Logan Wright

Brother Sinkovic

Python Assignment 1

Part I

Question 1

I will write some code to build a matrix, A, and write a corresponding command to print the i^{th} entry of the j^{th} column of A.

In [6]:
```
from numpy import array

A = array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [2, 4, 6]], dtype = float)

print(A)
print(f"\nA @ position (4, 2): {A[3, 1]}")
```
```
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]
 [2. 4. 6.]]

A @ position (4, 2): 4.0
```

Question 2

A new row is entered into the matrix A as it replaces the values that were in row 3.

In [4]:
```
A[2] = [7, 8, 10]

print(A)
```
```
[[ 1.  2.  3.]
 [ 4.  5.  6.]
 [ 7.  8. 10.]
 [ 2.  4.  6.]]
```

Question 3

In the following lines of code, we do the following things to A

1. $R_2 + (-4R_1)$
2. $R_3 + (-7R_1)$
3. $R_3 + \left(-\frac{8}{5}R_2\right)$

In [13]:
```
A = array([[1,2,3],[4,5,6],[7,8,9]], float)
print(A)
print()

A[1] = A[1] + A[0] * (-A[1,0])
A[2] = A[2] + A[0] * (-A[2,0])
print(A)
print()

A[2] = A[2] + A[1] * (-A[2,1] / A[1,1])
print(A)
```
```
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]

[[  1.   2.   3.]
 [  0.  -3.  -6.]
 [  0.  -6. -12.]]

[[ 1.  2.  3.]
 [ 0. -3. -6.]
 [ 0.  0.  0.]]
```

Question 4

Now, I will use the NumPy linear algebra solve function to solve a basic equation $Ax = b$.

In [14]:
```
from numpy.linalg import solve

A = array([[1, 2, 3], [4, 5, 6], [7, 8, 10]], dtype = float)
b = array([[-1], [4], [-10]], dtype = float)

print("The solution to Ax = b\n")

print(solve(A, b))
```
```
The solution to Ax = b

[[-14.66666667]
 [ 35.33333333]
 [-19.        ]]
```

Question 5

Now, I experiment with the NumPy matrix multiplication tool called "matmul". I will compute AA^T for the test.

In [17]:
```
from numpy import matmul

print(f"A A^T\n")
print(matmul(A, A.transpose()))
```
```
A A^T

[[ 14.  32.  53.]
 [ 32.  77. 128.]
 [ 53. 128. 213.]]
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R) SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX) instructions.
```

Question 6

Now, I will use the NumPy matrix inverse function to compute the same $Ax = b$ problem as above but instead by saying that $x = A^{-1}b$.

In [20]:
```
from numpy.linalg import inv

A_inverse = inv(A)

print("The inverse of A\n")
print(A_inverse)

print(f"\nThe solution to Ax = b:\n")
print(matmul(A_inverse, b))
```
```
The inverse of A

[[-0.66666667 -1.33333333  1.        ]
 [-0.66666667  3.66666667 -2.        ]
 [ 1.         -2.          1.        ]]

The solution to Ax = b:

[[-14.66666667]
 [ 35.33333333]
 [-19.        ]]
```

Question 7

Next, I solve the coefficients of an 5^{th} degree polynomial given the following points

(1, 1)

(10, 5)

(−10, 6)

(2, 100)

(−2, −30)

(−1, 60)

In [32]:
```
# Make B a function to generate the rows of A
B = lambda x: [x ** 5, x ** 4, x ** 3, x ** 2, x, 1]

# Define the x values to range from
x = [1, 10, -10, 2, -2, -1]

# Make the A matrix and fill it
A = []
for i in range(6):
    A.append(B(x[i]))

# Convert A to an array
A = array(A, dtype = float)

# Define a new b vector based on the points defined above
b = array([[1], [5], [6], [100], [-30], [60]], dtype = float)

print("Coefficient solutions to Ax = b\n")
print(solve(A, b))
```
```
Coefficient solutions to Ax = b

[[-2.12179082e-01]
 [-1.82554714e-02]
 [ 2.17275621e+01]
 [ 1.59127736e+00]
 [-5.10153830e+01]
 [ 2.89269781e+01]]
```