Nested Loops and Loop Coordinates

•••

Cal Poly CSC Tutoring Department

Review

- Python lists can contain anything as long as they are the same type of item
  - The lst1 shown below is a list of Strings
  - Lists can also contain lists which is called a 2D list

```
lst1 = [“eggs”, “bacon”, “cheese”]

lst2 = [[7, 3, 5], [3, 5, 4], [2, 3]]
```

Review

- When indexing into a list using brackets [], we get the item at that index
  - In this case the item is the sub-list

```
lst = [[7, 3, 5], [3, 5, 4], [2, 3]]
```

lst[0] lst[1] lst[2]

Review

- We can then index into those lists as well with another pair of brackets[]

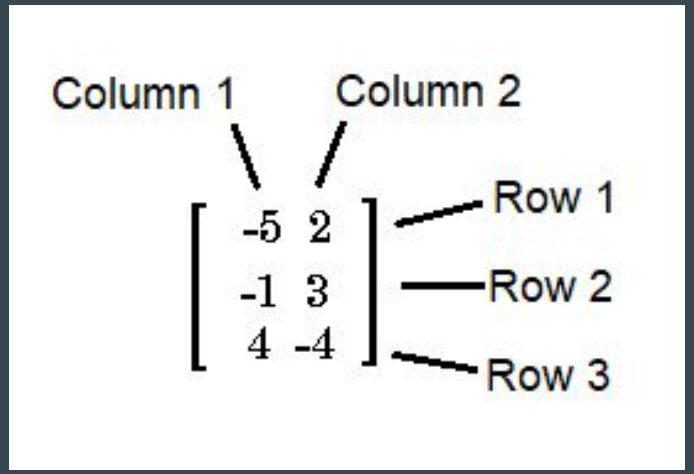
```
lst = [[7, 3, 5], [3, 5, 4], [2, 3]]
```

```
lst[0][1]
```

Why do we like 2D lists?

- They can represent grids or matrices
- A way of separating out distinct lists

```
lst = [[ 5, 2],
       [-1, 3],
       [4, -4]]
```



Iterating through a 2D list

```
for i in range(len(lst)):
    for j in range(len(lst[i])):
        print(lst[i][j])
```

i steps through each index in the main list starting at index 0

```
lst = [[7, 3, 5], [3, 5, 4], [2, 3]]
```

i = 0 i = 1 i = 2

Iterating through a 2D list

```
for i in range(len(lst)):
    for j in range(len(lst[i])):
        print(lst[i][j])
```

For every i:

j steps through the sub lists starting at 0

```
lst = [[7, 3, 5], [3, 5, 4], [2, 3]]
```

i = 0
j = 0

Iterating through a 2D list

```
Lst = [[7, 3, 5], [3, 5, 4], [2, 3]]

for i in range(len(lst)):

    for j in range(len(lst[i])):

        print(lst[i][j])
```

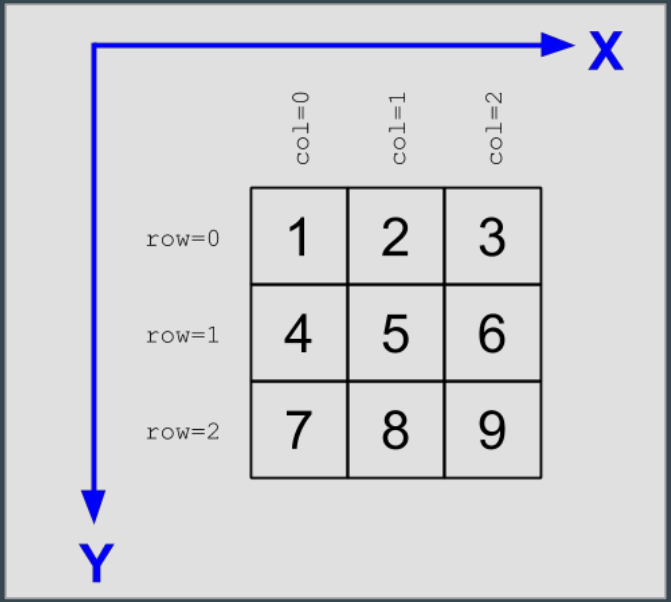
Output that's printed:

7
3
5
3
5
4
2
3

Rows and Columns

Another way to think of 2D lists is as rows and columns
We can use this to print all elements:

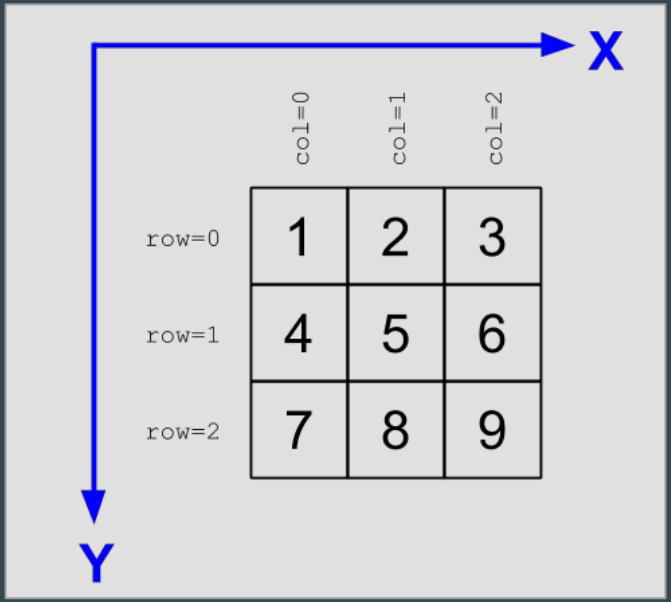
```
lst = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

for row in range(len(lst)):

    for col in range(len(row)):

        print(lst[row][col])
```



Rows and Columns

We can also use a 'for-each' loop if we don't need exact indexes like the range function:

```
lst = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
for row in lst:
    for col in row:
        print(col)
```



Reverse rows example

How can we reverse all rows in a list using a nested loop?

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

# we want to return [[3, 2, 1], [6, 5, 4], [9, 8, 7]]


def reverse_rows(matrix):

    pass
```

Reverse rows example

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

def reverse_rows(matrix):
    new_lst = []
    for row in lst:
        new_row = []
        for i in range(len(lst) - 1, -1, -1):
            new_row.append(row[i])
        new_lst.append(new_row)
```

Create Matrix Example

How can we create a matrix from a string?
Assume we are given the dimension

```
lst = “1 2 3 4 5 6 7 8 9”

# we want to return [[1, 2, 3], [4, 5, 6], [7, 8, 9]]



def create_matrix(string, width):

    pass
```

Create Matrix Example

How can we create a matrix from a string?
Assume we are given the dimension

```
lst = “1 2 3 4 5 6 7 8 9”

def create_matrix(string, width):
    matrix = []
    for i in range(0, len(lst), width / 2):
        row = []
        for j in range(i, i + width, 2):
            row.append(int(string[i][j])
        matrix.append(row)
```

Questions??