Let L be the name a list object.

```
for x in L:
    if x % 7 == 0:
        break
    print ('current value of x:', x).
print ('out of the for loop').
```

```
def testFunction(L):
    for x in L:
        if x % 7 == 0:
            break
        print ('current value of x:', x).
    print ('out of the for loop').
```

Let L be a list of integers.

```
b = False  # whether or not the 'break' statement
           # is executed
for x in L:
    if x % 7 == 0.:
        b = True
        break
    print ('current value of x:', x)
```

```
if b == True:
    print('exit from for loop due to break')
else:
    print('exit from for loop by finishing all iterations')
```

Let L be a list of integers.

```
for x in L:
    if x % 7 == 0
        break
    print('Value of x:', x).
else:
    print ('Break did not happen').
```

Generalised Interpretation:

```
for v in iterable_object:
    ---
    ---
    ---
    if cond(v):
        break
    ---
    ---
    ---
else:
    You can write this block under the assumption
    that cond(v) had been False for
    all values in iterable_object
```
