Iterators

→ What is an Iterator ?

It is an object that allows us to traverse over a list or collection. Iterators' purpose is to define the sequences and implement the iterator protocol that returns an object by using a next() method that contains the value and done.

→ DONE: It is a boolean value indicating whether any more elements in the sequence could be iterated upon.

→ VALUE: It is the current element of the sequence.

So, we can define iterators as an "object that knows how to ~~acco~~ access items from a collection one at a time, while keeping track of its current position within that sequence".

→ Suppose, we have an array, and it

contains five numbers, i.e., [1,2,3,4,5].
As we know, the iterator object has a next() method that returns the next item in the sequence. So, when we write next(), we will get the element of the array. The next() method returns an object with two properties: value and done. If there are elements present in the sequence that could be iterated, then the value contains the next element and done is set to false.
{ VALUE: 'NEXT DONE', DONE: FALSE }
If we call the next() method after the last value has been returned, then the next() returns the result object as following:
{ DONE: TRUE, VALUE: UNDEFINED }
Here the value of done property, which is true, indicates that there is no more value to return, and the value of the property is set to undefined.

Example

```
FUNCTION MYITERABLE() {
    LET COUNTER = 0;
    RETURN {
        NEXT: FUNCTION() {
            IF (COUNTER < 5) {
                COUNTER++;
```

Page No.
Date

```
        RETURN {DONE: FALSE, VALUE: counter};
      } ELSE {
        RETURN {DONE: TRUE, VALUE: UNDEFINED};
      }
    }
  }
}
```

CODE EXPLANATION:

The above code executes five steps, with the counter incrementing (counter++) every turn. First, we return the value 1, then the value 2, and so on till we get the last element 5, then we indicate that the end of the iteration has been reached, and the value becomes equal to undefined.