Q1.What is Iterator protocol?

Soln.

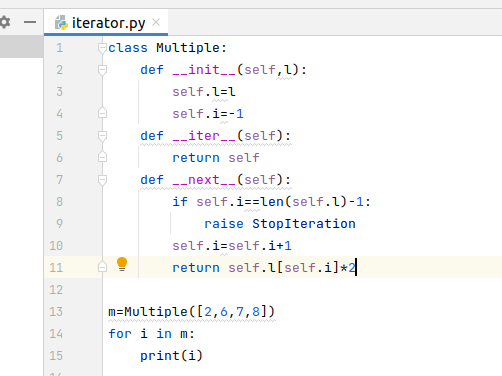
**iterator** **protocol** consist of the methods **\_\_iter\_\_() and \_\_next\_\_().**

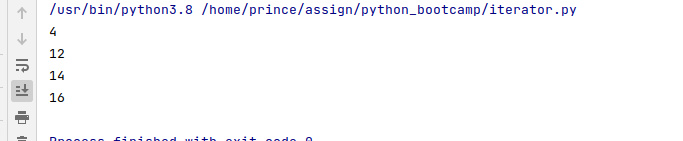
**The \_\_iter\_\_() method** always return the iterator object itself.

**The \_\_next\_\_() method** also allows you to do operations, and must return the next item in the sequence.

Q2.Write a custom Iterator to multiple elements of input array (all numeric) with 2 (i.e. [2,6,7,8] will be [4,12,14,16]

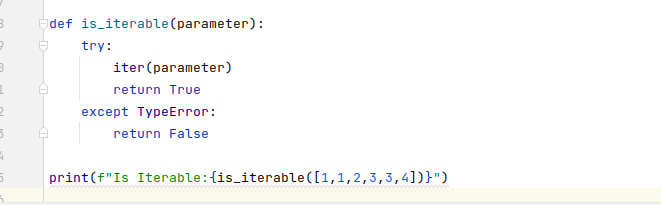
Soln.





Q3.Write a function to check if an object is iterable or not.

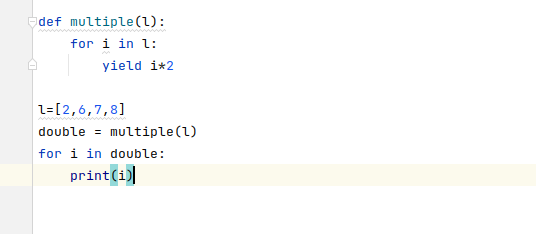
Soln.

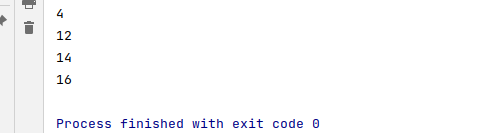




Q4.Rewrite 2. problem using "yield" keyword

Soln.





Q5.What benefit we get from using Iterators?

Soln.

1. Cleaner code
2. Iterators can work with infinite sequences
3. Iterators save resources

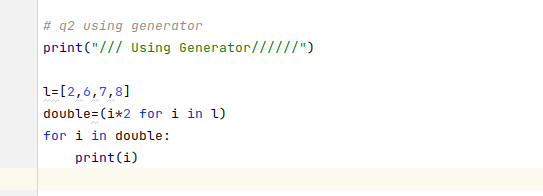
Q6.What are the limitations of Iterators?

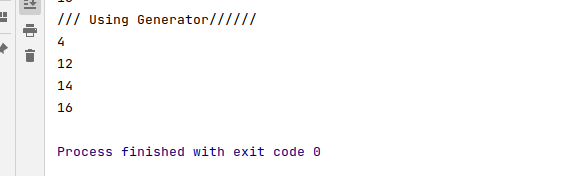
Soln.

If we need to backtrack while processing through a list, then iterators may not work at all.

Q7.Rewrite 2. problem using "Generator expression"

Soln.



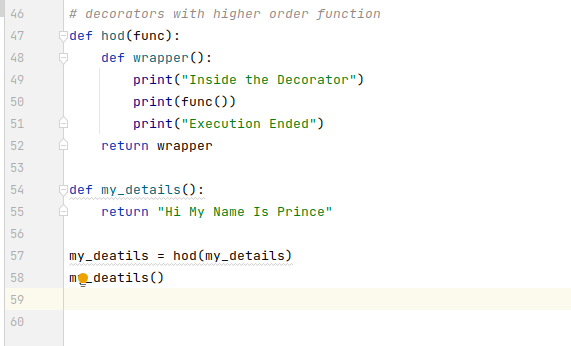


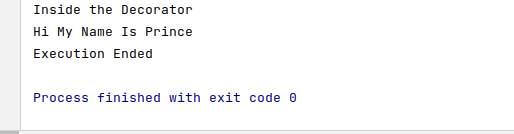
Q8.Explain how Decorator can be used as a higher order function?

Soln.

A function is called **Higher Order Function** if it contains other functions as a parameter or returns a function as an output.

**Decorators** allow us to wrap another function in order to extend the behavior of **wrapped** function, without permanently modifying it.

****



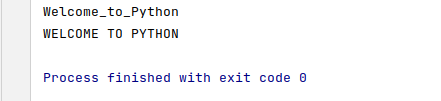
Q9.Implement 2 decorators upon below function (1. To replace underscores with single space, 2. To convert all letters to Capital case)

def greet():

return 'Welcome to Python'

Soln.





Q10.Convert a python list [77, 88, 44, 33] to [7, 8, 4, 3] using

A. Map

B. List comprehension

Soln.

