Solidity – Error Handling

There are many functions for error handling in Solidity. Errors can occur at compile time or runtime. Some of the runtime errors are out-of-gas error, data type overflow error, divide by zero error, array-out-of-index error, etc. There are three error statements, which are, **Require statements, Assert statements, Revert statements.**

**Require statements**: It declares prerequisites for running the function i.e. it declares the constraints which should be satisfied before executing the code. It accepts a single argument and returns a Boolean value after evaluation; it also has a custom string message option. If false then exception is raised and execution is terminated.

Example: require(Condition/expression, "Message/condition explanation");

**Assert statements:** Its syntax is similar to the require statement. It returns a Boolean value after the evaluation of the condition. Based on the return value either the program will continue its execution or it will throw an exception. Assert is used to check the current state and function conditions before the execution of the contract.

Example: assert (*condition/expression*);

**Revert statements:** This statement is similar to the require statement. It does not evaluate any condition and does not depend on any state or statement. It is used to generate exceptions, display errors, and revert the function call. This statement contains a string message which indicates the issue related to the information of the exception. Revert is used to handle the same exception types as require handles, but with little bit more complex logic.

Example: uint sum = \_num1 + \_num2;

         if(sum < 0 || sum > 255){

             revert(" Overflow Exist");