Example1

a\_0=[1,2,3]  
a\_1=[1,2,3]  
print(a\_0 is a\_1)  
print(a\_0 == a\_1)

>>>False

>>>Ture

In this case, list a\_0 and a\_1 are refer to the individual object with the same value.

Example2

a\_0=[1,2,3]  
a\_1=a\_0  
print(a\_0 is a\_1)  
print(a\_0 == a\_1)

>>>True

>>>True

In example2, list a\_0 and a\_1 is refer to the same object, a\_1 and a\_2 are in the same value.

a\_0, a\_1 are two references, [1,2,3] is object, a\_1=a\_0 is aliasing relate a\_0 to a\_1.

In part2 define a refer function below:

def refer(reference\_1):  
 reference2=reference\_1  
 return reference2  
  
k=[1,2,3]  
i=refer(k)  
print(k is i)

>>> True

In this example I assigned a list object to k, variable k is a reference, then I pass k as argument to function. Return value assigned the same object with k, then pass the return value to variable i, the result printed in console mean there are two references to the same object.

Question:

In 10.6 List methods present the useful method sort(), Is this built-in method an efficient way to sort a list? what other sort algorithm do you know?

Reference  
Downey, Allen (2015). *Think Python. How to Think Like a Computer Scientist. 2nd Edition, Version 2.2.23*