We worked out three examples:

1. Print the contents of a matrix, when implemented with nested lists:

   ```python
def show(matrix):
   for row in range(len(matrix)):
       for col in range(len(matrix[0])):
           print str(matrix[row][col]).rjust(4),
       print

show([[1, 2, 3], [-10, 100, 0], [12, 1, 200]])
```

   Notice how information about the size can be deduced from the matrix itself.

2. Sort a list of integers:

   ```python
def sort(seq):
   sorted = False
   while sorted == False:
       sorted = True # always start with hope
       for ind in range(len(seq)-1):
           if seq[ind] > seq[ind+1]: # sort ascending
               temp = seq[ind]
               seq[ind] = seq[ind+1]
               seq[ind+1] = temp
               sorted = False

   print seq

sort([1, 2, 1, 3, 1, 4, 1, 3, 2, 0])
```

3. Generate a random list of integers:

   ```python
import random

def generate(size):
    result = []
    for ind in range(size):
        result.append(random.randrange(-50, 50))
    return result

One can use this in conjunction with the previous function as follows:

   ```python
def sort(seq):
   ```
sorted = False
while sorted == False:
    sorted = True # always start with hope
    for ind in range(len(seq)-1):
        if seq[ind] > seq[ind+1]:  # sort ascending
            temp = seq[ind]
            seq[ind] = seq[ind+1]
            seq[ind+1] = temp
    sorted = False
print seq

The definition is repeated here just for convenience.

    a = generate(10)
    print a
    sort(a)

Just put everything (shown at this step) in one file, then run it.

Here’s an example of output:

    >>>
    [8, -27, 2, -11, 23, 32, 22, 41, 23, -45]
    [-45, -27, -11, 2, 8, 22, 23, 23, 32, 41]
    ```
    >>>

This was the first example of using return to provide a value to the code that calls the
function. A brief discussion further explained the difference between return and print.