3. Core Python Concepts.

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

1. **Integers(int)**

-Whole numbers (positive, negative, or zero).
-Immutable: Cannot be changed after creation.
-Used for counting, indexing, math operations without decimals.

Ex.

```
age = 20
temperature = -10
```

2. **Floats (float)**

- Numbers with decimal points (real numbers).
- Useful for precision math, measurements, percentages.

Ex.

```
price = 19.99
pi = 3.14159
```

3. **Strings (str)**

- Text data inside quotes (' or ").
- Ordered sequence of characters (text).

Ex.

```
name = "Umang"
greeting = 'Hello'
```

4. **Lists (list)**

-Ordered, mutable (changeable) sequence of elements.
-Storing collections of items that may change.

Ex.

```
fruits = ["apple", "banana", "cherry"]
mixed_list = [1, "hello", 3.14, True]
```

5. **Tuples (tuple)**

- Ordered, immutable (unchangeable) sequence of elements.

Ex.

```
coordinates = (10, 20)
colors = ("red", "green", "blue")
```

6. **Dictionaries (dict)**

- Unordered collection of key-value pairs.
- storing structure data

Ex.

```
person = {"name": "Umang", "age": 20, "city": "Gir-Somnath"}
```

7. **Sets (set)**

- Unordered collections of unique items..

Ex.

```
numbers = {1, 2, 3, 2, 1}
```

- **Python variables and memory allocation.**

- Variables in Python are references to objects in memory.
- A variable is a name bound an object in memory.
- A variable is simply a name that refers to a value stored in memory.

```
x = 10
```

x is the **variable name**.

10 is the **value**.

Memory allocation

- Memory allocation is the process of assigning space in memory to store data during program execution.
- In Python, memory allocation happens automatically when a variable is created — Python reserves memory and handles it internally.

- **Python operators: arithmetic, comparison, logical, bitwise**

1. Arithmetic Operators

+ Addition

- Subtraction

***** Multiplication

/ Division

// Floor Division

% Modulus

****** Exponentiation

2. Comparison (Relational) Operators

== Equal to

!= Not equal to

> Greater than

< Less than

>= Greater than or equal

<= Less than or equal

3. Logical Operators

and Returns True if both are true

or Returns True if at least one is true

not Reverses the boolean value

4. Bitwise Operators

& Bitwise AND

^ Bitwise XOR (exclusive OR)

~ Bitwise NOT (inverts bits)

<< Left shift

>> Right shift

`