Python Data Structures II: Ranges, Sets, Dictionaries

Hina Arora

Ranges

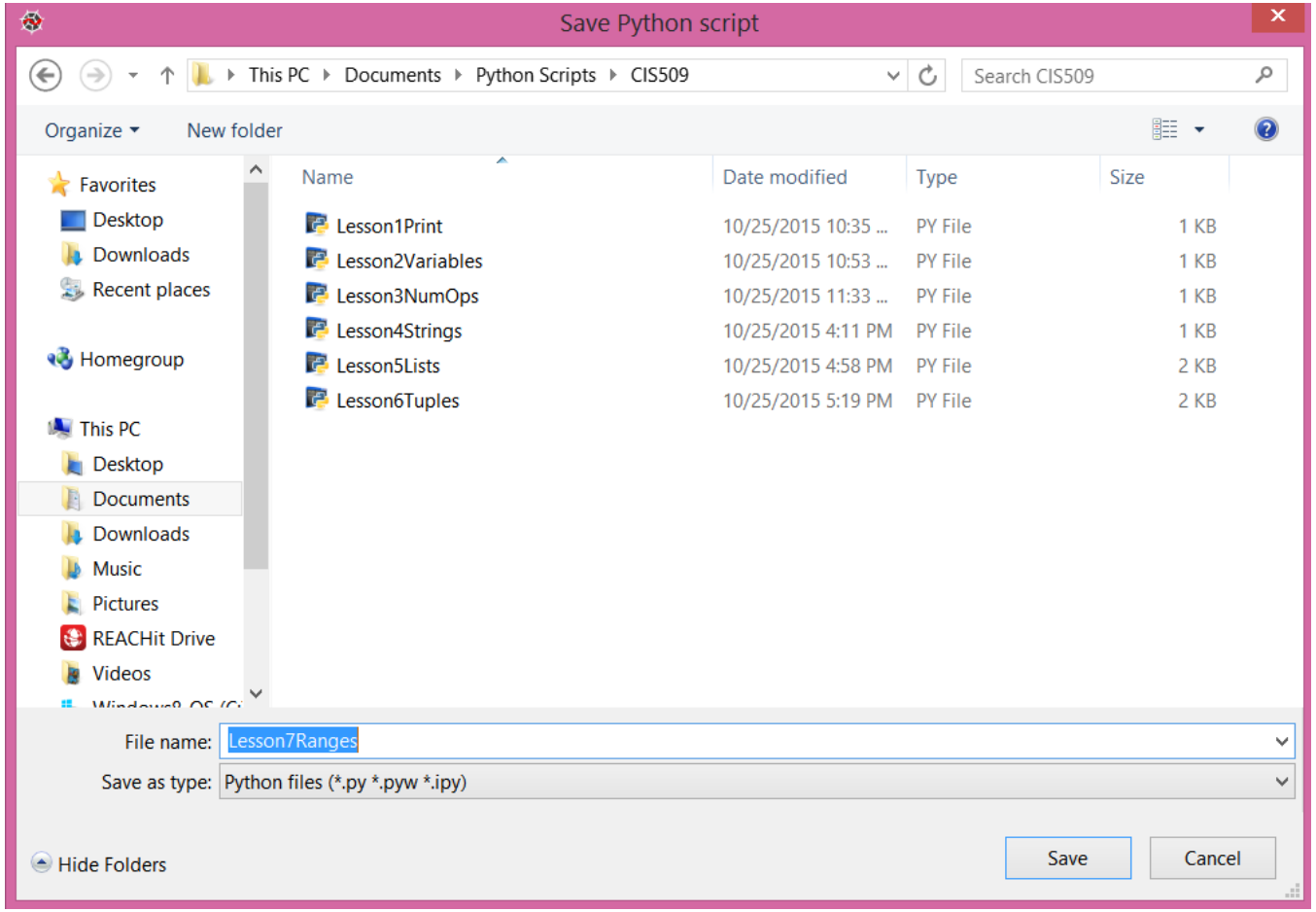
follow along!

(Reference: https://docs.python.org/3.1/tutorial/datastructures.html)

Ranges are immutable sequence of numbers typically used to loop through for loops . Ranges are indexed by number.
range (start=0, stop, step=1)

Create a new file called “Lesson7Range.py”

- Go to File -> New file…
- Go to File -> Save as…
- Go to CISPy directory
- Save file as “Lesson7Range.py”



Detour: we need to understand for loops to appreciate Ranges

```
10 # The for statement in Python differs a bit from what you may be used to.
11 # Rather than always iterating over an arithmetic progression of numbers, or
12 # giving you the ability to define both the iteration step and halting
13 # condition, Python's for statement iterates over the items of any iterable
14 # sequence (such as a list, tuple, or string), in the order that they appear
15 # in the sequence
16
17 words = ['cat', 'dog', 'cow', 'parrot', 'hamster', 'goat']
18 for w in words:
19     print (w, len(w))
20
21 print ()
22
```

```
# The range function generates arithmetic prgogressions

# range(n) -> 0,...,n-1 in increments of 1
for i in range(5):
    print (i, end=',')
print ()

```

```
# range(m,n) -> m,...,n-1 in increments of 1
# if m >= n, returns nothing
for i in range(3,10):
    print (i, end=',')
print ()

```

```
# counts in positive increments if n > m, and k is positive
# range(m,n,k) -> m,...,<=n-1 in increments of k
for i in range(3,10,2):
    print (i, end=',')
print ()

```

```
# counts in negative increments if n < m, and k is negative
# range(m,n,k) -> m,...,>=n+1 in increments of k
for i in range(10,-30,-5):
    print (i, end=',')
print ()

```

```
48 words = ['jane', 'john', 'mark', 'harry', 'mike', 'ed']
49 wordlist = []
50 for i in range(len(words)):
51     wordlist.append([i, words[i]])
52 print (wordlist)
53
54 print ()
55
```

```
56 # Note: The object returned by range() behaves as if it is a list,
57 # but in fact it isn't. It is an object which returns the successive items of
58 # the desired sequence when you iterate over it, but it doesn't really make
59 # the list, thus saving space.
60
61 r = range(0, 20, 2)
62 print (r)
63 print (list(r))
64 print (11 in r)
65 print (r.index(10))
66 print (r[5])
67 print (r[:5])
68 print (r[-1])
69
70 print ()
71
```

```
72 # Testing range objects for equality with == and != compares them as sequences.
73 # That is, two range objects are considered equal if they represent the same sequence of values
74 print (range(0) == range(2, 1, 3))
75
76 print ()
77
```

Will these statements work?

```
78 # test
79
80 for i in range():
81     print (i, end=',')
82 print()
83
84 for i in range(10):
85     print (i, end=',')
86 print()
87
88 for i in range(0,-10):
89     print (i, end=',')
90 print()
91
92 for i in range(0,-10,-2):
93     print (i, end=',')
94 print ()
95
```

Sets

follow along!

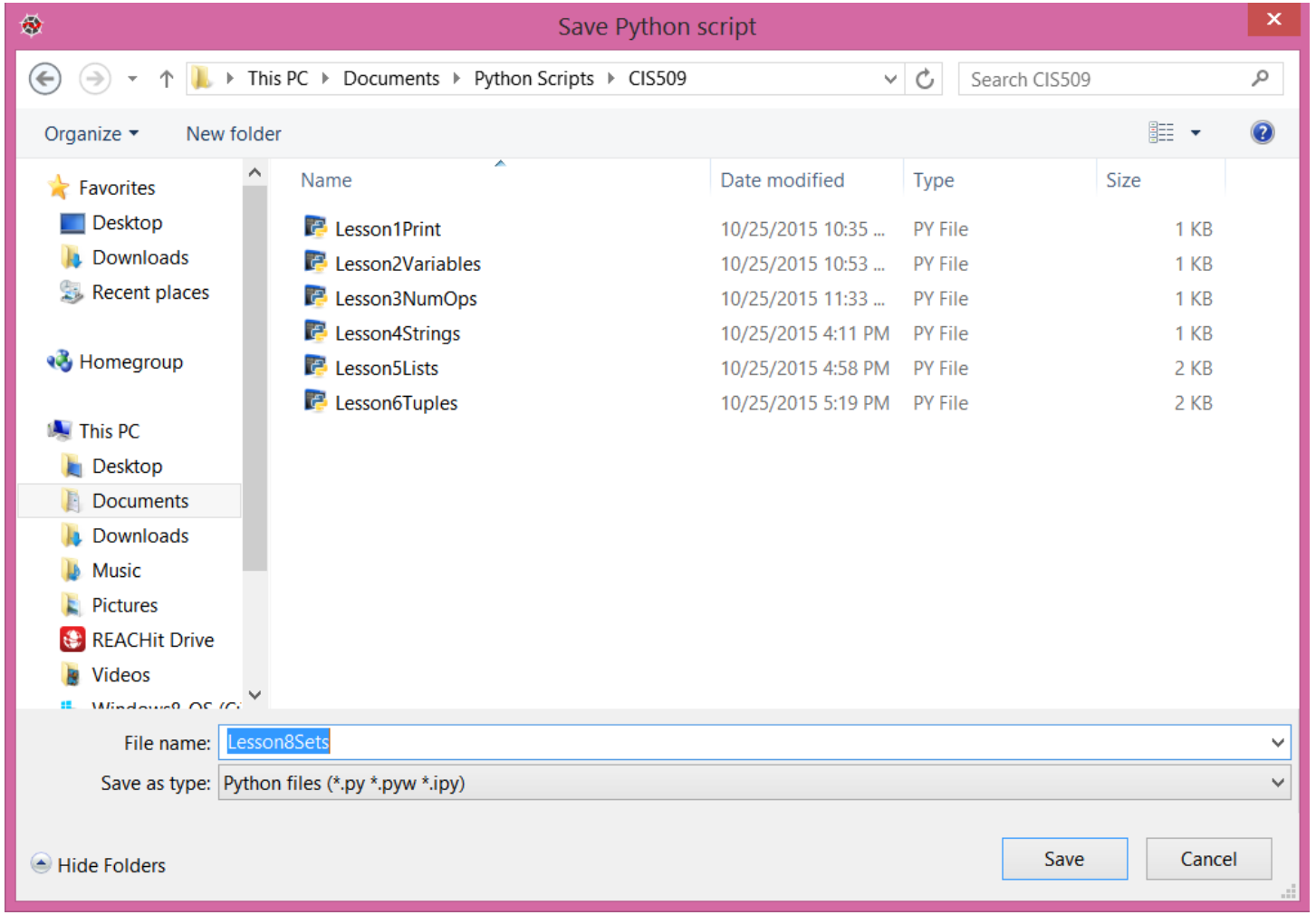
(Reference: https://docs.python.org/3.1/tutorial/datastructures.html)

Sets are unordered collection with no duplicate elements. Sets can not be indexed.

{a, b, c}

Create a new file called “Lesson8Sets.py”

- Go to File -> New file…
- Go to File -> Save as…
- Go to CISPy directory
- Save file as “Lesson8Sets.py”



```
# A set is an unordered collection with no duplicate elements.
# Basic uses include membership testing and eliminating duplicate entries.
# Set objects also support mathematical operations like union, intersection, difference, and symmetric difference.

# duplicates will automatically be removed
basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
print (basket)
print ()

# test for membership
print ('orange' in basket)
print ()

# another way to declare single character sets
someChars = set ('235345#$$%^$%&abdfsdf')
print (someChars)
print ()

```

```
28 # set operations
29
30 set1 = set ('abracadabra')
31 set2 = set ('alacazam')
32
33 print ("set1:         ", set1)
34 print ("set2:         ", set2)
35
36 # union
37 print ("set1 | set2: ", set1 | set2)
38
39 # intersection
40 print ("set1 & set2: ", set1 & set2)
41
42 # difference
43 print ("set1 - set2: ", set1 - set2)
44
45 # symmetric difference
46 print ("set1 ^ set2: ", set1 ^ set2)
47
48 print ()
```

Will these statements work?

```
50 # test
51
52 set1 = set ('abracadabra')
53 set2 = set ('alacazam')
54 print ((set1 ^ set2) - ((set1 | set2) - (set1 & set2)))
55 print ((set1 ^ set2) - ((set1 - set2) | (set2 - set1)))
56
57 set1 = {'dog', 'cat', 'deer'}
58 print (set1)
59
60 set2 = set ('sparrow', 'hawk', 'eagle')
61 print (set2)
62
63 set3 = set ('abc123')
64 print (set3)
65
66 set4 = set ('\'abc123\'')
67 print (set4)
68
69 print ()
70
```

Dictionaries

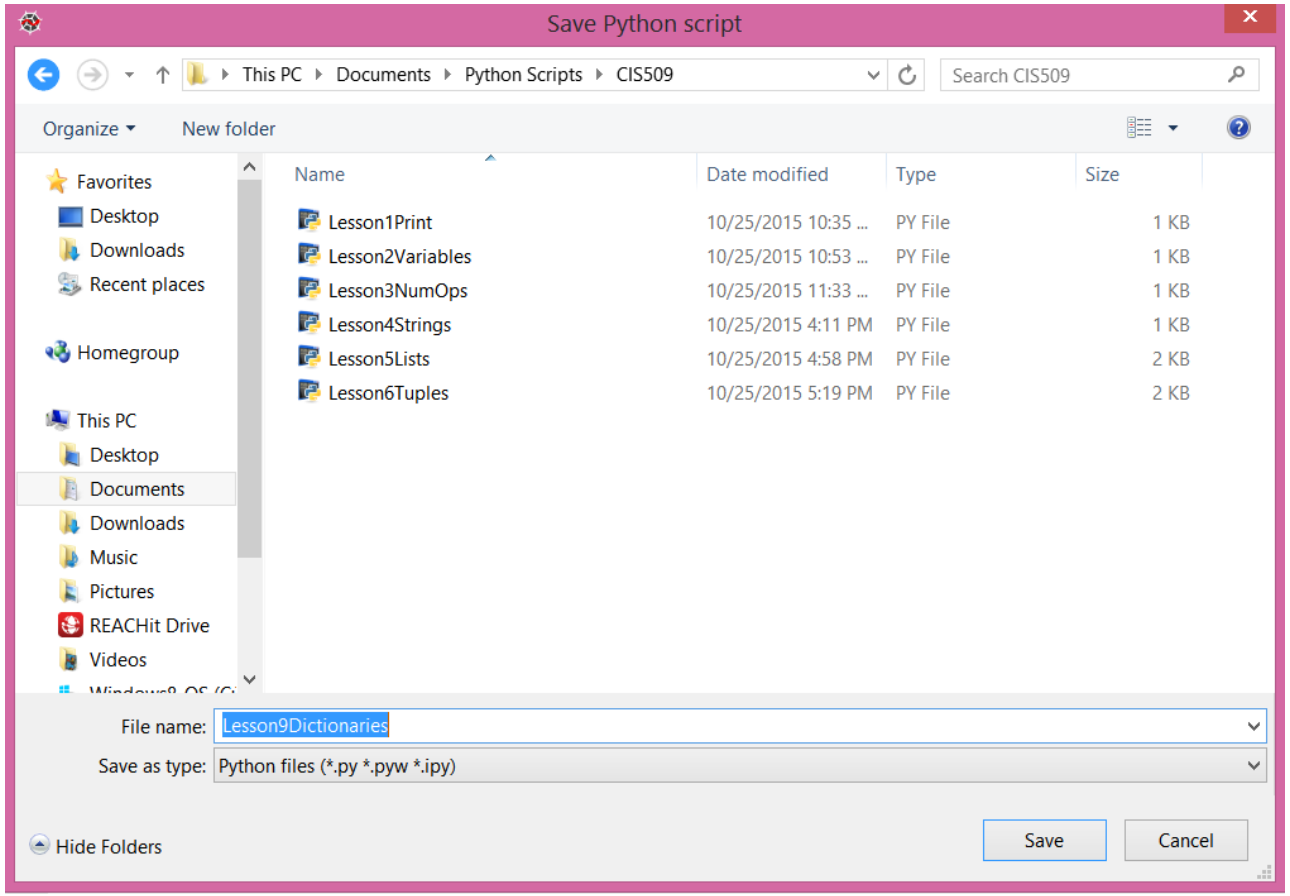
follow along!

(Reference: https://docs.python.org/3.1/tutorial/datastructures.html)

Dictionaries are unordered set of <key:value> pairs,
where the keys are of immutable type, and must be unique within one dictionary.
Dictionaries are indexed by keys.
{a:1, b:2, c:3}

Create a new file called “Lesson9Dictionary.py”

- Go to File -> New file…
- Go to File -> Save as…
- Go to CISPy directory
- Save file as “Lesson9Dictionary.py”



```
# Dictionaries are unordered set of <key:value> pairs, where the keys are of immutable type,
# and must be unique within one dictionary. Dictionaries are indexed by keys.
# Refered to as “associative memories” or “associative arrays” in other languages.

# this creates an empty dictionary
grades = {}
print (len(grades))
print ()

# this creates a dictionary directly using comma-separated key:value pairs
grades = {'jack':90, 'jill':100, 'joe':99, 'nat':95, 'eric':100, 'aubry':90}
print (grades)
print ()

# this creates a dictionary with the constructor using sequences of key:value pairs
grades = dict([('jack', 90), ('jill', 100), ('joe', 99), ('nat', 95), ('eric', 100), ('aubry', 90)])
print (grades)
print ()

# this creates a dictionary with the constructor using keyword arguments (can only be done if keys are simple strings)
grades = dict(jack=90, jill=100, joe=90, nat=95, eric=100, aubry=90)
print (grades)
print ()

```

```
# this returns list of all keys used in dicitionary in arbitrary order
print (grades.keys())
print ()

# this returns list of all keys used in dicitionary in sorted order
print (sorted(grades.keys()))
print ()

# you can provide the key to extract the corresponding value
print (grades['nat'])
print ()

# you can modify the value associated with a key
grades['aubry'] = 93
print (grades)
print ()

# you can delete a key:value pair from the dictionary
del grades['jack']
print (grades)
print ()

# you can check whether a key is in the dictionary
print ('hina' in grades)
print ()

```

Will these statements work?

```
62 imdbRating = dict(interstellar=8.7, unbroken=7.2, divergent=6.8, wild=7.2, neighbors=6.4)
63 print (imdbRating)
64
65 print(imdbRating['interstellar'])
66
67 imdbRating['unbroken']=10
68 print (imdbRating)
69
70 print(imdbRating['everest'])
71
72 print ()
```

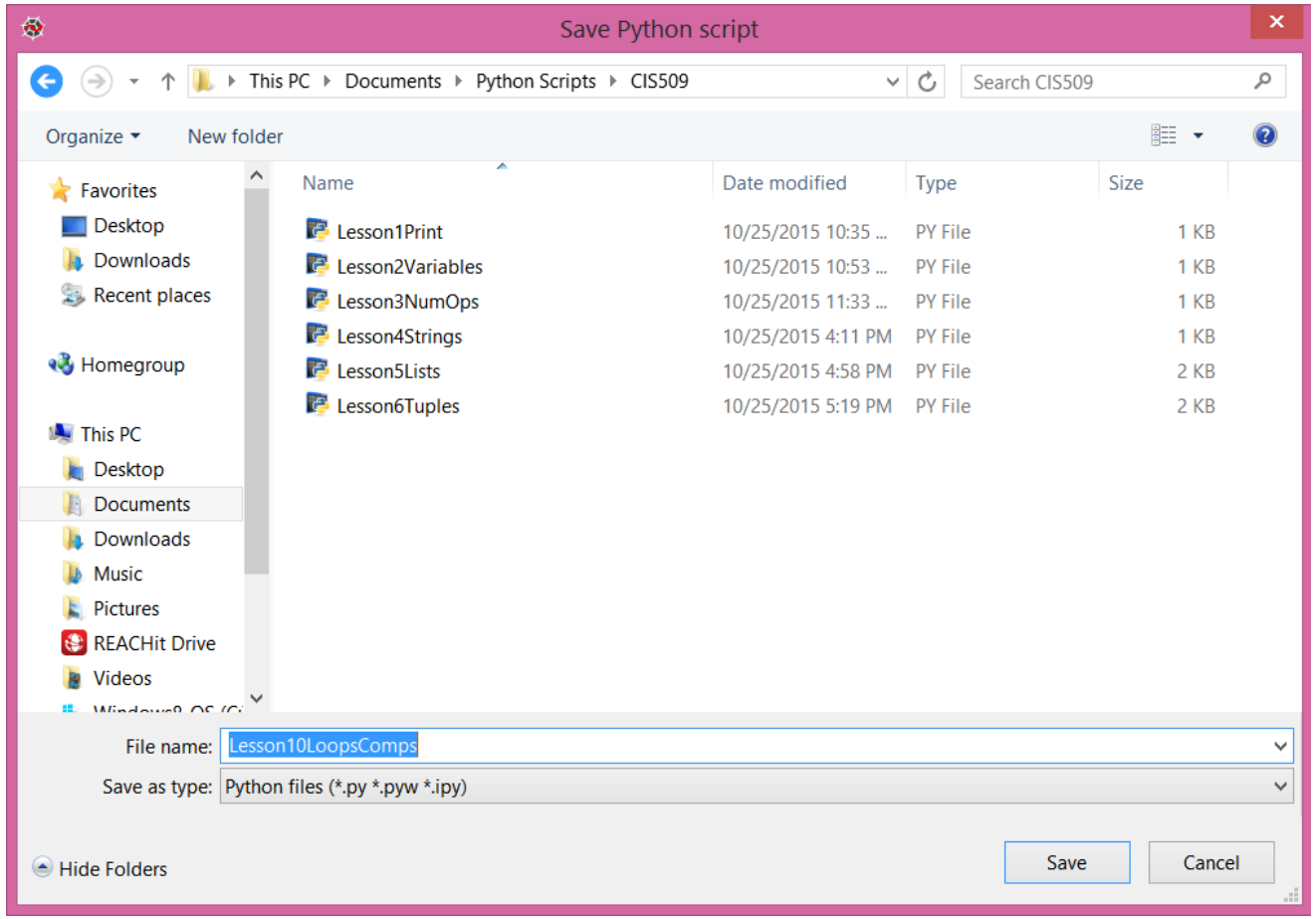
Loops and Comparisons

follow along!

(Reference: https://docs.python.org/3.1/tutorial/datastructures.html)

Create a new file called “Lesson10LoopsComps.py”

- Go to File -> New file…
- Go to File -> Save as…
- Go to CISPy directory
- Save file as “Lesson10LoopsComps.py”



```
# 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 ()
```

```
48 # Comparing Sequences and other types.
49 #
50 # Sequence objects may be compared to other objects with the same sequence type.
51 # The comparison uses lexicographical ordering: first the first two items are
52 # compared, and if they differ this determines the outcome of the comparison;
53 # if they are equal, the next two items are compared, and so on, until either
54 # sequence is exhausted. If two items to be compared are themselves sequences
55 # of the same type, the lexicographical comparison is carried out recursively.
56 # If all items of two sequences compare equal, the sequences are considered equal.
57 #
58 # Note that comparing objects of different types with < or > is legal provided
59 # that the objects have appropriate comparison methods. For example,
60 # mixed numeric types are compared according to their numeric value,
61 # so 0 equals 0.0, etc. Otherwise, rather than providing an arbitrary ordering,
62 # the interpreter will raise a TypeError exception.
63
64 # all of these evaluate to true
65
66 print ((1, 2, 3)              < (1, 2, 4))
67 print ([1, 2, 3]              < [1, 2, 4])
68 print ('ABC' < 'C' < 'Pascal' < 'Python')
69 print ((1, 2, 3, 4)           < (1, 2, 4))
70 print ((1, 2)                 < (1, 2, -1))
71 print ((1, 2, 3)             == (1.0, 2.0, 3.0))
72 print ((1, 2, ('aa', 'ab'))   < (1, 2, ('abc', 'a'), 4))
73
74 print ()
```

Will these statements work?

```
76 # test
77
78 ratings = {'everest':5, 'minions':3, 'big':4}
79 for k, v in ratings:
80     print (k, v)
81
82 scores = [9, 10, 8, 5, 10, 6]
83 for i, v in scores:
84     print (i, v)
85 print ()
86
87 ratings = {'everest':5, 'minions':3, 'big':4}
88 for x in reversed(ratings.keys()):
89     print (x)
90 print ()
```