AGENDA

- For loops review
- While loops review
- Nested loops
- Matrix and 2D arrays
FOR LOOPS REVIEW

- Definite Loops are implemented with for loops.
- For loops are traditionally used when you have a piece of code which you want to repeat a fixed number of times.
- The general form of for loop:
  - For `<var>` in `<sequence>`:
    - `<body>`
  - `<var>` is called the loop index, it takes consecutive values listed in sequence.
- There are two forms of for loop:
  - `for i in range(INTEGER)`
  - `for item in (LIST/STRING)`
### DIFFERENT WAYS TO LOOP

<table>
<thead>
<tr>
<th>myList = range(3)</th>
<th>for i in range(3):</th>
</tr>
</thead>
<tbody>
<tr>
<td>for i in myList:</td>
<td></td>
</tr>
<tr>
<td>print(i)</td>
<td>print(i)</td>
</tr>
<tr>
<td></td>
<td>&gt;&gt;</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>str = 'Hello'</th>
<th>str = 'Hello'</th>
</tr>
</thead>
<tbody>
<tr>
<td>for c in str:</td>
<td>for i in range(len(str)):</td>
</tr>
<tr>
<td>print(c)</td>
<td>print(str(i))</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>&gt;&gt;</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>e</td>
<td>e</td>
</tr>
<tr>
<td>l</td>
<td>l</td>
</tr>
<tr>
<td>l</td>
<td>l</td>
</tr>
<tr>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>
WHILE LOOPS REVIEW

- The general form of a while loop:
  ```python
  while <condition>:
    <body>
  ```
- The condition is a Boolean expression.
- The while will keep looping executing the body as long as the condition is True.
COUNT TO FIVE USING FOR AND WHILE LOOPS

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

>>> 1
  2
  3
  4
  5
```

```python
i = 0
while i<5:
    i = i + 1
    print (i)

>>> 1
  2
  3
  4
  5
```
NESTED LOOPS EXAMPLE

- We can define a loop within another loop.

- Each single iteration for the outer loop, all the iterations of the inner loop will be executed.

- The number of times the outer loop is executed = \( \text{len(strList)} \rightarrow 3 \)

- The number of times the inner loop is executed = \( \text{len(strList)} \times \text{len(range(2))} \rightarrow 3 \times 2 = 6 \)

```python
strList = ['a', 'b', 'c']
for i in strList:  # outer loop
    print('In the outer loop: ', 'i=', i)
    for j in range(2):  # inner loop
        print('In the inner loop', i, j)
    print('Finished inner loop')
print('Finished BOTH loops')
```

```plaintext
In the outer loop:  i= a
In the inner loop a 0
In the inner loop a 1
Finished inner loop
In the outer loop:  i= b
In the inner loop b 0
In the inner loop b 1
Finished inner loop
In the outer loop:  i= c
In the inner loop c 0
In the inner loop c 1
Finished inner loop
Finished BOTH loops
```
NUMBER OF EXECUTIONS PER LINE IN NESTED LOOPS

<table>
<thead>
<tr>
<th>Statement</th>
<th>Number of executions</th>
</tr>
</thead>
<tbody>
<tr>
<td>strList = ['a','b','c']</td>
<td>1</td>
</tr>
<tr>
<td>for i in strList:</td>
<td>3</td>
</tr>
<tr>
<td>print('In the outer loop: ', 'i=',i)</td>
<td>3</td>
</tr>
<tr>
<td>for j in range(2):</td>
<td>6</td>
</tr>
<tr>
<td>print('In the inner loop', i,j)</td>
<td>6</td>
</tr>
<tr>
<td>print ('Finished inner loop')</td>
<td>3</td>
</tr>
<tr>
<td>print ('Finished BOTH loops')</td>
<td>1</td>
</tr>
</tbody>
</table>
ANOTHER NESTED LOOPS EXAMPLE

• The number of times the print statement is executed = $\text{len(range(4))}\times\text{len(range(2))}=8$

```python
for i in range(4):  #The outer loop
    for j in range(2):  #The inner loop
        print(i,j)

> 0 0
0 1
1 0
1 1
2 0
2 1
3 0
3 1
```
NESTED LISTS NEED NESTED LOOPS

- Nested loops are suitable when working with nested lists. E.g. Given a nested list, print the sum of each inner list.

```python
def main():
    myList = [[1, 2, 3], [10, -5, 20], [40]]
    for lst in myList:
        sum = 0
        for number in lst:
            sum = sum + number
        print(sum)
main()

>>
6
30
40
```
DIFFERENCE BETWEEN BREAK AND CONTINUE

- Break terminates the loop
- Continue terminates the current iteration ONLY.

```python
xlist = [2, 4, -1, 8]
for num in xlist:
    if (num < 0):
        break
    print(num)
```

```
xlist = [2, 4, -1, 8]
for num in xlist:
    if (num < 0):
        continue
    print(num)
```

```
>>
2
4
```

```
>>
2
4
8
```
BREAK & CONTINUE IN NESTED LOOPS

- Given a nested list, add all **positive** numbers per inner list and print the sum for each inner list
- Which of these two pieces of code is correct?

```python
def main():
    myList = [[1, 2, 3], [10, -5, 20], [40]]
    for lst in myList:
        sum = 0
        for number in lst:
            if number < 0:
                continue
            sum = sum + number
        print (sum)

main()
```

```python
def main():
    myList = [[1, 2, 3], [10, -5, 20], [40]]
    for lst in myList:
        sum = 0
        for number in lst:
            if number < 0:
                break
            sum = sum + number
        print (sum)

main()
```
**BREAK & CONTINUE IN NESTED LOOPS**

- Given a nested list, add all **positive** numbers per inner list and print the sum for each inner list.
- Which of these two pieces of code is correct?

```python
def main():
    myList = [[1, 2, 3], [10, -5, 20], [40]]
    for lst in myList:
        sum = 0
        for number in lst:
            if (number < 0):
                continue
            sum = sum + number
        print(sum)
main()
>>
6
30
40
```

**CORRECT**

```python
def main():
    myList = [[1, 2, 3], [10, -5, 20], [40]]
    for lst in myList:
        sum = 0
        for number in lst:
            if (number < 0):
                break
            sum = sum + number
        print(sum)
main()
>>
6
10
40
```

**WRONG**
MATRICES AND 2D ARRAYS
Matrices in mathematics are arrays of numbers or variables arranged in both rows and columns.

Each number or variable contained within the matrix can be identified by its position in the row and column.
Each element of the matrix has a unique position determined by an index \( i \) and at index \( j \).

In the matrix below: the number 1, is defined to be in position 0,0 (located in row index 0 and column index 0).

What are the indexes of number 14 in this matrix?
MATRIX INDEXING EXERCISE

Indexing in the matrix:

```python
>>> myMatrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]
>>> print(myMatrix[0][0])
1
>>> print(myMatrix[3][2])
12
>>> print(myMatrix[1][2])
6
>>> print(myMatrix[2][0])
7
>>> print(myMatrix[2][3])
Traceback (most recent call last):
  File "<pyshell#5>"", line 1, in <module>
    print(myMatrix[2][3])
IndexError: list index out of range
```
MATRICES ARE REPRESENTED IN PYTHON USING NESTED LISTS

```python
>>> myMatrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]
>>> numRows = len(myMatrix)
>>> numColumns = len(myMatrix[0])
>>> print(numRows)
4
>>> print(numColumns)
3
```

CREATE A 5X4 MATRIX M POPULATED WITH ZEROES.

rows = 5
columns = 4
M = []
for i in range(rows):
    rowList = []
    for j in range(columns):
        rowList.append(0)
    M.append(rowList)

>>> print(M)
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
TRAVERSING AND PRINTING A MATRIX

Write a code to print the contents of a given matrix so that the output is formatted in rows and columns as shown in the figure.

We’re going to need nested loops to iterate through each row and column

```python
myMatrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]
for i in range(0, len(myMatrix)):
    if i != 0:
        print()
    for j in range(0, len(myMatrix[0])):
        print(str(myMatrix[i][j]) + ' ' * (10 - len(str(myMatrix[i][j]))), end = '')
```

TRaversing AND PRINTING A MATRIX

Write a code to print the contents of a given matrix so that the output is formatted in rows and columns as shown in the figure.

We’re going to need nested loops to iterate through each row and column.

```python
myMatrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]
for i in range(len(myMatrix)):
    if i != 0:
        print()
    for j in range(len(myMatrix[0])):
        print(str(myMatrix[i][j]) + ' ', end = '')
```

#rows = len(myMatrix)

TRaversing and printing a matrix

Write a code to print the contents of a given matrix so that the output is formatted in rows and columns as shown in the figure.

We’re going to need nested loops to iterate through each row and column

```python
myMatrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]
for i in range(0, len(myMatrix)):
    if(i != 0):
        print()
    for j in range(0, len(myMatrix[0])):
        print(str(myMatrix[i][j]) + 't', end = '')
```

#columns = len(myMatrix[0]) = len(myMatrix[1]) = len(myMatrix[2])
TRACING AND PRINTING A MATRIX

Write a code to print the contents of a given matrix so that the output is formatted in rows and columns as shown in the figure.

We’re going to need nested loops to iterate through each row and column

```python
myMatrix = [[1,2,3],[4,5,6],[7,8,9],[10,11,12]]
for i in range(0,len(myMatrix)):
    #if(i != 0):
        # print()
    for j in range(0,len(myMatrix[0])):
        print(str(myMatrix[i][j])+'\t',end = "")
```

Removing these lines
TRAVERSING AND PRINTING A MATRIX

Write a code to print the contents of a given matrix so that the output is formatted in rows and columns as shown in the figure.

We’re going to need nested loops to iterate through each row and column

myMatrix = [[1,2,3],[4,5,6],[7,8,9],[10,11,12]]
for i in range(0,len(myMatrix)):
    if(i != 0):
        print()
    for j in range(0,len(myMatrix[0])):
        #print(str(myMatrix[i][j])+\t',end = "")
        print(str(myMatrix[i][j])+\t')
CREATING A SPECIAL MATRIX

Exercise 3:
Write a code for creating a matrix with 5 rows and 4 columns. Filled with zeroes except for the elements on the diagonal be filled with ones. Diagonal elements are those having the row index equals to the column index.
CREATING A SPECIAL MATRIX

Exercise 3:
Write a code for creating a matrix with 5 rows and 4 columns. Filled with zeroes except for the elements on the diagonal be filled with ones.
Diagonal elements are those having the row index equals to the column index.

```python
rows = 5
columns = 4
Matrix = []
for i in range(rows):
    rowList = []
    for j in range(columns):
        if(i == j):
            rowList.append(1)
        else:
            rowList.append(0)
    Matrix.append(rowList)
```

```
1  0  0  0  0
0  1  0  0  0
0  0  1  0  0
0  0  0  0  1
0  0  0  0  0
```
EXERCISE:
WHAT SHOULD BE THE OUTPUT OF THE FOLLOWING CODE?

```python
rows = 5
columns = 5
M = []
for i in range(rows):
    rowList = []
    for j in range(columns):
        rowList.append(0)
    M.append(rowList)
for i in range(rows):
    if(i!= 0):
        print()
    for j in range(columns):
        if i+j == 4:
            M[i][j] = 1
            print(str(M[i][j])+'\t',end = '')
```

Creating 5x5 all zeroes matrix M
EXERCISE:
WHAT SHOULD BE THE OUTPUT OF THE FOLLOWING CODE?

```python
rows = 5
columns = 5
M = []
for i in range(rows):
    rowList = []
    for j in range(columns):
        rowList.append(0)
    M.append(rowList)
for i in range(rows):
    if (i != 0):
        print()
    for j in range(columns):
        if i+j == 4:
            M[i][j] = 1
            print(str(M[i][j]) + '\t', end = '')
```

Creating 5x5 all zeroes matrix M

```
0 0 0 0 1
0 0 0 1 0
0 0 1 0 0
0 1 0 0 0
1 0 0 0 0
```
The transpose of matrix $A$ is another matrix $B$ where the rows of $A$ are the columns of $B$, and the columns of $A$ in the rows of $B$.

Example:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}^T = \begin{pmatrix} 1 \\ 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$
WRITE A FUNCTION THAT TAKES A MATRIX $M$ AND RETURNS MATRIX $T$ THE TRANSPOSE OF MATRIX $M$.

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

$M$

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>7</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
</tr>
</tbody>
</table>

$T$
def transpose(M):
    rows = len(M)
    columns = len(M[0])
    transposeRows = columns
    transposeColumns = rows
    T = []
    for i in range(transposeRows):
        rowList = []
        for j in range(transposeColumns):
            rowList.append(0)
        T.append(rowList)
    for i in range(rows):
        for j in range(columns):
            T[j][i] = M[i][j]
    return T

Creating all zeroes transpose matrix T

Swapping items on rows of M to the columns of T
WRITE A FUNCTION THAT TAKES A MATRIX M AND RETURNS MATRIX T THE TRANSPOSE OF MATRIX M.

```python
def transpose(M):
    rows = len(M)
    columns = len(M[0])
    transposeRows = columns
    transposeColumns = rows
    T = []
    for i in range(transposeRows):
        rowList = []
        for j in range(transposeColumns):
            rowList.append(0)
        T.append(rowList)
    for i in range(rows):
        for j in range(columns):
            T[j][i] = M[i][j]
    return T
```

What do you think the following line will do?

```python
transpose(transpose(M))
```
Matrix multiplication is an important mathematical operation.

The result of multiplying two 2x2 matrices is a third 2x2 matrix.

Example:

\[ \begin{bmatrix} 1 & 0 \\ -3 & 2 \end{bmatrix} \quad \begin{bmatrix} -1 & 4 \\ 3 & 5 \end{bmatrix} \]

we need to multiply A, B to get matrix M

To calculate the elements in M:

\[ M[i][j] = \text{The row } A[i] \times \text{The column } B[j] \]

Final Answer:

\[ \begin{bmatrix} -1 & 4 \\ 9 & -2 \end{bmatrix} \]
MATRIX MULTIPLICATION

A = [[1,0],[-3,2]]
B = [[-1,4],[3,5]]
rows = 2
columns = 2

M = []
for i in range(rows):
    rowList = []
    for j in range(columns):
        rowList.append(0)
    M.append(rowList)
for i in range(rows):
    for j in range(columns):
        for k in range(rows):
            M[i][j] = M[i][j] + A[i][k]*B[k][j]

Creating all zeroes matrix M

Calculating each element in M

Final Answer: 
\[
\begin{bmatrix}
-1 & 4 \\
9 & -2
\end{bmatrix}
\]
QUESTIONS ?