```
# -*- coding: utf-8 -*-
"""
Created on Sat Oct 10 22:14:44 2015

@author: hina
"""

print ()

# Looping Techniques for the different data structures

# When looping through dictionaries, the key and corresponding value
# can be retrieved at the same time using the items() method.
grades = {'jack':90, 'jill':100, 'joseph':99, 'natasha':95, 'eric':100, 'audrey':90}
for k, v in grades.items():
    print (k, v)
print ()

# When looping through a sequence, the position index and corresponding value
# can be retrieved at the same time using the enumerate() function.
points = [90, 100, 99, 95, 100, 90]
for i, v in enumerate(points):
    print (i, v)
print ()

# To loop over two or more sequences at the same time,
# the entries can be paired with the zip() function.
names = ['jack', 'jill', 'joseph', 'natasha', 'eric', 'audrey']
points = [90, 100, 99, 95, 100, 90]
for n, p in zip(names, points):
    print(n, p)
print ()

# To loop over a sequence in sorted order, use the sorted() function
# which returns a new sorted list while leaving the source unaltered.
for x in sorted(grades.keys()):
    print (x)
print ()

# To loop over a sequence in reverse, first specify the sequence
# in a forward direction and then call the reversed() function.
for x in reversed(sorted(grades.keys())):
    print (x)
print ()

############################

# Comparing Sequences and other types.
#
# Sequence objects may be compared to other objects with the same sequence type.
# The comparison uses lexicographical ordering: first the first two items are
# compared, and if they differ this determines the outcome of the comparison;
# if they are equal, the next two items are compared, and so on, until either
# sequence is exhausted. If two items to be compared are themselves sequences
# of the same type, the lexicographical comparison is carried out recursively.
# If all items of two sequences compare equal, the sequences are considered equal.
```

```
#
# Note that comparing objects of different types with < or > is legal provided
# that the objects have appropriate comparison methods. For example,
# mixed numeric types are compared according to their numeric value,
# so 0 equals 0.0, etc. Otherwise, rather than providing an arbitrary ordering,
# the interpreter will raise a TypeError exception.

# all of these evaluate to true

print ((1, 2, 3)              < (1, 2, 4))
print ([1, 2, 3]              < [1, 2, 4])
print ('ABC' < 'C' < 'Pascal' < 'Python')
print ((1, 2, 3, 4)           < (1, 2, 4))
print ((1, 2)                 < (1, 2, -1))
print ((1, 2, 3)             == (1.0, 2.0, 3.0))
print ((1, 2, ('aa', 'ab'))   < (1, 2, ('abc', 'a'), 4))

print ()

# test

ratings = {'everest':5, 'minions':3, 'big':4}
for k, v in ratings:
    print (k, v)

scores = [9, 10, 8, 5, 10, 6]
for i, v in scores:
    print (i, v)
print ()

ratings = {'everest':5, 'minions':3, 'big':4}
for x in reversed(ratings.keys()):
    print (x)
print ()
```