Title: 7.2 Exercises

Author: Chad Wood

Date: 2 Feb 2022

Modified By: Chad Wood

Description: This program demonstrates the use of python to perform mathematical calculations on matrices.

Working with Matrices

Let A, B, and C be the matrices given below.

$$A = \begin{bmatrix} 1 & -2 & 3 & 7 \\ 2 & 1 & 1 & 4 \\ -3 & 2 & -2 & 10 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 1 & -3 & -2 \\ 10 & -1 & 2 & -3 \\ 5 & 1 & -1 & 4 \end{bmatrix} \qquad C = \begin{bmatrix} 4 & 0 & -2 & 3 \\ 3 & 6 & 9 & 7 \\ 2 & 2 & 5 & 1 \\ 9 & 4 & 6 & -2 \end{bmatrix}$$

1. Enter each of the matrices A, B, and C as Numpy matrix.

In [8]:
```
import numpy as np

# Builds matrices
A = np.matrix('1 -2 3 7; 2 1 1 4; -3 2 -2 10')
B = np.matrix('0 1 -3 -2; 10 -1 2 -3; 5 1 -1 4')
C = np.matrix('4 0 -2 3; 3 6 9 7; 2 2 5 1; 9 4 6 -2')
```

2. What are the dimensions of each of the matrices A, B, and C? Display these using Python code.

In [12]:
```
# Uses shape to return tuple of row count, col count
print(f'Matrix: A {A.shape}, B {B.shape}, C {C.shape}')
```

```
Matrix: A (3, 4), B (3, 4), C (4, 4)
```

3. Determine if each of the following matrix operations is defined. If it is defined, calculate the result using Python. If it is not defined, explain why it is not defined.

   (a) A + B

   (b) AB

   (c) AC

   (d) C^T

In [29]:
```
# (a) is defined, as seen via output:
print('A+B', A+B)
print('-'*50)

# (b) is undefined for the following reason:
print('A*B is undefined because matrix multiplication can be described as (r1, c1) * (r2, c2) where r1 is multiplied by c2.',
      'Because A contains 3 rows and B contains 4 columns, they can not be multiplied.')
print('-'*50)

# (c) is defined, as seen via output:
print('A*C', A*C)
print('-'*50)

# (d) is defined, as its merely a transposition of the data
print('C Transposed:', C.T)
```

```
A+B [[ 1 -1  0  5]
 [12  0  3  1]
 [ 2  3 -3 14]]
--------------------------------------------------
A*B is undefined because matrix multiplication can be described as (r1, c1) * (r2, c2) where r1 is multiplied by c2. Because A
contains 3 rows and B contains 4 columns, they can not be multiplied.
--------------------------------------------------
A*C [[ 67  22  37 -22]
 [ 49  24  34   6]
 [ 80  48  74 -17]]
--------------------------------------------------
C Transposed: [[ 4  3  2  9]
 [ 0  6  2  4]
 [-2  9  5  6]
 [ 3  7  1 -2]]
```

4. Illustrate the following property of matrix transpositions using these matrices: $(A + B)^T = A^T + B^T$.

In [31]:
```
# Performs both calculations, then tests if they are equal

trans1 = (A+B).T
trans2 = A.T + B.T

# Uses == to return true or false for each position
print(f'The transpositions are equal:\n {trans1==trans2}')
```

```
The transpositions are equal:
 [[ True  True  True]
 [ True  True  True]
 [ True  True  True]
 [ True  True  True]]
```

5. Illustrate the following property of matrix transposition using these matrices $(AC)^T = C^T A^T$.

In [32]:
```
# Performs both calculations, then tests if they are equal

trans1 = (A*C).T
trans2 = C.T * A.T

# Uses == to return true or false for each position
print(f'The transpositions are equal:\n {trans1==trans2}')
```

```
The transpositions are equal:
 [[ True  True  True]
 [ True  True  True]
 [ True  True  True]
 [ True  True  True]]
```

6. Find C^{-1} using Python code.

In [37]:
```
# ** is power. Reproduces the problem given
print(C**-1)
print('-'*50)

# Numpys matrix inversion function to compare results:
print(np.linalg.inv(C))

# Tests
C**-1 == np.linalg.inv(C)
```

```
[[ 0.14814815 -0.07407407  0.14814815  0.03703704]
 [-0.3         0.35       -1.05        0.25      ]
 [ 0.02962963 -0.11481481  0.52962963 -0.09259259]
 [ 0.15555556  0.02222222  0.15555556 -0.11111111]]
--------------------------------------------------
[[ 0.14814815 -0.07407407  0.14814815  0.03703704]
 [-0.3         0.35       -1.05        0.25      ]
 [ 0.02962963 -0.11481481  0.52962963 -0.09259259]
 [ 0.15555556  0.02222222  0.15555556 -0.11111111]]
```

Out[37]:
```
matrix([[ True,  True,  True,  True],
        [ True,  True,  True,  True],
        [ True,  True,  True,  True],
        [ True,  True,  True,  True]])
```

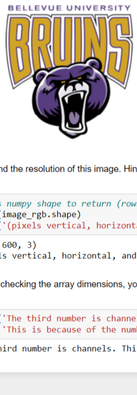
Working with Images

For this exercise, you will be working with the image file week7 image.jpg.

1. Use the following code to display the image.

```
#import the OpenCV library to read images
import cv2
#import matplotlib to display the image
import matplotlib.pyplot as plt

#import the image; by default, OpenCV imports images in the BGR format
image_bgr = cv2.imread('week7_image.jpg', cv2.IMREAD_COLOR) #you may need to modify your file path
#convert the image to the RGB formate
image_rgb = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)

#display the image
plt.imshow(image_rgb)
plt.axis("off")
plt.show()
```

In [39]:
```
import cv2 # Import OpenCV to read images
import matplotlib.pyplot as plt # matplotlib to display image

# Import image (Defaults to BGR)
image_bgr = cv2.imread('week7data/week7_image.jpg', cv2.IMREAD_COLOR)
# Converts to RGB
image_rgb = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)

# Displays image
plt.imshow(image_rgb)
plt.axis('off')
plt.show()
```



2. Find the resolution of this image. Hint: OpenCV imports the image as a Numpy array, where each pixel corresponds to an entry in the array.

In [43]:
```
# Uses numpy shape to return (rows, columns) since pixels correspond with entrees
print(image_rgb.shape)
print('(pixels vertical, horizontal, and channels)')
```

```
(453, 600, 3)
(pixels vertical, horizontal, and channels)
```

3. In checking the array dimensions, you should see that three numbers are displayed. What is this third number, and why is it there?

In [45]:
```
print('The third number is channels.',
      'This is because of the number of colors available in RGB (256x256x256) totaling 16,777,216')
```

```
The third number is channels. This is because of the number of colors available in RGB (256x256x256) totaling 16,777,216
```