Introduction to `NumPy`

NumPy, short for Numerical Python, is a powerful library for numerical computing in Python. It provides support for arrays, matrices, and many mathematical functions.

Step 1: Installation and Importing

In [ ]:
```
%pip install numpy
```

```
Requirement already satisfied: numpy in e:\codes\.venv\lib\site-packages (2.0.1)
Note: you may need to restart the kernel to use updated packages.
```

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

In [ ]:
```
# [3,4,5,6] #1D array

# [[3,4,5],[6,8,9]] #2D Array

# [[],[],[]] #3D ARRAY

np.__version__
```

Out[ ]:
```
'2.0.1'
```

Step 2: Creating Arrays

Scenario: Imagine you need to create arrays to store data for analysis, such as a list of temperatures over a week.

In [ ]:
```
#1D Array

temp_list=[23, 25, 19, 22, 30, 28, 24]
temperatures = np.array(temp_list)
print(temperatures)


type(temperatures)
```

Out[ ]:
```
numpy.ndarray
```

In [ ]:
```
#2D Array

matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(matrix)

#np.shape(matrix)
```

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

In [ ]:
```
#Zeros and Ones Arrays:

# Linear Algebra
# Calculus
# Probablity

import math

zeros_array = np.zeros((3, 3)) #(n,n)
ones_array = np.ones((2, 4))
print(zeros_array)
print(ones_array)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

Step 3: Array Operations

Scenario: You have two arrays representing the sales of two products over a week. You want to perform basic arithmetic operations to analyze the sales data.

In [ ]:
```
product1_sales = np.array([10, 20, 30, 40, 50, 60, 70]) #7
product2_sales = np.array([5, 15, 25, 35, 45, 55, 60]) #6

# Addition
total_sales = product1_sales + product2_sales
print(total_sales)

# Subtraction
sales_diff = product1_sales - product2_sales
print(sales_diff)

# Multiplication
sales_mult = product1_sales * product2_sales
print(sales_mult)

# Division
sales_div = product1_sales / product2_sales
print(sales_div)
```

```
[ 15  35  55  75  95 115 130]
[ 5  5  5  5  5  5 10]
[  50  300  750 1400 2250 3300 4200]
[2.         1.33333333 1.2        1.14285714 1.11111111 1.09090909
 1.16666667]
```

Step 4: Statistical Operations

Scenario: You have collected students' scores in a test and want to calculate statistical data like mean, median, and standard deviation

In [ ]:
```
scores = np.array([85, 90, 78, 92, 88, 76, 95, 89])

# Mean
mean_score = np.mean(scores)
print("Mean Score:", mean_score)

# Median
median_score = np.median(scores)
print("Median Score:", median_score)

# Standard Deviation
std_deviation = np.std(scores)
print("Standard Deviation:", std_deviation)
```

```
Mean Score: 86.625
Median Score: 88.5
Standard Deviation: 6.203577596838779
```

In [ ]:
```
scores =  np.array([[85, 90, 78, 92, 88, 76, 95, 89],[85, 90, 78, 92, 88, 76, 95, 8

#2 rows, 8 columns
scores.shape #1st parameter tells you about the dimensionality and 2nd Parameter te

#student_id, course_id
#Rows tell you about the number of data entry
#Columns tell you about parameters of the data.

[
 [85, 90, 78, 92, 88, 76, 95, 89],
 [85, 90, 78, 92, 88, 76, 95, 89]
                                    ]



# Create Matrices using Arange and Reshape

#np.arange(10).reshape(10,1)

np.arange(30).reshape(5,6) #reshape-->n1*n2==>
```

Out[ ]:
```
array([[ 0,  1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10, 11],
       [12, 13, 14, 15, 16, 17],
       [18, 19, 20, 21, 22, 23],
       [24, 25, 26, 27, 28, 29]])
```

Step 5: Indexing and Slicing

Scenario: You want to extract specific data from an array, such as temperatures on the first three days of the week.

In [ ]:
```
_list=[85, 90, 78, 92, 88, 76, 95, 89]

_list[2:6] #n1:n2-1
```

Out[ ]:
```
[78, 92, 88, 76]
```

In [ ]:
```
# temperatures = np.array([23, 25, 19, 22, 30, 28, 24])

# # Indexing
# first_day_temp = temperatures[0]
# print("Temperature on first day:", first_day_temp)

# # Slicing
# first_three_days_temp = temperatures[:3]
# print("Temperatures on first three days:", first_three_days_temp)

#5
#Index begins at 0...4
nArray=np.arange(30).reshape(5,6)
print(nArray)


#1st part: Slice rows
#2nd part: Slice elemnets inside the list
nArray[2:4,3:]
```

```
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]
 [12 13 14 15 16 17]
 [18 19 20 21 22 23]
 [24 25 26 27 28 29]]
```

Out[ ]:
```
array([[15, 16, 17],
       [21, 22, 23]])
```

Step 6: Reshaping Arrays

Scenario: You have a linear array of 12 elements and want to reshape it into a 3x4 matrix for easier analysis.

In [ ]:
```
linear_array = np.arange(17) #1,17
matrix_3x4 = linear_array.reshape((1,17))
print(matrix_3x4)

#np.diagnal()
```

```
[[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16]]
```

Step 7: Concatenation and Splitting

Scenario: You need to combine sales data from two weeks or split it into daily and weekly data.

In [ ]:
```
#Concatenation

week1_sales = np.array([10, 20, 30, 40, 50, 60, 70])
week2_sales = np.array([15, 25, 35, 45, 55, 65, 75])
```

```
total_sales = np.concatenate((week1_sales, week2_sales))
print(total_sales)
```

```
[10 20 30 40 50 60 70 15 25 35 45 55 65 75]
```

In [ ]:
```
#splitting

daily_sales = np.array([10, 20, 30, 40, 50, 60, 70, 15, 25, 35, 45, 55, 65, 75])
week1_sales, week2_sales = np.split(daily_sales, 2)
print("Week 1 Sales:", week1_sales)
print("Week 2 Sales:", week2_sales)
```

Step 8: Broadcasting

Scenario: You want to increase all temperatures by 2 degrees for a week.

In [ ]:
```
temperatures = np.array([23, 25, 19, 22, 30, 28, 24])
temperatures += 2
print("Updated Temperatures:", temperatures)
```

```
Updated Temperatures: [25 27 21 24 32 30 26]
```

Step 9: Advanced Operations

Scenario: You have data for two matrices representing different datasets and want to perform matrix multiplication.

In [ ]:
```
matrix1 = np.array([[1, 2], [3, 4]])
matrix2 = np.array([[5, 6], [7, 8]])

result = np.dot(matrix1, matrix2)
print(result)
```

```
[[19 22]
 [43 50]]
```

Important NumPy Methods

`np.arange(start, stop, step)` - Returns evenly spaced values within a given interval.

`np.linspace(start, stop, num)` - Returns evenly spaced numbers over a specified interval.

`np.random.rand(d0, d1, ..., dn)` - Generates random numbers.

`np.sum(array)` - Sum of array elements.

`np.max(array)` - Maximum value of array elements.

`np.min(array)` - Minimum value of array elements.

`np.argmax(array)` - Indices of the maximum values.

`np.argmin(array)` - Indices of the minimum values.

`np.unique(array)` - Find the unique elements of an array.

`np.transpose(array)` - Permute the dimensions of an array.

Conclusion

This tutorial provides a comprehensive overview of NumPy's capabilities. Practice these examples and scenarios to get a solid understanding of how NumPy can be used in various data analysis tasks. Encourage students to experiment with the code and come up with their own scenarios for a deeper understanding.

Calculate The Correlation of a Matrix.