3. Core Python Concepts **Theory**:

❖ Understanding data types: integers, floats, strings, lists, tuples, dictionaries, sets.

1. Integers (int)

   Whole numbers without decimals.

   Syntax : age = 25

2. Floats (float)

   Numbers with decimal points.

   Syntax : temperature = 36.6

3. Strings (str)

   Sequences of characters enclosed in quotes.

   Syntax : greeting = "Hello, World!"

4. Lists (list)

Ordered, mutable collections of items.

Syntax : fruits = ["apple", "banana", "cherry"]

Key Operations:

- append(): Adds an item to the end.
- remove(): Removes the first occurrence of an item.
- pop(): Removes and returns an item at a given index.
- sort(): Sorts the list in ascending order.

5. Tuples (tuple)

Ordered, immutable collections of items.

Syntax : coordinates = (10.0, 20.0)

Key Operations:

- Indexing and slicing: Access elements using indices.
- Unpacking: Assign tuple elements to variables.

6. Dictionaries (dict)

Unordered collections of key-value pairs.

Syntax : person = {"name": "Alice", "age": 30}

Key Operations:

- get(): Retrieves the value associated with a key.
- keys(): Returns all keys.
- values(): Returns all values.
- items(): Returns all key-value pairs.

7. Sets (set)

Unordered collections of unique items.

Syntax : unique_numbers = {1, 2, 3, 4}

Key Operations:

- add(): Adds an item.
- remove(): Removes an item.
- union(): Returns a set containing all items from both sets.
- intersection(): Returns a set containing items present in both sets.

❖ Python variables and memory allocation.

1. Variables

Unlike languages like C or Java, Python doesn't have traditional variables. Instead, it uses names that refer to objects in memory. These names can be reassigned to different objects, and multiple names can refer to the same object. This dynamic typing allows flexibility but requires careful memory management.

2. Memory Allocation: Stack vs. Heap

Python employs two primary areas for memory allocation:

- **Stack Memory**: Used for function calls and local variables. It's managed automatically and has a limited size.
- **Heap Memory**: Used for objects and data structures. Python's memory manager handles allocation and deallocation here. The heap is managed through a private heap space, and developers don't have direct access to it.

❖ Python operators: arithmetic, comparison, logical, bitwise.

1. Arithmetic Operators

These operators perform mathematical operations:

- **+** : Addition
- **-** : Subtraction
- ***** : Multiplication
- **/** : Division
- **//** : Floor Division
- **%** : Modulus
- ****** : Exponentiation

2. Comparison Operators

These operators compare two values and return a boolean result:

- **==** : Equal to

- **!=** : Not equal to
- **>** : Greater than
- **<** : Less than
- **>=** : Greater than or equal to
- **<=** : Less than or equal to
- **is** : True if both variables point to the same object
- **is not** : True if both variables do not point to the same object
- **in** : True if a value is found in the specified sequence
- **not in** : True if a value is not found in the specified sequence

3**. Logical Operators**

These operators are used to combine conditional statements:

- **and** : Returns True if both statements are true
- **or** : Returns True if one of the statements is true

- **not** : Returns True if the statement is false

4. Bitwise Operators

These operators perform bit-level operations:

- **&** : Bitwise AND
- **|** : Bitwise OR
- **^** : Bitwise XOR
- **~** : Bitwise NOT
- **<<** : Left shift
- **>>** : Right shift