1. square bracket refers to list and parenthesis refers to tuples.

2. Iterator

An iterator is an object which contains a countable number of values and it is used to iterate over iterable objects like list, tuples, sets, etc. Iterators are implemented using a class and a local variable for iterating is not required here, It follows lazy evaluation where the evaluation of the expression will be on hold and stored in the memory until the item is called specifically which helps us to avoid repeated evaluation. As lazy evaluation is implemented, it requires only 1 memory location to process the value and when we are using a large dataset then, wastage of RAM space will be reduced the need to load the entire dataset at the same time will not be there.

Using an iterator-

iter() keyword is used to create an iterator containing an iterable object.

next() keyword is used to call the next element in the iterable object.

After the iterable object is completed, to use them again reassign them to the same object.

Generators

It is another way of creating iterators in a simple way where it uses the keyword “yield” instead of returning it in a defined function. Generators are implemented using a function. Just as iterators, generators also follow lazy evaluation. Here, the yield function returns the data without affecting or exiting the function. It will return a sequence of data in an iterable format where we need to iterate over the sequence to use the data as they won’t store the entire sequence in the memory.

3. If a function contains at least one yield statement (it may contain other yield or return statements), it becomes a generator function.

4. In its simplest form, a yield statement looks much like a return statement, except that instead of stopping execution of the function and returning, yield instead provides a value to the code looping over the generator and pauses execution of the generator function.

5. Map VS List Comprehension

List comprehension is more concise and easier to read as compared to map.

List comprehension allows filtering. In map, we have no such facility. For example, to print all even numbers in range of 100, we can write [n for n in range(100) if n%2 == 0]. There is no alternate for it in map

List comprehension are used when a list of results is required as map only returns a map object and does not return any list.

List comprehension is faster than map when we need to evaluate expressions that are too long or complicated to express

Map is faster in case of calling an already defined function (as no lambda is required).