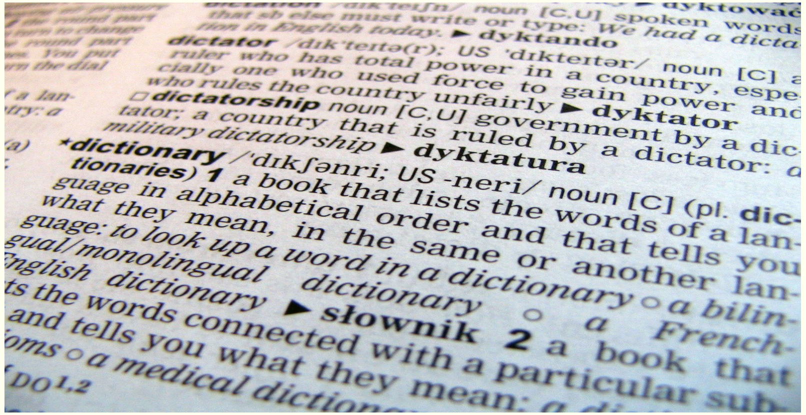
Unit:2 ; Class:8

Dictionary and Sets

Instructor: Musfique Ahmed

What is a Dictionary?

- A collection of key-value pairs.
- Unordered, changeable, and does not allow duplicate keys.
- Keys must be unique and immutable (e.g., strings, numbers, tuples).



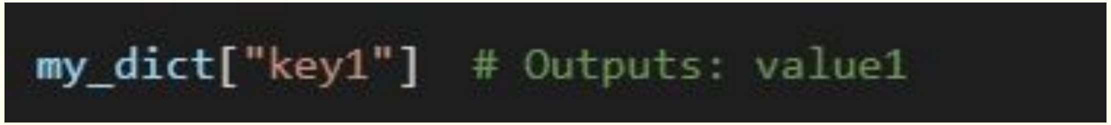
Syntax:

```
my_dict = {"key1": "value1", "key2": "value2"}
```

Practice Problem:
Create a dictionary with three key-value pairs representing a student's name, age, and grade. Print the dictionary.

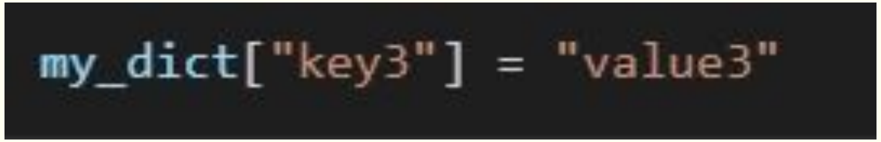
Accessing and Modifying a Dictionary

- Access values using keys:



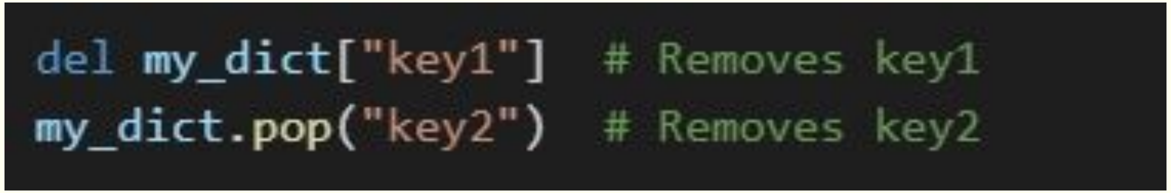
```
my_dict["key1"]  # Outputs: value1
```

- Add or update key-value pairs:



```
my_dict["key3"] = "value3"
```

- Remove items using `del` or `pop()`:



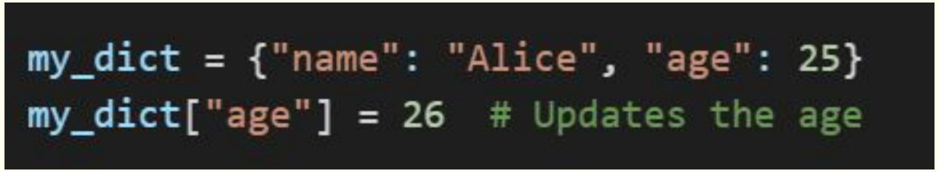
```
del my_dict["key1"]  # Removes key1
my_dict.pop("key2")  # Removes key2
```

Practice Problem:

Using the dictionary from the previous problem, update the grade, add a new key-value pair for the school name, and remove the age.

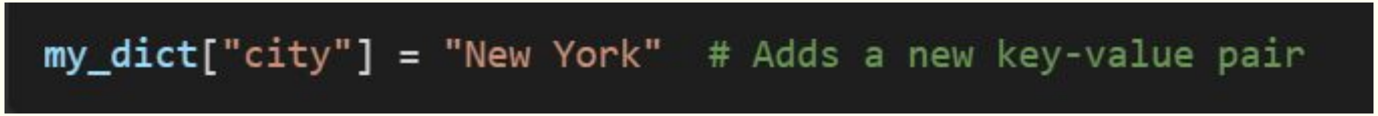
Modifying Dictionary Elements

- Change the value of an existing key:



```python
my_dict = {"name": "Alice", "age": 25}
my_dict["age"] = 26  # Updates the age
```

- Add new key-value pairs dynamically:

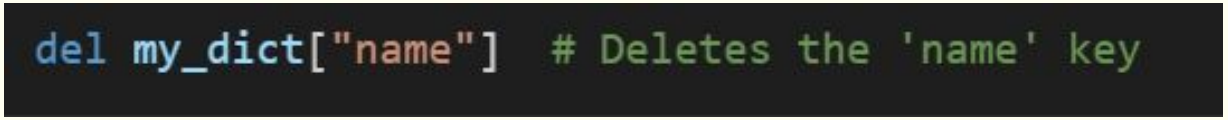


```python
my_dict["city"] = "New York"  # Adds a new key-value pair
```

Practice Problem: Create a dictionary of your favorite books and their authors. Update one author's name and add a new book.

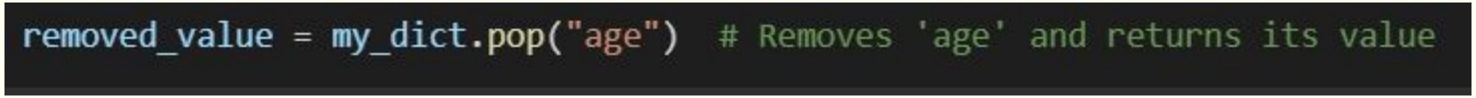
Deleting Elements in a Dictionary

- Remove a specific key:



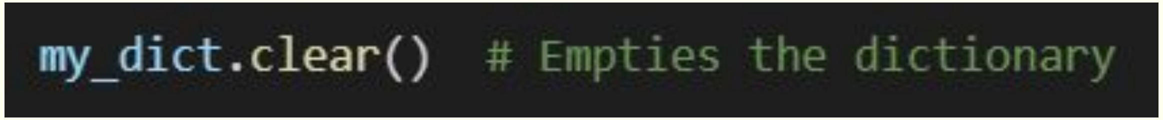
```python
del my_dict["name"]  # Deletes the 'name' key
```

- Use the `pop()` method:



```python
removed_value = my_dict.pop("age")  # Removes 'age' and returns its value
```

- Clear all elements:

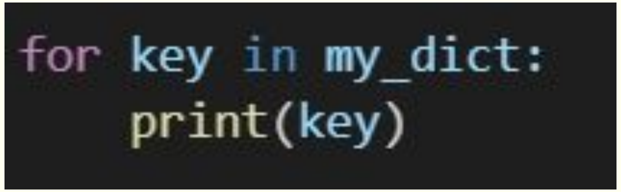


```python
my_dict.clear()  # Empties the dictionary
```

Practice Problem: Create a dictionary of three cities and their populations. Remove one city using del and another using pop().

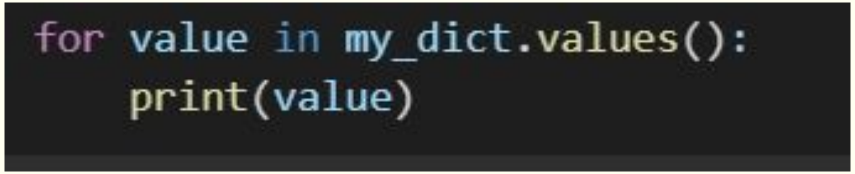
Looping Through a Dictionary

- Loop through keys:



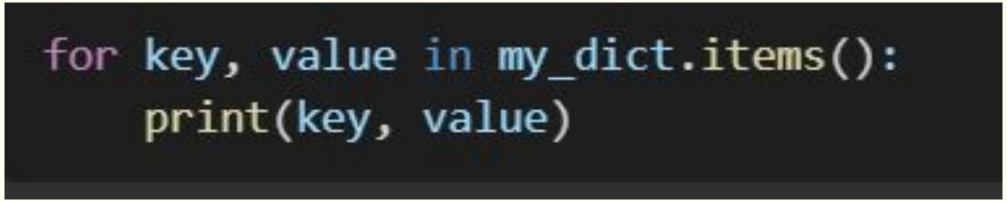
```
for key in my_dict:
    print(key)
```

- Loop through values:



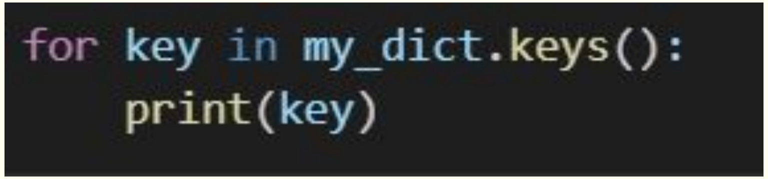
```
for value in my_dict.values():
    print(value)
```

- Loop through key-value pairs:



```
for key, value in my_dict.items():
    print(key, value)
```

- Access all keys:



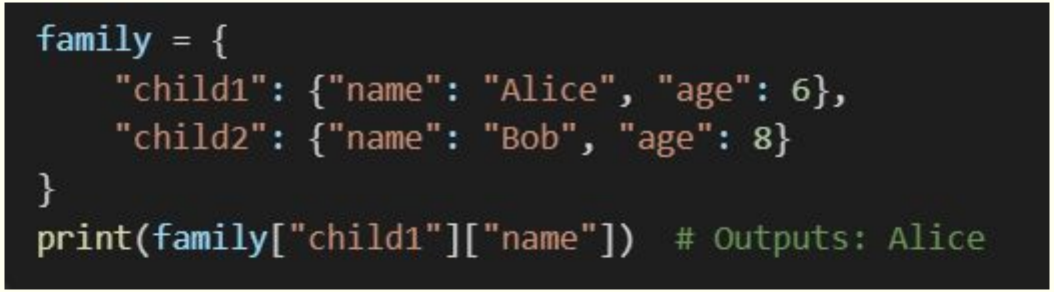
```
for key in my_dict.keys():
    print(key)
```

Practice Problem: Write a program to loop through a dictionary of student names and their marks, printing each name and mark.

Nested Dictionaries

- A dictionary can contain another dictionary as a value.

- Dictionary inside dictionary:

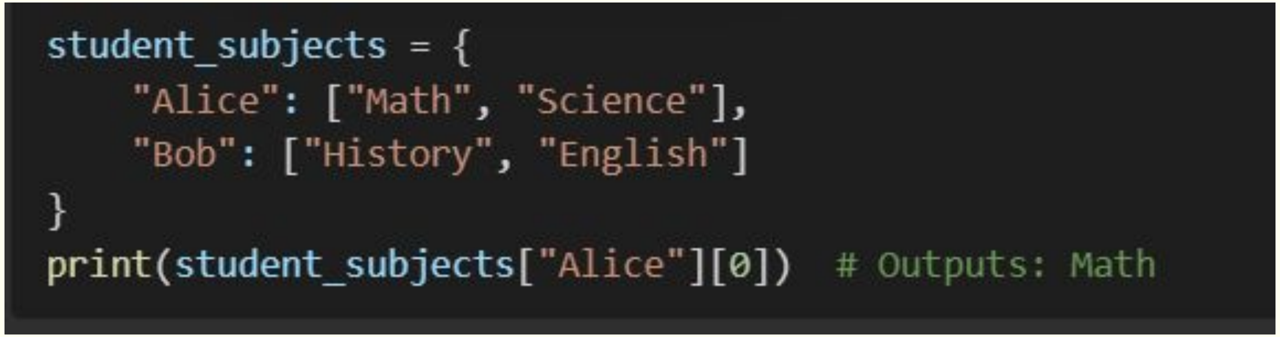


```python
family = {
    "child1": {"name": "Alice", "age": 6},
    "child2": {"name": "Bob", "age": 8}
}
print(family["child1"]["name"])  # Outputs: Alice
```

Practice Problem: Create a nested dictionary representing a class of students, where each student has a dictionary of their subjects and marks. Print one student's details.

List Inside a Dictionary

- You can store lists as values in a dictionary.

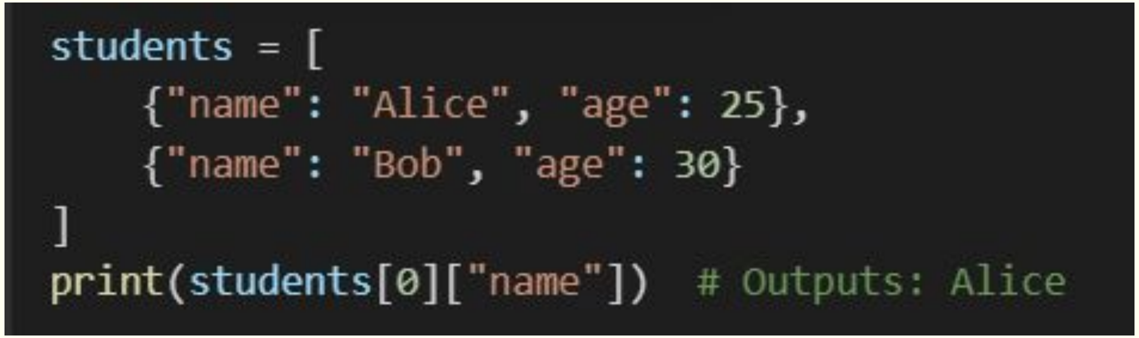


```python
student_subjects = {
    "Alice": ["Math", "Science"],
    "Bob": ["History", "English"]
}
print(student_subjects["Alice"][0])  # Outputs: Math
```

Practice Problem: Create a dictionary where keys are countries and values are lists of popular cities. Access a city from one country.

Dictionary Inside a List

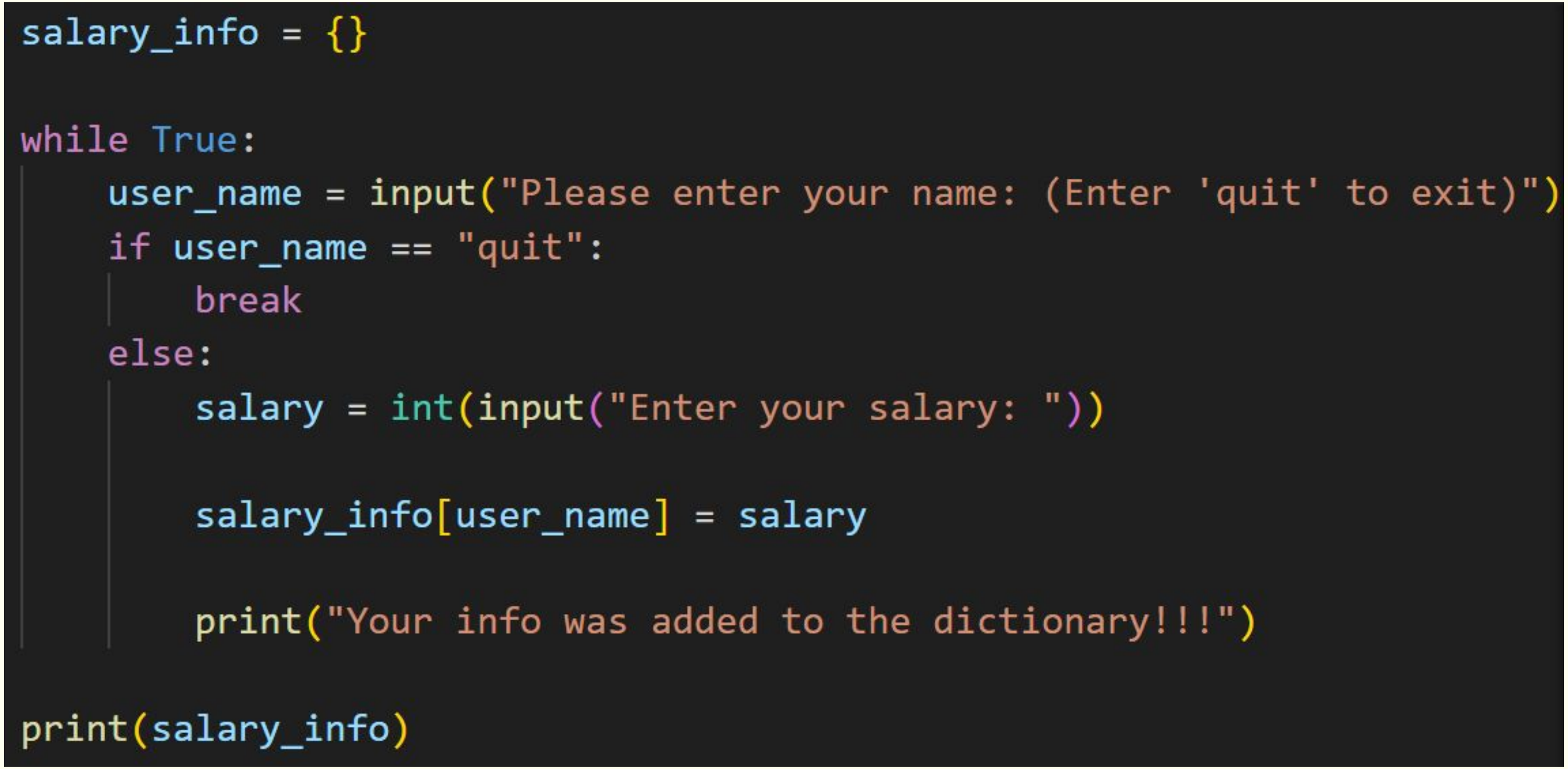
- A list can store multiple dictionaries.



```
students = [
    {"name": "Alice", "age": 25},
    {"name": "Bob", "age": 30}
]
print(students[0]["name"])  # Outputs: Alice
```

Practice Problem: Create a list of dictionaries, where each dictionary represents a book with its title and author. Access the author of the second book.

Creating a dictionary from scratch



```
salary_info = {}

while True:
    user_name = input("Please enter your name: (Enter 'quit' to exit)")
    if user_name == "quit":
        break
    else:
        salary = int(input("Enter your salary: "))

        salary_info[user_name] = salary

        print("Your info was added to the dictionary!!!")

print(salary_info)
```

Introduction to Sets

- A collection of unique and unordered elements.
- Useful for removing duplicates and performing mathematical set operations.

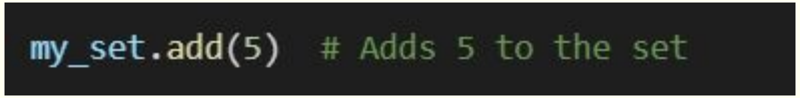
Syntax:

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

❖ Use set(list_name) to get unique values from a list.

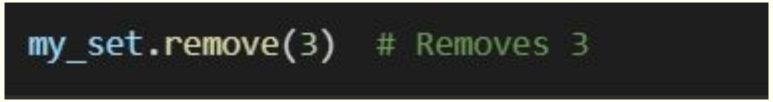
Set Operations

- Add items using `add()`:



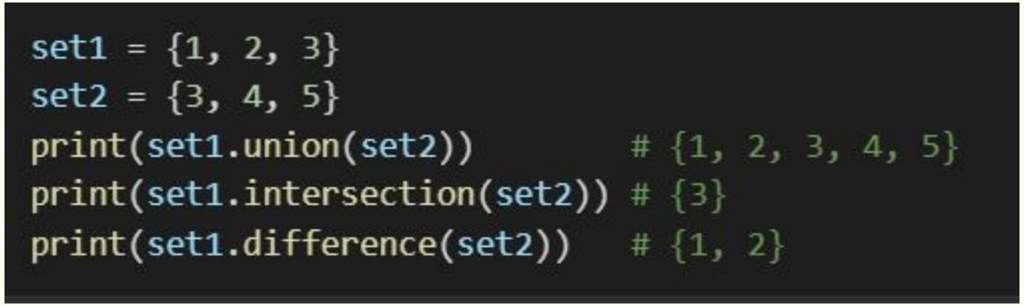
```python
my_set.add(5)  # Adds 5 to the set
```

- Remove items using `remove()` or `discard()`:



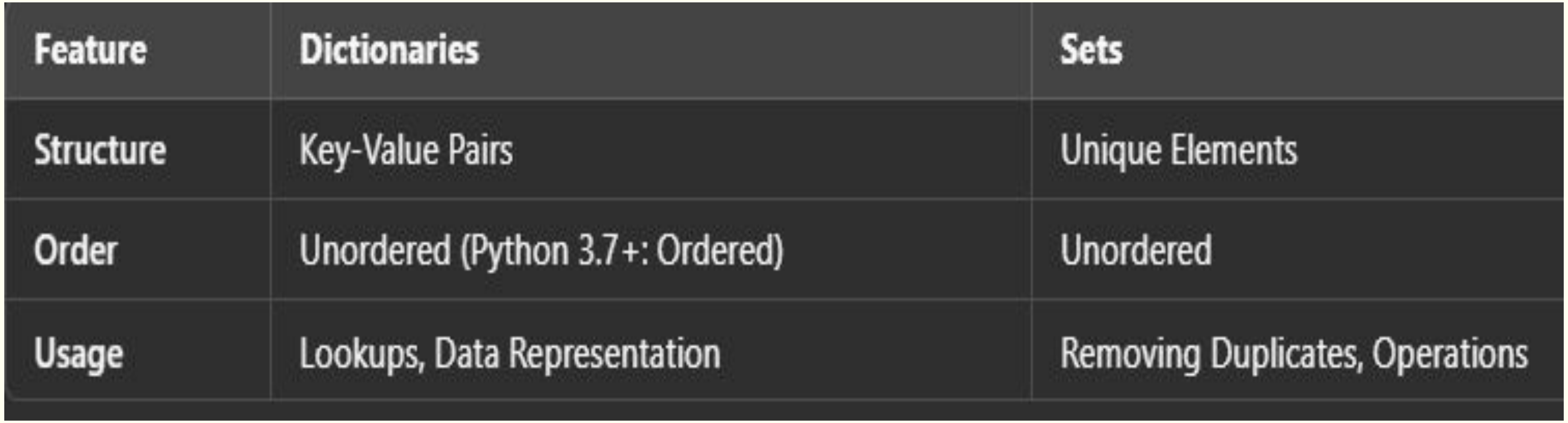
```python
my_set.remove(3)  # Removes 3
```

- Perform union, intersection, and difference:



```python
set1 = {1, 2, 3}
set2 = {3, 4, 5}
print(set1.union(set2))        # {1, 2, 3, 4, 5}
print(set1.intersection(set2)) # {3}
print(set1.difference(set2))   # {1, 2}
```

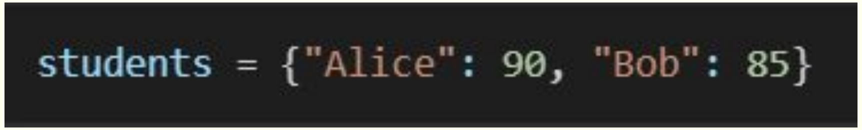
Comparing Dictionaries and Sets



Feature	Dictionaries	Sets
Structure	Key-Value Pairs	Unique Elements
Order	Unordered (Python 3.7+: Ordered)	Unordered
Usage	Lookups, Data Representation	Removing Duplicates, Operations

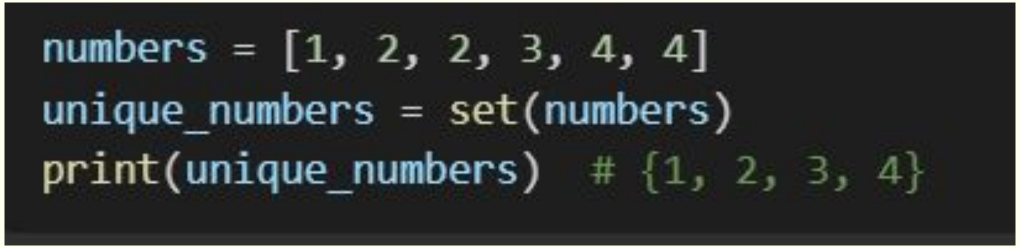
Real-Life Examples

- **Dictionaries:**
  - Storing student records:



```python
students = {"Alice": 90, "Bob": 85}
```

- **Sets:**
  - Removing duplicate items from a list:



```python
numbers = [1, 2, 2, 3, 4, 4]
unique_numbers = set(numbers)
print(unique_numbers)  # {1, 2, 3, 4}
```

Practice Problems

1. Create a dictionary to store names and ages of 3 friends. Add a new friend and their age.
2. Update the age of one friend and remove another friend from the dictionary.
3. Create a set of 5 numbers. Add two more numbers and remove one.
4. Write a program to find the union and intersection of two sets.
5. Loop through a dictionary to print each key-value pair.

Thank You

Do the Quiz Please, you have 10 minutes to do that!