Week 9 - NumPy

In [1]:
```
import numpy as np
```

Exercise 1 - Creating arrays and accessing elements

Question 1.

In [2]:
```
a = np.array([5, 4, 9, 2, 0, 4, 7, 2])
```

Question 2.

In [3]:
```
# print the first entry
print(a[0])

# print the last entry
print(a[-1])

#print the second-to-sixth entries using colon notation
print(a[1:6])
```

```
5
2
[4 9 2 0 4]
```

Question 3.

In [4]:
```
# changing the last entry to 9
a[-1] = 9
print(a)

# changing the first three entries to 1
a[0:3] = 1
print(a)
```

```
[5 4 9 2 0 4 7 9]
[1 1 1 2 0 4 7 9]
```

Question 4.

In [5]:
```
r = np.random.randint(1, 10, 20)
print(r)
```

```
[7 9 3 1 1 6 3 9 2 3 6 5 3 4 5 2 1 2 7 3]
```

Question 5.: Running `idx = r < 5` creates an array of True/False values. The values of True occur in places where the entries of r satisfy $r < 5$. The values of False occur in places where the entries of r satisfy $r \geq 5$. Running `r[idx] = 0` assigns the value of 0 to entries of r in places where `idx` has the value True. The final result is that entries of r that are smaller than 5 are set to 0.

In [6]:
```
idx = r < 5
print(idx)

r[idx] = 0
print(r)
```

```
[False False  True  True  True False  True False  True  True False False
  True  True False  True  True  True False  True]
[7 9 0 0 0 6 0 9 0 0 6 5 0 0 5 0 0 0 7 0]
```

Question 6.

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

```
[[6 2 3]
 [4 4 1]
 [8 5 6]]
```

Question 7.

In [8]:
```
# updating the entry in the second row, first column
A[1,0] = 9

# updating the entry in the last row and last column
A[-1,-1] = 0

# printing the updated array
print('A =', A, '\n')

# printing all entries in the second row
print('The entries in the second row are', A[1,:])
```

```
A = [[6 2 3]
 [9 4 1]
 [8 5 0]]

The entries in the second row are [9 4 1]
```

Question 8.

In [9]:
```
# creating an array of zeros
A = np.zeros((2, 2))

# setting the values of the rows
A[0,:] = 1
A[1,:] = 2

# printing the result
print('A =', A, '\n')

# now doing the case when N = 5
N = 5
B = np.zeros((5, 5))
for i in range(5):
    B[i,:] = i + 1
print('B = ', B)
```

```
A = [[1. 1.]
 [2. 2.]]

B =  [[1. 1. 1. 1. 1.]
 [2. 2. 2. 2. 2.]
 [3. 3. 3. 3. 3.]
 [4. 4. 4. 4. 4.]
 [5. 5. 5. 5. 5.]]
```

Exercise 2 - Performing operations on arrays

Question 1.

In [10]:
```
a = np.array([3, 5, 2])
b = np.array([6, 3, 1])

# calculating c
c = a + 2 * b
print('c = ', c)

# calculating the dot product between a and b using the 'dot' method
print('a.b =', a.dot(b))

# calculating the dot product using the dot function
print('a.b =', np.dot(a, b))

# calculating the dot product using element-by-element multiplication and np.sum
print('a.b =', np.sum(a * b))
```

```
c =  [15 11  4]
a.b = 35
a.b = 35
a.b = 35
```

Question 2.

In [11]:
```
# creating an array of t values using linspace
t = np.linspace(0, 5, 500)

# creating an array to store y(t)
y = t**2 * np.exp(-2 * t)

# finding the max value of y
y_max = np.max(y)
print('the max of y is', y_max)

# finding the value of t where y reaches its max
t_max = t[y == y_max]
print('y is maximal when t =', t_max[0])
```

```
the max of y is 0.1353347404497369
y is maximal when t = 1.002004008016032
```

Question 3.

In [12]:
```
# computing I numerically
x = np.linspace(0, 5, 50)
y = x / (1 + x)
I_approx = np.trapz(y, x)
print('The approximate value of I is', I_approx)


# repeat the calculation with 500 points between 0 and 5
x = np.linspace(0, 5, 500)
y = x / (1 + x)
I_approx2 = np.trapz(y, x)
print('The new approximation to I is', I_approx2)

# the approximation has become more accurate
```

```
The approximate value of I is 3.207397837710025
The new approximation to I is 3.208232396499138
```

Question 4.

In [13]:
```
g = [3.7, 8.9, 9.8, 3.7, 25, 10, 8.9, 11]

print('The min is', np.min(g))
print('The max is', np.max(g))
print('The mean is', np.mean(g))
print('The median is', np.median(g))
```

```
The min is 3.7
The max is 25.0
The mean is 10.125
The median is 9.350000000000001
```

Question 5.

In [14]:
```
A = np.array([[1, 2, 3], [3, 2, 1], [2, 4, 6]])
B = np.array([[1, 5, 0], [0, 1, 1], [4, 3, 1]])

# computing C = A + 2B
C = A + 2 * B
print('C =', C, '\n')

# computing AB and BA
print('AB =', A @ B, '\n')
print('BA =', B @ A, '\n')
```

```
C = [[ 3 12  3]
 [ 3  4  3]
 [10 10  8]]

AB = [[13 16  5]
 [ 7 20  3]
 [26 32 10]]

BA = [[16 12  8]
 [ 5  6  7]
 [15 18 21]]
```

Question 6.

In [15]:
```
# computing the transpose of A
A_trans = np.transpose(A)

# printing the result
print('The transpose of A is\n', A_trans, '\n')
```

```
The transpose of A is
 [[1 3 2]
 [2 2 4]
 [3 1 6]]
```

Question 7.

In [16]:
```
A = np.array([[1, 0, 0, -1], [1, -2, 1, 0], [0, 1, -2, 1], [2, 0, 0, 1]])
b = np.array([0, 1, 1, 2])

x = np.linalg.solve(A, b)
print('x =', x)
print('Ax - b = ', A @ x - b)
```

```
x = [ 0.66666667 -0.33333333 -0.33333333  0.66666667]
Ax - b =  [1.11022302e-16 2.22044605e-16 0.00000000e+00 0.00000000e+00]
```

Exercise 3 - Weather prediction

Question 1

In [17]:
```
# create the transition matrix
P = np.array([[0.5, 0.3, 0.2], [0.4, 0.2, 0.4], [0.6, 0.2, 0.2]])

# create the state vector for today
x_0 = np.array([1, 0, 0])

# compute the state vector for tomorrow
x_1 = x_0 @ P

print('The probability that it will be sunny tomorrow is', x_1[0])
```

```
The probability that it will be sunny tomorrow is 0.5
```

Question

In [18]:
```
x_2 = x_1 @ P
print('The probability that it will rain in two days is', x_2[-1])
```

```
The probability that it will rain in two days is 0.26
```

Question

In [19]:
```
# create the state vector for today
x = np.array([1, 0, 0])

# use a for loop to repeatedly calculate xP
for n in range(7):
    x = x @ P

print('The seven-day forecast is')
print('Sunny:', round(x[0], 2))
print('Cloudy:', round(x[1], 2))
print('Rainy:', round(x[2], 2))
```

```
The seven-day forecast is
Sunny: 0.5
Cloudy: 0.25
Rainy: 0.25
```