In many programming languages, data types are broadly categorized into primitives and objects, each with distinct characteristics and behaviors.

Primitives:

- **Simple Values:**

  Primitives represent the most basic, fundamental data types and typically hold a single, atomic value. Examples include numbers (integers, floats), booleans (true/false), characters, and in some languages, strings.

- **Immutable:**

  Once a primitive value is created, it cannot be changed. Any operation that appears to modify a primitive actually creates a new primitive with the updated value.

- **Passed by Value:**

  When a primitive is passed to a function or assigned to another variable, a copy of its value is made. Changes to the copy do not affect the original.

- **Stored on the Stack:**
  Primitives are often stored directly on the call stack, which is faster for access and management.

Objects:

- **Complex Structures:**

  Objects are more complex data types that can encapsulate multiple values (properties) and associated functions (methods). They are instances of classes, which serve as blueprints.

- **Mutable:**

  Objects are generally mutable, meaning their internal state (properties) can be changed after creation.

- **Passed by Reference:**

  When an object is passed to a function or assigned to another variable, a reference (or pointer) to the original object in memory is passed. Modifications made through the reference directly affect the original object.

- **Stored on the Heap:**

Objects are typically stored in the heap memory, a more flexible area for dynamic memory allocation, and variables hold references to these locations.

Key Differences Summarized:

Feature	Primitives	Objects
Data Structure	Simple, atomic values	Complex, composed of properties and methods
Mutability	Immutable (cannot be changed)	Mutable (can be changed)
Passing	Passed by value (copy of value)	Passed by reference (reference to memory)
Memory	Stored on the stack	Stored on the heap (references on stack)