**Advanced Data Structures**

* **Sets** are unordered, unique collections of elements, useful for ensuring no duplicates and fast membership testing.

**Example**

set1 = {1, 2, 3, 4}

set2 = {3, 4, 5, 6}

# Union (combine elements from both sets)

union\_set = set1 | set2

print(union\_set) # Output: {1, 2, 3, 4, 5, 6}

# Intersection (get common elements)

intersection\_set = set1 & set2

print(intersection\_set) # Output: {3, 4}

# Difference (get elements in set1 but not in set2)

difference\_set = set1 - set2

print(difference\_set) # Output: {1, 2}

# Symmetric Difference (elements in either set but not both)

symmetric\_diff\_set = set1 ^ set2

print(symmetric\_diff\_set) # Output: {1, 2, 5, 6}

* **Frozensets** are immutable versions of sets, suitable for use as dictionary keys or in places where immutability is required.

**Example**

fs1 = frozenset([1, 2, 3])

fs2 = frozenset([3, 4, 5])

# Union

print(fs1 | fs2) # Output: frozenset({1, 2, 3, 4, 5})

# Intersection

print(fs1 & fs2) # Output: frozenset({3})

* **Nested Data Structures** like lists of lists, dictionaries of lists, and lists of dictionaries allow you to organize and structure data in a more complex way, which is useful for representing real-world problems (e.g., matrices, employee records, etc.).

**List of Lists** (2D matrix):

matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

print(matrix[0]) # Output: [1, 2, 3]

print(matrix[1][2]) # Output: 6

**Dictionary of Lists**:

students = {

'John': [90, 85, 88],

'Sara': [92, 80, 95],

'Mike': [78, 84, 89]

}

print(students['John']) # Output: [90, 85, 88]

**List of Dictionaries:**

employees = [

{'name': 'Alice', 'age': 30, 'department': 'HR'},

{'name': 'Bob', 'age': 25, 'department': 'IT'},

{'name': 'Charlie', 'age': 35, 'department': 'Finance'}

]

print(employees[1]['name']) # Output: Bob