1)what is a difference between enclosing a list comprehension in square bracket and parentheses ?

Ans : The difference between the two kinds of expressions is that the List comprehension is enclosed in square brackets [] while the Generator expression is enclosed in plain parentheses ().

l = [n\*2 for n in range(1000)] # List comprehension

g = (n\*2 for n in range(1000)) # Generator expression.

2)what is the relationship between generators & iterators ?

Ans :

| Iterators | Generators |
| --- | --- |
| Class is used to implement an iterator | Function is used to implement a generator. |
| Local Variables aren’t used here. | All the local variables before the yield function are stored. |
| Iterators are used mostly to iterate or convert other objects to an iterator using iter() function | Generators are mostly used in loops to generate an iterator by returning all the values in the loop without affecting the iteration of the loop. |
| Iterator uses iter() and next() functions. | Generator uses yield keyword. |

3)what are the signs that a function is a generators functions ?

Ans : A generator is a special type of function which does not return a single value, instead, it returns an iterator object with a sequence of values. In a generator function, a yield statement is used rather than a return statement. The following is a simple generator function.

4)what is the purpose of a yield statement ?

Ans : The yield statement returns a generator object to the one who calls the function which contains yield, instead of simply returning a value.

5)what is the relationship between map calls & list comprehensions ?make a comparison and contrast between the two.

Ans : we want to compute this function for different values in a single line of code . This is where map() function plays its role ! map() function returns a map object(which is an iterator) of the results after applying the given function to each item of a given iterable (list, tuple etc.)

Syntax: map(funcname, iterables)

Parameters:

funcname: It is the name of the function which is already defined and is to be executed for each item.

iterables: It can be list, tuples or any other iterable object.

Return Type: Returns a map object after applying the given function to each item of a given iterable (list, tuple etc.)

Example:

# function to double the number

def num (n) :

return n \* 2

lst = [2, 44, 5.5, 6, -7]

suppose we want to call function

# 'num' for each element of list,

# we use map

# creates a map object

x = map(num, lst)

print(x)

print(list(x))

Output:

<map object at 0x7f859f3f05c0>

[4, 88, 11.0, 12, -14]

List Comprehension is a substitute for the lambda function, map(), filter() and reduce(). It follows the form of the mathematical set-builder notation. It provide a concise way to create lists.

Syntax:

[ expression for item in list if conditional ]

Parameters:

Expression – based on the variable used for each element

for ..in – ‘for’ followed by the variable name to use, followed by ‘in’

if – to filter.

Example:

lst = [2, 44, 5.5, 6, -7]

# to double the number

# list comprehension

X= [i \* 2 for i in list ]

print(x)

Output:

[4, 88, 11.0, 12, -14]