More on lists

L = ["a", "b", "c", "d", "e"]

print(L[0])  
a
print(L[4])  
e
print(L[5])  
error

L[1] = "x"  
[a, x, c, d, e]

s = L[0] + L[1]  
ab
More on lists

```
x = 5
y = x
print(x)
y = y + 2
print(y)
print(x)
```
More on lists

x = 5
y = x
print(x)  5
y = y + 2
print(y)
print(x)
More on lists

```python
x = 5
y = x
print(x)  # 5
y = y + 2
print(y)  # 7
print(x)  # 5
```

More on lists

```python
x = 5
y = x
print(x)  # Output: 5
y = y + 2
print(y)  # Output: 7
print(x)  # Output: 5
```
More on lists

```python
x = ["a", "b", "c"]
y = x
y[0] = "d"

print(y)
print(x)
```
More on lists

```python
x = ["a", "b", "c"]
y = x
y[0] = "d"

print(y)  # Output: ['d', 'b', 'c']
print(x)  # Output: ['d', 'b', 'c']
```
More on lists

```python
x = ['a', 'b', 'c']
y = x
y[0] = 'd'

print(y)  # ['a', 'b', 'd']
print(x)  # ['a', 'b', 'c']
```
More on lists

\[ x = 5 \]

\[ x : 5 \]
More on lists

\[ x = ['a', 'b', 'c'] \]
More on lists

```python
x = ["a", "b", "c"]
y = x
```

![Diagram showing the relationship between the addresses and content of x and y.]
More on lists

```python
x = ["a", "b", "c"]
y = x
y[0] = "d"

print(y)
print(x)
```
# add one to each element of each sub-list
L = [[1,2,3], [4,5,6], [7,8,9]]
# add one to each element of each sub-list
L = [[1,2,3], [4,5,6], [7,8,9]]

for i in range(0,len(L)):
    L[i] = L[i] + 1
# add one to each element of each sub-list
L = [[1,2,3], [4,5,6], [7,8,9]]

for i in range(0, len(L)):
    L[i] = L[i] + 1

for i in range(0, len(L)):
# add one to each element of each sub-list

\[ L = [[1,2,3], [4,5,6], [7,8,9]] \]

```python
for i in range(0, len(L)):
    L[i] = L[i] + 1

for i in range(0, len(L)):
    L[i][0] = L[i][0] + 1
    L[i][1] = L[i][1] + 1
    L[i][2] = L[i][2] + 1
```
# add one to each element of each sub-list
L = [[1,2], [3,4,5], [6,7,8,9]]

for i in range(0,len(L)):
    L[i][0] = L[i][0] + 1
    L[i][1] = L[i][1] + 1
    L[i][2] = L[i][2] + 1
# add one to each element of each sub-list
L = [[1, 2], [3, 4, 5], [6, 7, 8, 9]]

for i in range(0, len(L)):
    L[i][0] = L[i][0] + 1
    L[i][1] = L[i][1] + 1
    L[i][2] = L[i][2] + 1
# add one to each element of each sub-list
L = [[1,2], [3,4,5], [6,7,8,9]]

for i in range(0,len(L)):
# add one to each element of each sub-list
L = [[1,2], [3,4,5], [6,7,8,9]]

for i in range(0,len(L)):
    for j in range(0,len(L[i])):
# add one to each element of each sub-list
L = [[1,2], [3,4,5], [6,7,8,9]]

for i in range(0,len(L)):
    for j in range(0,len(L[i])):
        L[i][j] = L[i][j] + 1
nested for loops

```python
for i in range(0,5):
    print(i)
```
nested for loops

```python
for i in range(0, 5):
    print(i)
```

```
0
1
2
3
4
```
nested for loops

for i in range(0,5):
    for j in range(0,5):
        print i,j
nested for loops

for i in range(0,5):
    for j in range(0,5):
        print(i,j)

0,0
nested for loops

for i in range(0,5):
    for j in range(0,5):
        print(i, j)

0,0
0,1
nested for loops

```python
for i in range(0, 5):
    for j in range(0, 5):
        print(i, j)
```

0,0
0,1
0,2
0,3
nested for loops

```python
for i in range(0,5):
    for j in range(0,5):
        print(i,j)
```

```
0,0
0,1
0,2
0,3
0,4
1,0 ...
```
# --- DRILL ---
# draw a 9 x 9 checkerboard with alternating blue and red squares – you should use nested loops to iterate through each row/column

import drawSvg as draw
d = draw.Drawing(200, 200, origin='center')
d.append(draw.Rectangle(-90,-100,20, 20, fill='blue'))
# --- DRILL ----
# draw a 9 x 9 checkerboard with alternating blue and red squares – you should use nested loops to iterate through each row/column

```python
import drawSvg as draw
d = draw.Drawing(200, 200, origin='center')

c = 1 # control color
for y in range(0,9):
    for x in range(0,9):
        if( c == 1 ):
            d.append(draw.Rectangle(-90+x*20, -100+y*20, 20, 20, fill='blue'))
        c = 0 # change color
        else:
            d.append(draw.Rectangle(-90+x*20, -100+y*20, 20, 20, fill='red'))
        c = 1 # change color
d```

def reverse_list(l):
    for i in range(0, len(l)/2):
        j = len(l) - 1 - i # right index
        temp = l[i] # swap
        l[i] = l[j] #
        l[j] = temp #

test_list = [1, 3, 5, 7, 9, 11, 13, 15]
print(test_list)
reverse_list(test_list)
print(test_list)
def reverse_list(l):
    for i in range(0, len(l) // 2):
        j = len(l) - 1 - i  # right index
        temp = l[i]  # swap
        l[i] = l[j]  #
        l[j] = temp  #

test_list = [1, 3, 5, 7, 9, 11, 13, 15]
print(test_list)
reverse_list(test_list)
print(test_list)
def reverse_list(l):
    for i in range(0, len(l)/2):
        j = len(l) - 1 - i  # right index
        temp = l[i]  # swap
        l[i] = l[j]  #
        l[j] = temp  #

test_list = [1, 3, 5, 7, 9, 11, 13, 15]
print(test_list)
reverse_list(test_list)
print(test_list)

[1, 3, 5, 7, 9, 11, 13, 15]
[15, 13, 11, 9, 7, 5, 3, 1]
More on lists

def reverse_list(l):
    for i in range(0, len(l) // 2):
        j = len(l) - 1 - i  # right index
        temp = l[i]  # swap
        l[i] = l[j]  #
        l[j] = temp  #

test_list = [1, 3, 5, 7, 9, 11, 13, 15]
print(test_list)
reverse_list(test_list)
print(test_list)

test_list = [1, 3, 5, 7, 9, 11, 13, 15]
new_list = test_list
reverse_list(new_list)
print(new_list)
print(test_list)
More on lists

def reverse_list(l):
    for i in range(0, len(l)/2):
        j = len(l) - 1 - i  # right index
        temp = l[i]  # swap
        l[i] = l[j]  #
        l[j] = temp  #

test_list = [1, 3, 5, 7, 9, 11, 13, 15]
print(test_list)
reverse_list(test_list)
print(test_list)

test_list = [1, 3, 5, 7, 9, 11, 13, 15]
new_list = test_list
reverse_list(new_list)
print(new_list)
print(test_list)
More on lists

def reverse_list(l):
    for i in range(0, len(l) // 2):
        j = len(l) - 1 - i  # right index
        temp = l[i]  # swap
        l[i] = l[j]  #
        l[j] = temp  #

# Example usage

test_list = [1, 3, 5, 7, 9, 11, 13, 15]
print(test_list)
reverse_list(test_list)
print(test_list)

new_list = test_list
reverse_list(new_list)
print(new_list)
print(test_list)
More on lists

def reverse_list(l):
    for i in range(0, len(l) // 2):
        j = len(l) - 1 - i  # right index
        temp = l[i]  # swap
        l[i] = l[j]  #
        l[j] = temp  #

test_list = [1, 3, 5, 7, 9, 11, 13, 15]
print(test_list)
reverse_list(test_list)
print(test_list)

test_list = [1, 3, 5, 7, 9, 11, 13, 15]
new_list = test_list
reverse_list(new_list)
print(new_list)
print(test_list)

test_list = [1, 3, 5, 7, 9, 11, 13, 15]
copy_list = list(test_list)  # make a copy
reverse_list(copy_list)
print(copy_list)
print(test_list)
More on lists

test_list = [1, 3, 5, 7, 9, 11, 13, 15]
print(test_list)
reverse_list(test_list)
print(test_list)

test_list = [1, 3, 5, 7, 9, 11, 13, 15]
new_list = test_list
reverse_list(new_list)
print(new_list)
print(test_list)

test_list = [1, 3, 5, 7, 9, 11, 13, 15]
copy_list = list(test_list) # make a copy
reverse_list(copy_list)
print(copy_list)
print(test_list)
def reverse_list(l):
    for i in range(0, len(l)/2):
        j = len(l) - 1 - i  # right index
        temp = l[i]  # swap
        l[i] = l[j]
        l[j] = temp

test_list = [1, 3, 5, 7, 9, 11, 13, 15]
print(test_list)
reverse_list(test_list)
print(test_list)

new_list = test_list
reverse_list(new_list)
print(new_list)
print(test_list)

test_list = [1, 3, 5, 7, 9, 11, 13, 15]
copy_list = list(test_list)  # make a copy
reverse_list(copy_list)
print(copy_list)
print(test_list)
# sorting a list
alphabetized_ivies = sorted(ivies)

# appending to a list: not dwards[6] = "Grumpy"
dwarfs = ["Happy", "Dopey", "Doc", "Sleepy", "Bashful", "Sneezy"]
dwarfs.append("Grumpy")  # what is this notation?
['Happy', 'Dopey', 'Doc', 'Sleepy', 'Bashful', 'Sneezy', 'Grumpy']

# insert before an index
dwarfs.insert( 1, "Grumpy" )  # what is this notation?
['Happy', 'Grumpy', 'Dopey', 'Doc', 'Sleepy', 'Bashful', 'Sneezy', 'Grumpy']

# deleting from a list
del dwarfs[1]
['Happy', 'Dopey', 'Doc', 'Sleepy', 'Bashful', 'Sneezy', 'Grumpy']
# --- DRILL -----
# write some code that computes the length and
# position of the largest list in a list of lists
# --- DRILL ----
# write some code that computes the length and
# position of the largest list in a list of lists
L = [[1,2,3],[4,5,6,7],[8,9],[10]]

maxLength = 0
maxPos = 0
i = 1
for l in L:
    if( len(l) > maxLength ):
        maxLength = len(l)
        maxPos = i
    i = i + 1

print( "max length is: " + str(maxLength) + " at position: " + str(maxPos) )
# --- DRILL ---
# write some code that computes the length and
# position of the largest list in a list of lists
L = [[[1,2,3],[4,5,6,7],[8,9],[10]]

maxLength = 0
maxPos = 0
for i in range(0,len(L)):
    if( len(L[i]) > maxLength ):
        maxLength = len(L[i])
        maxPos = i

print( "max length is: " + str(maxLength) + " at position: " + str(maxPos) )
# --- DRILL ----
# write some code that computes the length and position of the largest list in a list of lists
L = [[1,2,3],[4,5,6,7],[8,9],[10]]

maxLength = 0
maxPos = 0
for i in range(0,len(L)):
    n = len(L[i]) # only compute once
    if( n > maxLength ):
        maxLength = n
        maxPos = i
print("max length is: " + str(maxLength) + " at position: " + str(maxPos) )