Lists and dictionaries

Lists: ordered collections of things

```
In [1]: pets = ['fido', 'molly', 'tweety']
        pets[0]      # get 1st element of list
Out[1]: 'fido'       # result is a string

In [2]: pets[1:3]    # get 2nd and 3rd element
Out[2]: ['molly', 'tweety'] # result is a list
```

Dictionaries: unordered collections of key-value pairs

```
In [1]: pets = {'fido':'dog', 'molly':'cat'}
        pets['fido'] # return the value for key 'fido'
Out[1]: 'dog'


In [2]: 'molly' in pets   # does dict have key 'molly’?
Out[2]: True              # yes


In [3]: 'tweety' in pets   # does dict have key 'tweety'?
Out[3]: False              # no
```

Conditional code execution

if/else statements

```
if condition:
    statement
else:
    alternative statement
```

`if/else` statements

```
if condition:
    statement
else:   # optional, can be omitted
    alternative statement
```

if/else statements

```
if condition:
    statement
```

Simple `if`/`else` example

```
In [1]: if 2<3:
            print("yes")
        else:
            print("no")

Out[1]: yes
```

Simple if/else example

```
In [1]: if 3<2:
            print("yes")
        else:
            print("no")

Out[1]: no
```

Indentation defines code blocks

```
In [1]: if 3<2: # False
            print("1") # not run
            print("2") # not run
            print("3") # not run
        print("4")     # run

Out[1]: 4
```

Indentation defines code blocks

```
In [1]: if 2<3: # True
            print("1") # run
            print("2") # run
            print("3") # run
        print("4")     # run


Out[1]: 1
        2
        3
        4
```

Doing things multiple times (loops)

for loops

```
for variable in list:
   statement
```

for-loop example

```
In [1]: for name in ["John", "Sara", "Bill"]:
            print(name)

Out[1]: John
        Sara
        Bill
```

Again, indentation defines code blocks

```
In [1]: for name in ["John", "Sara", "Bill"]:
            print("----")   # run for every name
            print(name)     # run for every name
        print("----")       # run once

Out[1]: ----
        John
        ----
        Sara
        ----
        Bill
        ----
```

We use `for` loops when we want to do something a number of times

```
In [1]: for i in range(5):  # range(5) creates the
            print("Hello!") # numbers from 0 to 4

Out[1]: Hello!
        Hello!
        Hello!
        Hello!
        Hello!
```

We use `for` loops when we want to do something a number of times

```
In [1]: for i in range(5):        # range(5) creates the
            print("Hello:", i) # numbers from 0 to 4

Out[1]: Hello: 0
        Hello: 1
        Hello: 2
        Hello: 3
        Hello: 4
```

One more example:
Make a list of the numbers 1 through 5

```
In [1]: result = [] # start with empty list
        for i in range(1, 6): # count from 1 to 5
            result.append(i)
        print(result)

Out[1]: [1, 2, 3, 4, 5]
```

Combining loops and conditional execution

We often combine `for` loops and `if` statements

Typical example:

Loop over all elements in a list, and do an action if some condition is met.

Example:
Find names starting with 'S'

```
In [1]: for name in ["John", "Sara", "Bill"]:
            if name[0]=='S':
                print(name, "starts with S")
            else:
                print(name, "doesn't start with S")

Out[1]: John doesn't start with S
        Sara starts with S
        Bill doesn't start with S
```

Example:
Count names starting with 'S'

```
In [1]: count = 0     # start with count of 0
        for name in ["John", "Sara", "Bill"]:
            if name[0]=='S':
                count += 1  # increase count by 1
        print(count) # print final result

Out[1]: 1
```

Last example: Count how often letters occur in a string

```
In [1]: sentence = "Time flies like an arrow."
        # first we count, using a dict
        counts = {} # empty dict
        for c in sentence:
            if c in counts:  # have we seen this letter before?
                counts[c]+=1 # yes, increase count by 1
            else:
                counts[c]=1      # no, set count to 1

        # now that we have the counts, we print them
        for c in counts:     # loop over all letters in the dict
            print(c, "appears", counts[c], "times.")
```

Last example: Count how often letters occur in a string

```
Out[1]: i appears 3 times.
        k appears 1 times.
        o appears 1 times.
        r appears 2 times.
        l appears 2 times.
          appears 4 times.
        n appears 1 times.
        m appears 1 times.
        f appears 1 times.
        e appears 3 times.
        . appears 1 times.
        s appears 1 times.
        T appears 1 times.
        a appears 2 times.
        w appears 1 times.
```