2D Array, Nested Loops

One dimensional arrays and lists

- We have learned lists and some applications.
- For example, for a given list of numbers a_list
  - Compute the sum
  - Compute the average

```
def compute_sum(a_list):
   sum = 0
   for i in range(len(a_list)):
      sum += a_list[i]
   return sum
```

```
def compute_avg(a_list):
   sum = compute_sum(a_list)
   return sum / len(a_list)
```

2D arrays and lists

- Many other applications require 2D arrays and lists
- For example, if we want to compute the average test scores of a class in which we have `n` students and `k` tests. The data will look something like the following.

Name/Test	Test1	Test2	Test3
Allan Johnson	88	82	91
Marco Lima	83	79	86
Jane Rubio	77	88	93
Maria Zhang	85	86	92

- In your up-coming labs and homework you will see other applications.
- We usually need nested loops to handle 2D data.

Sizing up arrays…

How could we create this rectangular array of `0`s?

```
[[0,0,0,0,0],
 [0,0,0,0,0],
 [0,0,0,0,0]]
```

```
x = 3*[ 5*[0] ]
```

or

```
x = 5*[ 3*[0] ]
```

Sizing up arrays…

How could we create this rectangular array of `0`s?

```
x = 3*[ 5*[0] ]
```

```
          ],
    ,0,0,0],
[0,0,0,0,0]]
```

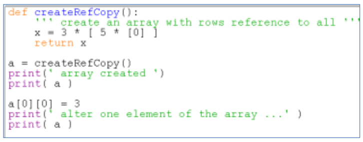
or

```
x = 5
```

but NEITHER ONE works!

because lists are handled by reference !

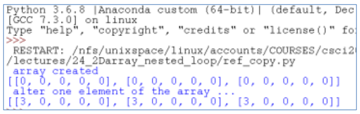
Try ref_copy.py



```
def createRefCopy():
    ''' create an array with rows reference to all '''
    x = 3 * [ 5 * [0] ]
    return x

a = createRefCopy()
print(' array created ')
print( a )

a[0][0] = 3
print(' alter one element of the array ...' )
print( a )
```



```
Python 3.6.8 |Anaconda custom (64-bit)| (default, Dec
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license()" fo
>>>
 RESTART: /nfs/unixspace/linux/accounts/COURSES/csci2
/lectures/24_2Darray_nested_loop/ref_copy.py
 array created
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
 alter one element of the array ...
[[3, 0, 0, 0, 0], [3, 0, 0, 0, 0], [3, 0, 0, 0, 0]]
```

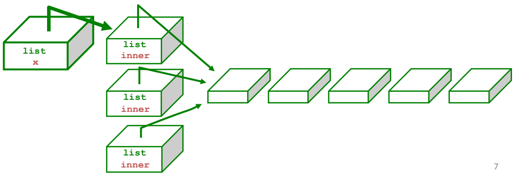
What's really going on?

```
x = 3*[ 5*[0] ]
```

```
inner = 5*[0]
x = 3*[inner]
```

"shallow copy"

copies the list **reference**, not the list data



Safely creating arrays…

```
def create_one_row( width ):
    """ does just that """

    row = []   # start with nothing

    for col in range( width ):  # loop and append!

      row = row + [0]

    return row
```

So, how would you create a ***list of rows***!?

Safely creating arrays…

```
def create2d_array( width, height ):
    """ does just that """

    x = []   # start with nothing

    for row_count in range( height ):
      row = [0] * width
      x = x + [row]

    return x
```

the same approach as before!

24_create_arrays.py

Exercise

Starting with the 2d array **x** shown here, what are the values in **x** after running this code?

x Before

	col 0	col 1	col 2	col 3
row 0	1	2	3	4
row 1	5	6	7	8
row 2	9	10	11	12

```
def mystery(x):
    """ what happens to x ? """

    NUM_ROWS = len(x)
    NUM_COLS = len(x[0])

    for row in range( 0,NUM_ROWS ):
      for col in range( 0,NUM_COLS ):
        if row == col:
          x[row][col] = 42
        else:
          x[row][col] += 1
```

x After

What are the resulting values in x?

Exercise

Starting with the 2d array **x** shown here, what are the values in **x** after running this code?

x Before

	col 0	col 1	col 2	col 3
row 0	1	2	3	4
row 1	5	6	7	8
row 2	9	10	11	12

```
def mystery(x):
    """ what happens to x ? """

    NUM_ROWS = len(x)
    NUM_COLS = len(x[0])

    for row in range( 0,NUM_ROWS ):
      for col in range( 0,NUM_COLS ):
        if row == col:
          x[row][col] = 42
        else:
          x[row][col] += 1
```

x After

42	3	4	5
6	42	8	9
10	11	42	13

What are the resulting values in x?

$Maximum Profit$

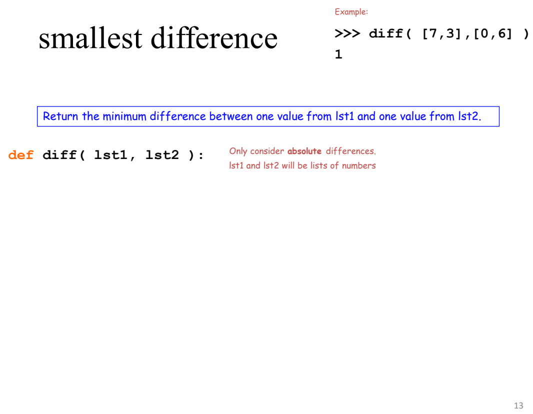
Your stock's prices by the day :

```
prices = [ 40, 80, 10, 30, 27, 52, 5, 15 ]
```

A good investment strategy: maximize your profit!

Day	Price	Stocks valid to sell
0	40.0	40.0
1	80.0	40.0, 80.0
2	10.0	40.0, 80.0, 10.0
3	30.0	40.0, 80.0, 10.0, 30.0
4	27.0	…
5	52.0	…
6	5.0	…
7	15.0	…

you *must* sell ***after*** you buy.



smallest difference

Example:

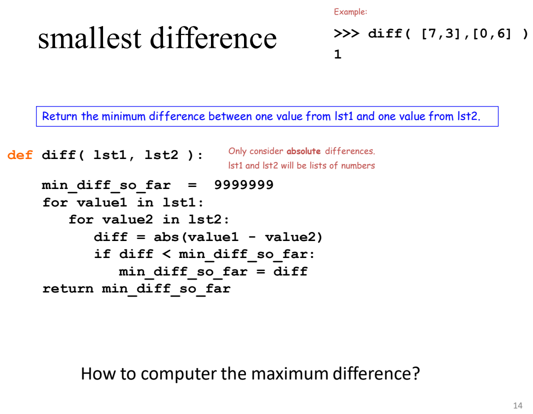
```
>>> diff( [7,3],[0,6] )
1
```

Return the minimum difference between one value from lst1 and one value from lst2.

```
def diff( lst1, lst2 ):
```

Only consider **absolute** differences.
lst1 and lst2 will be lists of numbers

13



smallest difference

Example:

```
>>> diff( [7,3],[0,6] )
1
```

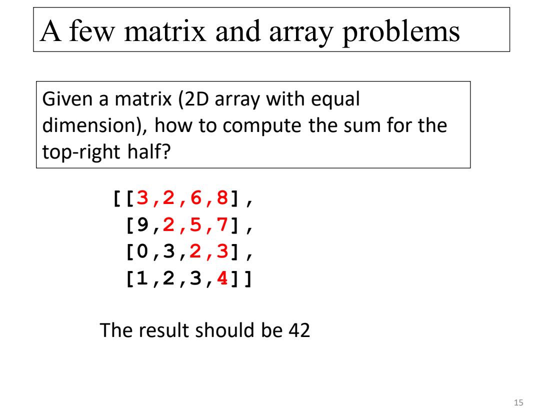
Return the minimum difference between one value from lst1 and one value from lst2.

```
def diff( lst1, lst2 ):
    min_diff_so_far  =  9999999
    for value1 in lst1:
       for value2 in lst2:
          diff = abs(value1 - value2)
          if diff < min_diff_so_far:
             min_diff_so_far = diff
    return min_diff_so_far
```

Only consider **absolute** differences.
lst1 and lst2 will be lists of numbers

How to computer the maximum difference?

14



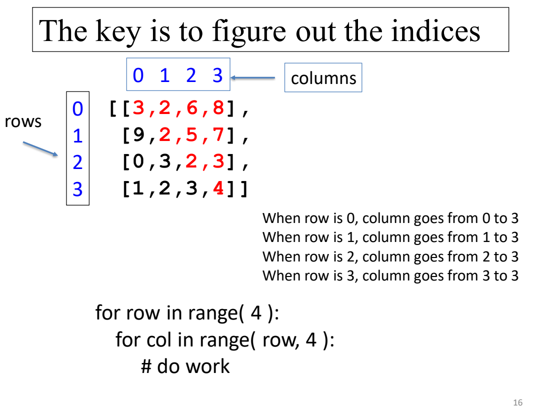
A few matrix and array problems

Given a matrix (2D array with equal dimension), how to compute the sum for the top-right half?

```
[[3,2,6,8],
 [9,2,5,7],
 [0,3,2,3],
 [1,2,3,4]]
```

The result should be 42

15



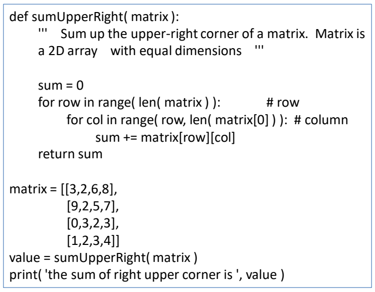
The key is to figure out the indices

```
          0  1  2  3   ← columns
rows   0 [[3,2,6,8],
       1  [9,2,5,7],
       2  [0,3,2,3],
       3  [1,2,3,4]]
```

When row is 0, column goes from 0 to 3
When row is 1, column goes from 1 to 3
When row is 2, column goes from 2 to 3
When row is 3, column goes from 3 to 3

```
for row in range( 4 ):
    for col in range( row, 4 ):
        # do work
```

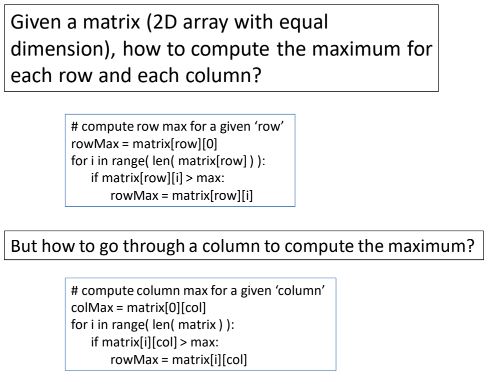
16



```
def sumUpperRight( matrix ):
      '''    Sum up the upper-right corner of a matrix.  Matrix is
      a 2D array     with equal dimensions     '''

      sum = 0
      for row in range( len( matrix ) ):            # row
            for col in range( row, len( matrix[0] ) ):  # column
                  sum += matrix[row][col]
      return sum

matrix = [[3,2,6,8],
            [9,2,5,7],
            [0,3,2,3],
            [1,2,3,4]]
value = sumUpperRight( matrix )
print( 'the sum of right upper corner is ', value )
```



Given a matrix (2D array with equal dimension), how to compute the maximum for each row and each column?

```
# compute row max for a given 'row'
rowMax = matrix[row][0]
for i in range( len( matrix[row] ) ):
    if matrix[row][i] > max:
        rowMax = matrix[row][i]
```

But how to go through a column to compute the maximum?

```
# compute column max for a given 'column'
colMax = matrix[0][col]
for i in range( len( matrix ) ):
    if matrix[i][col] > max:
        rowMax = matrix[i][col]
```

In addition to the row and column maximum, find the maximum of the entire matrix?

```
def findMax( matrix, rowMax, colMax ):
    '''  Given a matrix, find and return the global max, an
         array of row  max and an array of column max    '''

    max = matrix[0][0]                    # current max
    for i in range( len( matrix) ):      # find each row max
        rowMax[i] = findRowMax( matrix, i )
        if rowMax[i] > max:
            max = rowMax[i]

    for i in range( len( matrix[0] ) ):    # find each column max
        colMax[i] = findColMax( matrix, i )
        if colMax[i] > max:
            max = colMax[i]

    return max
```