IT2313 - Programming for Data Science

Practical 4

NumPy for Numerical Computations (Part 2)

In this lesson, we will continue to practise our NumPy knowledge.

- Shapes and Operation
- Basic Math Operation
- Matrix and Manipulation
- Broadcasting
- Case Study with actual dataset
- Some challenging concepts and questions

We need to import the neccessary libraries

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

Q1 - Create an array with sequence of alternate numbers from 0 to 18 and save it to A

In [49]:
```
# Question 1 - Type your answers here
```

```
[ 0  2  4  6  8 10 12 14 16]
```

Q2 - Print the shape of the array A

In [50]:
```
# Question 2 - Type your answers here
```

```
(9,)
```

Q3 - Create an array with sequence of numbers from 1 to 9 and save it to B, Reshape array B to a 3 by 3 array and display array B

In [51]:
```
# Question 3 - Type your answers here
```

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

Q4 - Print the shape of the array B

In [52]:
```
# Question 4 - Type your answers here
```

```
(3, 3)
```

Q5 - Let create the following arrays

[[0 1] [2 3]]

[[10.1 13.4] [15.5 21.3]]

In [53]:
```
# Question 5 - Type your answer here
```

```
[[0 1]
 [2 3]]
[[10.1 13.4]
 [15.5 21.3]]
```

Q6 - Using the above array,

- (a) sum up array x and y using the "+" operator and add function
- (b) subtract array y from x using the "-" operator and subtract function
- (c) multiply array x and y using the "*" operator and multiply function
- (d) divide up y from x using the "/" operator and divide function

```
In [54]: # Question Q6(a) - Type your answer here
```

```
[[10.1 14.4]
 [17.5 24.3]]

[[10.1 14.4]
 [17.5 24.3]]
```

```
In [55]: # Question Q6(b) - Type your answer here
```

```
[[-10.1 -12.4]
 [-13.5 -18.3]]

[[-10.1 -12.4]
 [-13.5 -18.3]]
```

```
In [56]: # Question Q6(c) - Type your answer here
```

```
[[ 0.   13.4]
 [31.   63.9]]

[[ 0.   13.4]
 [31.   63.9]]
```

```
In [57]: # Question Q6(d) - Type your answer here
```

```
[[0.         0.07462687]
 [0.12903226 0.14084507]]

[[0.         0.07462687]
 [0.12903226 0.14084507]]
```

Q7 - Let use the following command to generate a 5x5 matrix

mat = np.arange(1, 26).reshape(5, 5)

Flatten the array, shuffle it, and then reshape it back

flat_mat = mat.flatten() np.random.shuffle(flat_mat) mat = flat_mat.reshape(5, 5)

- (a) Compute the mean, median, standard deviation of a numpy array in each of the rows
- (b) Compute the mean, median, standard deviation of a numpy array in each of the columns
- (c) Compite the mean, median, standard deviation for the matrix

```
In [58]: # Set a random seed for reproducibility
         # This is to ensure every run will produce the same random number
         seed_value = 42
         np.random.seed(seed_value)

         mat = np.arange(1, 26).reshape(5, 5)
         flat_mat = mat.flatten()
         np.random.shuffle(flat_mat)
```

```
mat = flat_mat.reshape(5, 5)
print(mat)
```

```
[[ 9 17  1 24 12]
 [10 14  2 23  6]
 [ 3 13 16  4  5]
 [21 18 22 19 25]
 [ 8 11 15 20  7]]
```

In [59]:
```
# Question Q7(a) - Type your answers here
# Compute mean, median, and standard deviation along each row
mean_values =
median_values =
std_dev_values =

print("Mean along each row:", mean_values)
print("Median along each row:", median_values)
print("Standard deviation along each row:", std_dev_values)
```

```
Mean along each row: [12.6 11.   8.2 21.  12.2]
Median along each row: [12. 10.  5. 21. 11.]
Standard deviation along each row: [7.70973411 7.21110255 5.26877595 2.44948974 4.79165942]
```

In [60]:
```
# Question Q7(b) - Type your answers here
# Compute mean, median, and standard deviation along each column
mean_values =
median_values =
std_dev_values =

print("Mean along each column:", mean_values)
print("Median along each column:", median_values)
print("Standard deviation along each column:", std_dev_values)
```

```
Mean along each column: [10.2 14.6 11.2 18.  11. ]
Median along each column: [ 9. 14. 15. 20.  7.]
Standard deviation along each column: [5.9126982  2.57681975 8.28009662 7.23878443 7.40270221]
```

In [61]:
```
# Question Q7(c) - Type your answers here
# Compute mean, median, and standard deviation for the entire matrix
mean_value =
median_value =
std_dev_value =

print("Mean of the matrix:", mean_value)
print("Median of the matrix:", median_value)
print("Standard deviation of the matrix:", std_dev_value)
```

```
Mean of the matrix: 13.0
Median of the matrix: 13.0
Standard deviation of the matrix: 7.211102550927978
```

Q8 - Let use the following command to generate a 2x2 matrix.

The student just need to know the application in Python.

mat = np.arange(1,5).reshape(2,2)

- (a) Compute the Matrix Transpose
- (b) Compute the Matrix Power
- (c) Compute the Matrix Rank
- (d) Compute the Matrix Determinant
- (e) Compute the Eijenvalues and Eigenvectors

Reference Notes:

For (a) Transpose is "Flipping" a matrix over its diagonal. The rows and columns get swapped. https://www.mathsisfun.com/definitions/transpose-matrix-.html

For (b) Raising a matrix to a power involves multiplying the matrix by itself a certain number of times. The power of a matrix is similiar to the concept of exponentiation for numbers. https://www.mathsisfun.com/algebra/matrix-multiplying.html

For (c) The rank is how many of the rows are "unique": not made of other rows. (Same for columns.) https://www.mathsisfun.com/algebra/matrix-rank.html

For (d) The determinant is a special number that can be calculated from a matrix https://www.mathsisfun.com/algebra/matrix-determinant.html

For (e) The Eigenvector and Eigenvalue https://www.mathsisfun.com/algebra/eigenvalue.html

In [62]:
```
mat = np.arange(1,5).reshape(2,2)
mat
```

Out[62]:
```
array([[1, 2],
       [3, 4]])
```

In [63]:
```
# Question Q8(a) - Type your answers here
```

```
[[1 3]
 [2 4]]
```

In [64]:
```
# Question Q8(b) - Type your answers here
# Specify the power
```

```
[[ 7 10]
 [15 22]]
```

In [65]:
```
# Question Q8(c) - Type your answers here
```

```
2
```

In [66]:
```
# Question Q8(d) - Type your answers here
# Calculate the determinant
```

```
-2.0000000000000004
```

In [67]:
```
# Question Q8(e) - Type your answers here
# Calculate the Eigenvalues and Eigenvectors
```

```
Eigenvalues:  [-0.37228132  5.37228132]
Eigenvectors:  [[-0.82456484 -0.41597356]
 [ 0.56576746 -0.90937671]]
```

Q9 - You are given any array A of shape (3,1) and Array B of shape(4,1).

If you add them up using the following code, A + B = C, what will be shape of Array C?

Can you explain how the Broadcasting rules are being applied?

In [2]:
```
# Question 9  - Type your answers here
import numpy as np

# Create arrays A and B
A = np.array([[1], [2], [3]])       # Shape (3, 1)
B = np.array([[4], [5], [6], [7]])  # Shape (4, 1)

# Attempt to add the two arrays
try:
    C = A + B
    print("Array C:\n", C)
except ValueError as e:
    print("Error:", e)
```

```
Error: operands could not be broadcast together with shapes (3,1) (4,1)
```

Question 9 - Type your answers here

Initial Shapes

- Array A:
- Array B:

Broadcasting Rules

Compare shapes from the last dimension to the first:

- A:
- B:

Shapes are Compared

- Last dimension:
- Second Last dimension:

Q10 - Change the array of floats to an array of (a) integers and (b) strings.

arr = np.array([1.3, 2.2, 3.1])

In [69]:
```
# Question 10(a) - Type your answers here
```

```
[1 2 3]
int32
```

In [70]:
```
# Question 10(b) - Type your answers here
```

```
['1.3' '2.2' '3.1']
<U32
```

Question 11

Write a Python program that does the following:

- Imports the NumPy library.
- Creates a list named num_list containing the values [10, 20, 30, 40, 50].
- Uses the np.random.choice() function from the NumPy library to randomly select a single element from num_list.
- Prints the selected element.

In [71]:
```
# Question 11 - Type your answers here
```

```
40
```

Question 12

Write a Python program that does the following:

- Imports the NumPy library.
- Creates a list named num_list containing the values [10, 20, 30, 40, 50].

- Uses the np.random.choice() function from the NumPy library to select 3 elements from num_list with replacement.
- Assigns a probability distribution p = [0, 0, 0, 1, 0] such that only the fourth element in num_list (40) is selected each time.
- Prints the resulting array

```
In [72]: # Question 12 - Type your answers here
```

```
[40 40]
```

Question 13

Write a Python program that does the following:

- Imports the NumPy library.
- Creates a list named num_list containing the values [10, 20, 30, 40, 50].
- Uses the np.random.choice() function from the NumPy library to select 3 elements from num_list with replacement.
- Sets a probability distribution p = [0, 0, 0.5, 0.5, 0] such that only the elements 30 and 40 are selected, each with an equal probability of 50%.
- Prints the resulting array.

```
In [73]: # Question 13 - Type your answers here
```

```
[30 40 30]
```

Question 14

Write a Python code that does the following:

- Initialize a 1D NumPy array named ini_array1 with the following values: [1, 2, -3, 4, -5, -6]. Print the initial array.
- Replace all negative values in the array ini_array1 with 0. **Use conditional indexing to accomplish this.**
- Print the updated array ini_array1 after replacing negative values with 0
- Repeat ini_array1 with the following values: [1, 2, -3, 4, -5, -6]. Print the initial array.
- Replace all negative values in the array ini_array1 with 0. **Use np.where to accomplish this.**

```
In [74]: # Question 14 - Type your answers here
```

```
initial array [ 1  2 -3  4 -5 -6]
New resulting array:  [1 2 0 4 0 0]
```

```
In [75]: # Question 14 - Type your answers here
```

```
initial array [ 1  2 -3  4 -5 -6]
New resulting array:  [1 2 0 4 0 0]
```

Case Studies

Question 15

The dataset essentially captures information about different athletes, their ages, the sports they participate in, their physical attributes (weight and height), and a performance metric (score). Below are the columns description:

- athlete: The column represents the name of the athlete participating in sports.

- age: Indicates the age of the athlete.
- sport: Describes the type of sport in which the athlete is involved (e.g., Basketball, Soccer, Swimming).
- weight: Represents the weight of the athlete.
- height: Indicates the height of the athlete.
- score: This column provides a score or performance metric for the athlete in their respective sport.

In [76]:
```
# Q15 - Type your answer here

# Read the CSV file into a NumPy array
athlete_data = np.genfromtxt('sports_dataset.csv', delimiter=',', dtype=None, names=True, enc

# Print the array to verify the data
print(athlete_data)
```

```
[('John', 25, 'Basketball', 80, 190, 30)
 ('Alice', 22, 'Soccer', 65, 170, 25) ('Bob', 28, 'Swimming', 75, 180, 35)
 ('Eva', 24, 'Tennis', 68, 175, 28) ('Chris', 30, 'Running', 70, 175, 40)
 ('Sara', 23, 'Basketball', 78, 185, 32)
 ('Mike', 26, 'Soccer', 70, 175, 28) ('Lily', 27, 'Swimming', 73, 178, 38)
 ('Alex', 29, 'Tennis', 72, 180, 33) ('Emma', 24, 'Running', 68, 172, 36)]
```

In [77]:
```
for row in athlete_data:
    print(row)
```

```
('John', 25, 'Basketball', 80, 190, 30)
('Alice', 22, 'Soccer', 65, 170, 25)
('Bob', 28, 'Swimming', 75, 180, 35)
('Eva', 24, 'Tennis', 68, 175, 28)
('Chris', 30, 'Running', 70, 175, 40)
('Sara', 23, 'Basketball', 78, 185, 32)
('Mike', 26, 'Soccer', 70, 175, 28)
('Lily', 27, 'Swimming', 73, 178, 38)
('Alex', 29, 'Tennis', 72, 180, 33)
('Emma', 24, 'Running', 68, 172, 36)
```

- (a) Calculate the average age of the athletes.
- (b) Find the athlete with the highest score. Print their name and score.
- (c) Filter and display athletes who participate in Tennis.
- (d) Calculate the average height of athletes who play Basketball.
- (e) Find the athlete with the lowest weight. Print their name and weight.
- (f) Filter and display athletes who are younger than 25 years old.
- (g) Calculate the total score of all athletes.
- (h) Filter and display athletes with a score greater than 30.

In [78]:
```
# Question 15(a) - Calculate the average age of the athletes
# Type your answer here
```

```
The average age of the athletes is: 25.8
```

In [79]:
```
# Question 15(b) - Find the athlete with the highest score. Print their name and score.
# Type your answer here
```

```
The athlete with the highest score is Chris with a score of 40.
```

In [80]:
```
# Question 15(c) - Filter and display athletes who participate in Tennis.
# Type your answer here
```

```
Athletes who play Tennis:
[('Eva', 24, 'Tennis', 68, 175, 28) ('Alex', 29, 'Tennis', 72, 180, 33)]
```

```
In [81]: # Question 15(d) - Calculate the average height of athletes who play Basketball
         # Type your answer here
```

```
The average height of Basketball players is: 187.5
```

```
In [82]: # Question 15(e) - Find the athlete with the lowest weight. Print their name and weight.
         # Type your answer here
```

```
The athlete with the lowest weight is Alice with a weight of 65.
```

```
In [83]: # Question 15(f) - Filter and display athletes who are younger than 25 years old.
         # Type your answer here
```

```
Athletes younger than 25:
[('Alice', 22, 'Soccer', 65, 170, 25) ('Eva', 24, 'Tennis', 68, 175, 28)
 ('Sara', 23, 'Basketball', 78, 185, 32)
 ('Emma', 24, 'Running', 68, 172, 36)]
```

```
In [84]: # Question 15(g) - Calculate the total score of all athletes.
         # Type your answer here
```

```
The total score of all athletes is: 325
```

```
In [85]: # Question 15(h) - Filter and display athletes with a score greater than 30.
         # Type your answer here
```

```
Athletes with a score greater than 30:
[('Bob', 28, 'Swimming', 75, 180, 35)
 ('Chris', 30, 'Running', 70, 175, 40)
 ('Sara', 23, 'Basketball', 78, 185, 32)
 ('Lily', 27, 'Swimming', 73, 178, 38) ('Alex', 29, 'Tennis', 72, 180, 33)
 ('Emma', 24, 'Running', 68, 172, 36)]
```

Challenging Questions

Question 16

Given a system of linear equations:

2x + 3y = 8

5x - y = -2

a) Represent this system of equations as a matrix equation of the form Ax = b.

b) Solve the system of equations using NumPy's linalg.solve() function.

Question 16(a) - Type your answers here

```
In [6]: # Question 16(b) - Type your answers here
```

```
[0.11764706 2.58823529]
```

$2(0.118)+3(2.588)=8.000$

$5(0.118)-(2.588)=-2.000$

The solutions $x \approx 0.118$ and $y \approx 2.588$ satisfy both equations

Question 17

Finding the k smallest values of a NumPy array input = [23, 12, 1, 3, 4, 5, 6]

k = 4

output = [1, 3, 4, 5]

In [88]:
```
# Question 17 - Type your answers here
```

```
The Original Array Content:
 [23 12  1  3  4  5  6]

4 smallest elements of the array
[1 3 4 5]
```

Question 18

Find the most frequent value in a NumPy array

input = [1, 1, 1, 2, 3, 4, 2, 4, 3, 3]

output = [1, 3]

Hints for students:

- Apply bincount() method of NumPy to get the count of occurrences of each element in the array.
- Apply argmax() method to get the value having a maximum number of occurrences(frequency).

In [89]:
```
# Question 18 - Type your answers here
```

```
Original array:
[1 1 1 2 3 4 2 4 3 3]

Most frequent value in above array
1 3
```

Question 19

Write a Python program that performs the following steps:

- Imports the NumPy library.
- Initializes a 2D NumPy array named ini_array with the following values (with some NaN entries):
- [[1.3, 2.5, 3.6, NaN],

[2.6, 3.3, NaN, 5.5], [2.1, 3.2, 5.4, 6.5]- Prints the initial array.

- Calculates the mean of each column, ignoring any NaN values, and stores the result in col_mean.
- Replaces each NaN value with the mean of the respective column.
- Prints the final array with NaN values replaced by the column averages.]

In [90]:
```
# Question 19 - Type your answers here
```

```
initial array:
 [[1.3 2.5 3.6 nan]
 [2.6 3.3 nan 5.5]
 [2.1 3.2 5.4 6.5]]

columns mean:
 [2.  3.  4.5 6. ]

final array:
 [[1.3 2.5 3.6 6. ]
 [2.6 3.3 4.5 5.5]
 [2.1 3.2 5.4 6.5]]
```

Question 20

Using back the same array in Question 19. Let replace the NaN with the average row contents

In [91]:
```
# Question 20 - Type your answers here
y
```

```
initial array:
 [[1.3 2.5 3.6 nan]
 [2.6 3.3 nan 5.5]
 [2.1 3.2 5.4 6.5]]

rows mean:
 [2.46666667 3.8        4.3       ]

final array:
 [[1.3        2.5        3.6        2.46666667]
 [2.6        3.3        3.8        5.5       ]
 [2.1        3.2        5.4        6.5       ]]
```

Additional Informations on Sorting

Practical 3 Question 13(f) np.argsort() and Practical 4 Question 17 np.sort().

In NumPy, np.sort() and np.argsort() are both used for sorting, but they serve different purposes:

np.sort():

- np.sort(array) sorts the elements of an array in ascending order (by default) and returns a sorted copy of the array.
- It changes the order of values in the array, but does not provide information about the original indices.

  ```
  import numpy as np

  arr = np.array([3, 1, 2])
  sorted_arr = np.sort(arr)
  print(sorted_arr)

  [1,2,3]
  ```

np.argsort():

- np.argsort(array) returns the indices that would sort an array.
- This means it doesn't change the original order of the values but gives the indices in a sequence that would produce a sorted array.
- Useful when you want to know the sorted order without actually sorting the array.

```
import numpy as np

arr = np.array([3, 1, 2])
sorted_indices = np.argsort(arr)
print(sorted_indices)
print(arr[sorted_indices])

[1,2,0]
[1,2,3]
```

```
In [95]: arr = np.array([3, 1, 2])
         sorted_arr = np.sort(arr)
         print(sorted_arr)
```

```
[1 2 3]
```

```
In [94]: arr = np.array([3, 1, 2])
         sorted_indices = np.argsort(arr)
         print(sorted_indices)      # Output: [1 2 0]
         print(arr[sorted_indices]) # arr[sorted_indices] will give the sorted array [1 2 3]
```

```
[1 2 0]
[1 2 3]
```