1) . What is the difference between enclosing a list comprehension in square brackets and parentheses?

Answer: Square brackets are lists while parentheses are tuples. A list is mutable, meaning you can change its contents. while tuples are not.

The other main difference is that a tuple is hashable, meaning that you can use it as a key to a dictionary, among other things.

2) What is the relationship between generators and iterators?

Answer: A python generator is a subclass of iterator.

```
import collections, types
issubclass(types.GeneratorType, collections.Iterator)

OUTPUT >>> True
```

3) What are the signs that a function is a generator function?

Answer: A generator function will contain the yield keyword which saves the current state of the function and its variables.

Execution of the function doesnt return a value directly unless the result is parsed with next function or converted into a list after parsing with next.

4) What is the purpose of a yield statement?

Answer: Yield is a keyword in Python that is used to return from a function without destroying the states of its local variable and when the function is called, the execution starts from the last yield statement. Any function that contains a yield keyword is termed a generator.

5) What is the relationship between map calls and list comprehensions? Make a comparison and contrast between the two.

Answer: The relationship between map and list comprehension are as follows :-

- List comprehension is more concise and easier to read then map
- list comprehension allows filtering, whereas in map we cannot.
- List comprehension is used when a list of result is required, and map only returns a map object and does not return a list
- List comprehension is faster then map when we would like to evaluate long expressions that are complicated
- Map is faster in case of calling an already declared function

List comprehension example:

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

x = [i * 2 for i in lst ]
print(x)

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

Map comprehension example:

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

def num (n) :
   return n * 2

x = map(num, lst)
print(x)
print(list(x))


OUTPUT >>> <map object at 0x0000017C98785FD0>
[4, 88, 11.0, 12, -14]
```