Pre-Class Reading: Primitive & Abstract Data Structures in Python

1. Learning Objectives

In today's session, you will learn:

- The differences between primitive and abstract data structures.
- How data structures help organize and manage data in programming.
- How to use Python's basic data types and structures effectively.
- An introduction to the concept of abstract data structures and their importance.

By the end of the class, you'll understand which data structures to use for different programming tasks and why they're crucial in creating efficient code.

2. Key Concepts or Vocabulary

- **Primitive Data Structures**: Basic data types provided by Python, like integers, floats, strings, and booleans.
- **Abstract Data Structures (ADS)**: More complex data structures built on top of primitive types, like lists, stacks, queues, dictionaries, and trees.
- **Mutable vs. Immutable**: Mutable objects can be changed after creation (like lists), while immutable objects cannot be modified (like strings and tuples).
- **Array**: A data structure that stores elements in a contiguous block of memory (Python's `list` is a type of array).
- **Stack**: A Last In, First Out (LIFO) data structure; think of a stack of books where you can only access the top book.
- **Queue**: A First In, First Out (FIFO) data structure, like a line of people waiting in a queue.
- **Dictionary**: A data structure that holds key-value pairs, enabling efficient data retrieval based on keys.

3. Real-World Examples

1. **Primitive Data Structures**: Think of primitive data types as the foundation of a building. Numbers, characters, and booleans are like the bricks and beams used to construct everything else.
2. **Abstract Data Structures**:
   - **Lists**: Consider a grocery list. You add items to it, and you may remove items as you buy them. Python lists work similarly, allowing us to add, remove, or update elements.
   - **Stacks**: Imagine stacking plates in a cupboard. The last plate you place on the stack is the first one you take off—a stack is exactly like this.
   - **Queues**: Picture standing in line at a coffee shop. The first person to enter the line is the first to be served, just like how a queue functions.
   - **Dictionaries**: Think of a phone book where you look up a person's phone number by their name. Dictionaries work the same way; they allow us to retrieve values (like phone numbers) by using keys (like names).

4. Mini Code Snippets

Primitive Data Types

```
# Integer (whole number)
age = 25  # Age of a person

# Float (decimal number)
height = 5.9  # Height in feet

# String (text)
name = "Alice"  # Person's name

# Boolean (True/False)
is_student = True  # Indicates if the person is a student
```

Abstract Data Structures

List

```
# A list of favorite fruits
fruits = ["apple", "banana", "cherry"]

# Adding an item to the list
fruits.append("orange")  # fruits becomes ["apple", "banana", "cherry", "orange"]

# Accessing an item
print(fruits[1])  # Outputs "banana"
```

Stack (using a list)

```
# A simple stack of books
stack = []

# Adding books to the stack
stack.append("Book 1")
stack.append("Book 2")

# Removing the top book
top_book = stack.pop()  # "Book 2" is removed
```

Queue (using a list with collections.deque for efficiency)

```
from collections import deque

# A queue of people
queue = deque(["Alice", "Bob", "Charlie"])

# Adding a person to the end of the queue
queue.append("David")

# Removing the person at the front of the queue
first_person = queue.popleft()  # "Alice" is removed
```

Dictionary

```python
# A dictionary of items with their prices
prices = {"apple": 0.5, "banana": 0.3, "cherry": 0.7}

# Accessing a price by item
print(prices["apple"])  # Outputs 0.5
```

5. Pre-Reading Questions or Simple Exercises

1. **Identify Data Types**: Look around your room and identify three objects. Try to classify these as different data types. For example, “chair” could be a string, and the number of chairs could be an integer.
2. **Mini Coding Task**: Write a list of three favorite movies and add a fourth movie to it. Print out the updated list.
3. **Explore Dictionaries**: Think of two things you can categorize by a unique attribute, like “people and their favorite colors.” Try creating a dictionary in Python where you assign each person a favorite color.

6. Common Mistakes or Misconceptions

- **Confusing Lists and Arrays**: In many programming languages, lists and arrays are distinct, but in Python, the `list` type behaves similarly to arrays in other languages.
- **Misusing Mutable vs. Immutable Types**: Beginners often try to modify immutable types, like strings or tuples. Remember, you cannot change the contents of these objects directly—Python will throw an error.
- **Using Wrong Data Structures**: Sometimes students try to use a list as a stack or queue but forget to follow LIFO/FIFO rules. Remember, stacks are LIFO, and queues are FIFO.

7. A Teaser for the Lecture

Why do some data structures let you access elements faster than others? And why is that speed crucial for real-world applications? Tomorrow, we'll dive deeper into how different data structures affect the speed and efficiency of programs, and how to pick the right one for any coding task.

Looking forward to helping you build a solid foundation in Python data structures!