1. Solidity is a domain-specific language of choice for programming contracts in Ethereum. Other features of the language include inheritance, libraries, and the ability to define composite data types.
2. It is a statically typed language, which means that variable type checking in solidity is carried out at compile time. Each variable, either state or local, must be specified with a type at compile time. This is beneficial in the sense that any validation and checking is completed at compile time and certain types of bugs, such as interpretation of data types, can be caught earlier in the development cycle instead of at run time, which could be costly.
3. Solidity has two categories of data types: value types and reference types.
4. Data Types –
   * + Boolean ---This data type has two possible values, true or false, for example: bool v = true; This statement assigns the value true to v.
     + String literals specify a set of characters written with double or single quotes.
     + Hexadecimal literals Hexadecimal literals are prefixed with the keyword hex and specified within double or single quotation marks
     + Enums - This allows the creation of user-defined types.
5. Functions –
6. Internal -These can be used only within the context of the current contract.
7. External functions External functions can be called via external function calls. A function in solidity can be marked as a constant. Constant functions cannot change anything in the contract; they only return values when they are invoked and do not cost any gas
8. Reference types As the name suggests, these types are passed by reference
9. Arrays -----Arrays represent a contiguous set of elements of the same size and type laid out at a memory location. The concept is the same as any other programming language. Arrays have two members named length and push:
10. Structs--- These constructs can be used to group a set of dissimilar data types under a logical group. These can be used to define new types.
11. **Data location** --- the location can be storage or memory. This is applicable to arrays and structs and can be specified using the storage or memory keywords. **Calldata is another memory location** that is used to **store function arguments**.. By default, parameters of functions are stored in memory, whereas all other **local variables & State variables, on the other hand, are required to use storage.**
12. Mappings are used for a key to value mapping. This is a way to associate a value with a key,

mapping (string => uint) bids;

bids["packt"] = 10;

**This is basically a dictionary or a hash table where string values are mapped to integer**

**values. The mapping named bids has a packt string value mapped to value 10.**

1. Global Variables – please read- <https://www.bitdegree.org/learn/solidity-units-and-global-variables/>
2. **Solidity provides functions and special variables that always exist inside the global namespace, are mainly used for providing information about the blockchain itself.**

* **Example** - block.number (uint): number of the current block
* block.gaslimit (uint): gaslimit of the current block
* block.timestamp (uint): timestamp of the current block as seconds passed since the unix epoch
* msg.gas (uint): gas that remains
* msg.data (bytes): complete calldata

1. Control structures available in solidity are if - else, do, while, for, break, continue, return. They work in a manner similar to how they work in C-language or JavaScript.
2. Events in solidity can be used to log certain events in EVM logs. These are quite useful when external interfaces are required to be notified of any change or event in the contract. These logs are stored on the blockchain in transaction logs, LEARN MORE ON EVENTS.
3. Inheritance is supported in solidity. The is keyword is used to derive a contract from another contract, this doesn’t work for private state variables.

- EXAMPLE -

Contract 1 is Contract2 {

}