1. Variables (foundation stone)

A variable is a named storage location that holds a value which can change during program execution.

Think of it as:

a label stuck on a box in memory.

Core anatomy of a variable

Every variable has:

- Name – how you refer to it
- Value – what it stores right now
- Type – what kind of data it can store
- Scope – where it can be used
- Lifetime – how long it exists

Example (Java):

int age = 21;

- name → age
- type → int
- value → 21

Same idea in Python / JS, different rules about types.

———

2. Variable Types (data types)

Primitive / basic types (most important for prep)

These store single values.

Integer types
Used for whole numbers.

```
int a = 10;
long b = 1000000000L;
```

Use when:

- counting
- indexing arrays
- loop counters

Floating-point types
Used for decimals.

```
float x = 3.14f;
double y = 3.14159;
```

⚠️ Floating numbers are approximate, not exact.
Never compare them directly using == in serious code.

Character type
Stores one character.

char c = 'A';

Internally stored as a number (ASCII / Unicode).
This matters in DSA problems.

Boolean type
Stores only true or false.

boolean isValid = true;

This is the fuel for conditionals and loops.

———

Non-primitive / reference types (conceptual view)

These store references (addresses), not raw values.

Examples:

- String
- arrays
- objects
- lists

String name = "Dhanush";
int[] arr = {1,2,3};

Important interview line:

Primitives store values, references store locations.

———

3. Operators (how values interact)

An operator performs an action on variables or values.

Think of operators as verbs.
Variables are nouns.

———

4. Types of Operators (very exam-heavy)

1️⃣ Arithmetic Operators

Used for calculations.

Operator	Meaning
+	addition
-	subtraction
*	multiplication
/	division
%	modulus (remainder)

int a = 10;
int b = 3;

a + b → 13
a / b → 3 // integer division
a % b → 1

⚠️ Interview trap:

5 / 2 = 2 // not 2.5

2️⃣ Assignment Operators

Used to store or update values.

int x = 5;
x += 3; // x = x + 3 → 8
x -= 2; // x = x - 2 → 6

Shortcut operators improve readability and speed.

3️⃣ Relational (Comparison) Operators

They compare values and return boolean.

Operator	Meaning
==	equal
!=	not equal
>	greater
<	less
>=	greater or equal
<=	less or equal

int a = 10;
int b = 5;

a > b → true

a == b → false

⚠️ Critical:
= assigns
== compares

———

4️⃣ Logical Operators

Used to combine conditions.

Operator	Meaning
&&	AND
`	
!	NOT

age >= 18 && hasID == true

Short-circuit behavior (very important):

- false && anything → stops immediately
- true || anything → stops immediately

Interviewers love this.

———

5️⃣ Unary Operators

Operate on one operand.

```
int x = 5;
x++;   // post-increment
++x;   // pre-increment
```

Difference:

- x++ → use first, then increment
- ++x → increment first, then use

Classic MCQ trap.

———

6. Operator Precedence (silent killer)

Order of execution:

1. ()
2. Unary (++, --, !)
3. * / %
4. + -
5. Relational
6. Logical

7. Assignment

```
int x = 10 + 2 * 5; // 20, not 60
```

---

7. One-line mental model (revision gold)

- Variables store state
- Types restrict what can be stored
- Operators change or compare state
- Booleans drive decisions