**Chapter 5: Python Data Structures**

**Lecture Notes**

**Introduction:**

In this chapter, we will dive into various data structures that are fundamental for handling and organizing data. Understanding data structures is crucial for efficient data manipulation and analysis in Python. We will cover lists, tuples, sets, dictionaries, and arrays. Each data structure has its unique characteristics and operations, allowing us to store and access data in different ways. By mastering these concepts, you will gain the ability to effectively manage and manipulate data in Python.

**1. Lists and its Operations:**

Introduction to Lists:

Lists are one of the most versatile data structures in Python.

They can store a collection of elements of different data types.

Lists are ordered and mutable, allowing modification of elements.

Example:

my\_list = [1, 2, 3, 'apple', 'banana']

**List Operations:**

* Accessing elements: Using index values to access individual elements or slices of a list.
* Modifying elements: Changing the value of specific elements in a list.
* Adding elements: Appending, inserting, or extending a list with new elements.
* Removing elements: Removing elements by index or value.
* List methods: A variety of built-in methods for list manipulation, such as append(), sort(), reverse(), etc.

Example:

my\_list = [1, 2, 3]

print(my\_list[0]) # Output: 1

my\_list[1] = 'apple'

print(my\_list) # Output: [1, 'apple', 3]

my\_list.append('banana')

print(my\_list) # Output: [1, 'apple', 3, 'banana']

**2. Tuples and its Operations:**

Introduction to Tuples:

Tuples are similar to lists but are immutable, meaning their elements cannot be modified after creation.

They are often used to store related pieces of data together.

Example:

my\_tuple = (1, 2, 'apple', 'banana')

**Tuple Operations:**

* Accessing elements: Using index values to access individual elements or slices of a tuple.
* Tuple methods: A few built-in methods such as count() and index() are available for tuple manipulation.

Example:

my\_tuple = (1, 2, 'apple')

print(my\_tuple[2]) # Output: 'apple’

**3. Sets and its Operations:**

Introduction to Sets:

Sets are unordered collections of unique elements in Python.

They are useful for operations such as removing duplicates or testing membership.

Example:

my\_set = {1, 2, 3, 3, 'apple', 'banana'}

**Set Operations:**

* Adding elements: Using the add() method to add elements to a set.
* Removing elements: Using the remove() or discard() method to remove elements from a set.
* Set operations: Union, intersection, difference, and symmetric difference operations on sets.

Example:

my\_set = {1, 2, 3}

my\_set.add('apple')

print(my\_set) # Output: {1, 2, 3, 'apple'}

**4. Dictionaries and its Operations:**

Introduction to Dictionaries:

Dictionaries are key-value pairs that allow mapping one value (the key) to another (the value).

They are unordered and mutable data structures.

Example:

my\_dict = {'name': 'John', 'age': 25, 'city': 'New York'}

**Dictionary Operations:**

* Accessing elements: Retrieving values by their corresponding keys.
* Modifying elements: Changing the value associated with a specific key.
* Adding elements: Adding new key-value pairs to a dictionary.
* Removing elements: Removing key-value pairs from a dictionary.
* Dictionary methods: A variety of built-in methods for dictionary manipulation, such as keys(), values(), items(), etc.

Example:

my\_dict = {'name': 'John', 'age': 25}

print(my\_dict['name']) # Output: 'John'

my\_dict['age'] = 30

print(my\_dict) # Output: {'name': 'John', 'age': 30}

