1. Data types in Solidity:

* Value type
* Reference type

1. Value Type: It holds data within the memory owned by it (same as location of the variable). Max size allowed is 32bytes. When assigned to another variable or sent as an argument in a function, a new variable instance is created and the value is copied into it. Both the variables will be independent of each other.

* bool
* uint
* int
* address
* byte
* enum

1. Reference type: Values are stored separately. The variable only stores the address/pointer of the memory location. They can acquire value of more than 32bytes. When assigned to another variable or sent as an argument in another function, the pointer is copied to new variable rather than the value. Change in original or new variable will change the value in both the variables (as both point to the same memory location).

* arrays
* structs
* string
* mappings

1. Storage and Memory Data Location:

Variables can either have storage data location or memory data location, depending on its use. Storage variables are costly as it occupies permanent space on the EVM. Some comparative features are listed below:

|  |  |
| --- | --- |
| **Storage** | **Memory** |
| Global | Within Function |
| Permanent | Temporary |
| State Variable | Function Parameter |
| Expensive | Cheaper |

Data location depends on the following two factors:

* 1. Location of variable declaration
  2. Data type of variable (i.e., Value type or reference type)

1. Rules for data location of variables:
   1. Rule 1:

State variables, whether value type or reference type, are stored in storage data location. No need to mention storage or memory while declaring state variables.

* 1. Rule 2:

Variables declared as function parameters including return parameters are always stored in memory data location. Memory data location needs to be specified for reference data types to avoid compilation error. Not applicable for mapping as mapping cannot be declared as function parameter.

* 1. Rule 3: (Declarations within a function)

Value type variables declared within a function are always stored in memory (memory data location) and cannot be stored as storage.

Reference type variables require explicit specification while declaration as to whether the variable is to be stored in storage or memory. Furthermore, Reference type variables stored as storage should always point to a state variable therefore, value cannot be assigned directly and it only creates a reference to state variable (does not create a copy). Mapping cannot be declared in a function, however, mappings in a function can refer to mappings declared as state variables.

* 1. Rule 4:

Arguments supplied by callers to function parameters are always stored in calldata data location.

**Assignment Behaviour**

* 1. Rule 5:

Assignment to state variable from another state variable always creates a new copy whether it is a value type or reference type variable.

* 1. Rule 6:

Assignment of memory variable to state variable always creates a new copy whether it is a value type or reference type variable.

* 1. Rule 7:

Assignment of state variable to a memory variable always creates a new copy whether it ia value type or reference type variable.

* 1. Rule 8:

Assignment from one memory variable to another memory variable:

* Creates a copy for value type variables
* Do not create a copy for reference type variables

1. Bytes:

Refers to 8bit signed integers. Everything is stored in bits consisting of binary values 0 and 1.

Ranges from bytes1 to bytes32. It cannot directly store integers. Need to convert to binary first.

1. Arrays:

Arrays can have a compile time fixed (pre-determined) size or they can be dynamic.