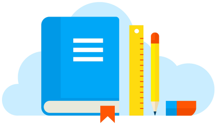
Python For Machine Learning
NumPy Cheat Sheet

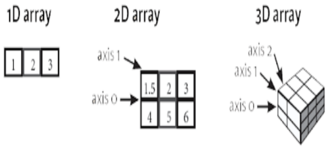


Class Notes

The NumPy library is the core library for scientific computing Python. It provides a high performance multidimensional array object , and tools for working with these arrays

Use the following import convention :

import numpy as np



1. Creating Arrays

1D array:

```
# A simple vector
arr = np.array([1, 2, 3])
```

2D array (Matrix):

```
# 2x3 matrix(3 rows, 2 column)
matrix = np.array([[1, 2], [3, 4], [5, 6]])
```

Zeros & Ones:

```
# 2x3 matrix of zeros
zeros = np.zeros((2, 3))
# 3x2 matrix of ones
ones = np.ones((3, 2))
```

Identity matrix:

```
# 3x3 identity matrix
identity = np.eye(3)
```

Range of values:

```
# Array: [0, 2, 4, 6, 8]
range_array = np.arange(0, 10, 2)
```

Linspace:

```
# Array: [0, 0.25, 0.5, 0.75, 1]
linspace_array = np.linspace(0, 1, 5)
```

Random numbers:

```
# 2x3 matrix with values between 0 and 1
random_vals = np.random.rand(2, 3)
# 2x3 matrix from a standard normal distribution
randn_vals = np.random.randn(2, 3)
```

2. Array Operations

Reshape:

```
# Changes shape, keeps data
reshaped = matrix.reshape(2, 3)
```

Transpose:

```
# Switches rows with columns
transpose = matrix.T
```

Element-wise operations:

```
# Adds elements at the same positions
sum_arrays = arr + arr
# Multiplies elements at the same positions
product_arrays = arr * arr
```

Dot product:

```
# For vectors, it's a scalar product. For matrices, it's matrix multiplication.
dot_product = np.dot(arr, arr)
```

Aggregation:

```
# Sum of elements
array_sum = np.sum(arr)
# Average of elements
array_mean = np.mean(arr)
# Standard deviation
array_std = np.std(arr)
```

3. Indexing & Slicing

```
# Access element at 2nd row, 2nd column
element = matrix[1, 1]
# Get entire 2nd row
row = matrix[1, :]
# Get entire 2nd column
column = matrix[:, 1]
```

4. Boolean Indexing

```
# Checks if elements are greater than 2
condition = matrix > 2
# Returns elements that meet the condition
filtered_matrix = matrix[condition]
```

5. Broadcasting

```
# Adds [1, 2, 3] to every row of arr
result = arr + np.array([1, 2, 3])
```

6. Linear Algebra

Inverse:

```
# Finds the inverse of a square matrix
inverse = np.linalg.inv(matrix)
```

Matrix multiplication:

```
# Multiplies two matrices
result = matrix1 @ matrix2
```

7. Stacking & Splitting

Vertical & Horizontal stacking:

```
# Combines matrices vertically
vstacked = np.vstack((matrix1, matrix2))
# Combines matrices horizontally
hstacked = np.hstack((matrix1, matrix2))
```

Splitting:

```
# Divides an array into 3 parts
split_array = np.array_split(arr, 3)
```