Detailed Data Structures Tutorial

1. Tuples

Introduction:

- Immutable and ordered collections.

- Syntax: tuple\_name = (item1, item2, ...)

- Use-cases: Fixed data items, function return values.

- Code: example\_tuple = (1, "hello", 3.14)

Basic Operations:

- Accessing: tuple\_name[index] -> first\_element = example\_tuple[0] # Returns 1

- Slicing: tuple\_name[start:end] -> slice\_tuple = example\_tuple[1:] # Returns ("hello", 3.14)

- Concatenation and repetition: concatenated = example\_tuple + (4, 5); repeated = example\_tuple \* 2

- Iteration: for item in example\_tuple: print(item)

Advanced Concepts:

- Tuple unpacking: a, b, c = example\_tuple

- Tuples as return values: def min\_max(numbers): return min(numbers), max(numbers)

Example Problems:

- Tuple creation and element access: my\_tuple = (10, 20, 30); second\_element = my\_tuple[1] # 20

2. Dictionaries

Introduction:

- Mutable, unordered key-value pairs.

- Syntax: dict\_name = {'key1': value1, 'key2': value2}

- Use-cases: Storing data items with keys.

- Code: example\_dict = {"name": "Alice", "age": 25}

Basic Operations:

- Accessing: dict\_name['key'] -> name = example\_dict["name"] # "Alice"

- Adding/updating: dict\_name['new\_key'] = value -> example\_dict["city"] = "New York"

- Deleting: del dict\_name['key'] -> del example\_dict["age"]

- Iteration: for key, value in example\_dict.items(): print(key, value)

Advanced Concepts:

- Dictionary comprehensions: squares = {x: x\*x for x in range(6)}

- Nested dictionaries: nested\_dict = {"first": {"id": 1, "name": "Alice"}, "second": {"id": 2, "name": "Bob"}}

Example Problems:

- Dictionary operations: my\_dict = {"fruit": "apple", "color": "red"}; my\_dict["taste"] = "sweet"

3. Sets

Introduction:

- Unordered collections of unique elements.

- Syntax: set\_name = {item1, item2, ...}

- Use-cases: Unique item collection, set operations.

- Code: example\_set = {1, 2, 3}

Basic Operations:

- Adding/removing: set\_name.add(item) / set\_name.remove(item) -> example\_set.add(4); example\_set.remove(2)

- Membership test: if 1 in example\_set: print("Found 1 in set")

- Size: len(set\_name) -> set\_size = len(example\_set)

Advanced Concepts:

- Set operations: another\_set = {3, 4, 5}; union\_set = example\_set | another\_set

- Set comprehensions: squared\_set = {x\*x for x in range(6)}

Example Problems:

- Set creation and manipulation: set1 = {1, 2, 3}; set2 = {3, 4, 5}; union\_result = set1 | set2 # {1, 2, 3, 4, 5}