Chapter 4 - NumPy Basics: Arrays and Vectorized Computation

W. Mckinney, “Numpy Basics: Arrays and Vectorized Computation,” in *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyther*, 3rd ed., O’Reilly, 2022, pp. 83-121.

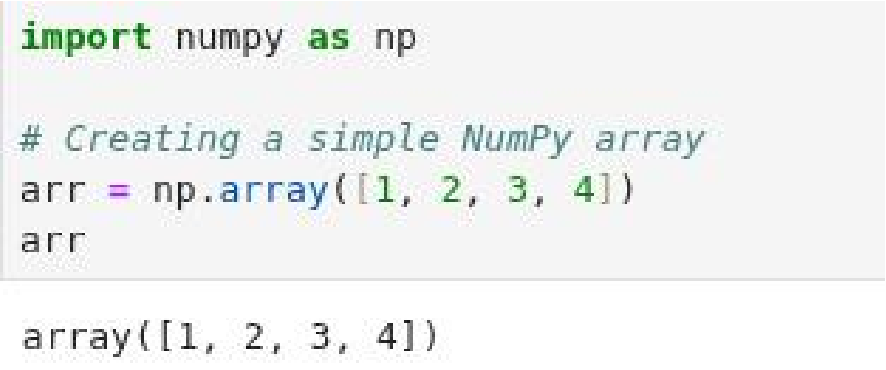
https://datamineaz.org/

Numpy

- NumPy, short for Numerical Python, is one of the most important foundational packages for numerical computing in Python.
  - ndarray, an efficient multidimensional array providing fast array-oriented arithmetic operations and flexible broadcasting capabilities
  - Mathematical functions for fast operations on entire arrays of data without having to write loops
  - Tools for reading/writing array data to disk and working with memory-mapped files
  - Linear algebra, random number generation, and Fourier transform capabilities
  - A C API for connecting NumPy with libraries written in C, C++, or FORTRAN

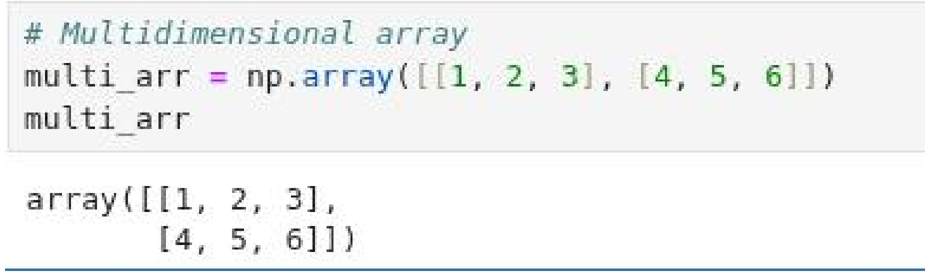
Numpy Arrays

- Creating arrays



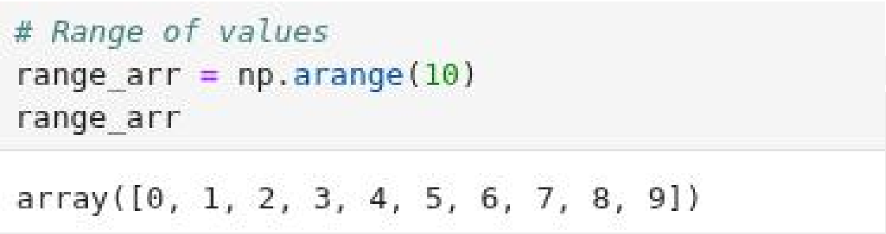
```python
import numpy as np

# Creating a simple NumPy array
arr = np.array([1, 2, 3, 4])
arr
```

```
array([1, 2, 3, 4])
```



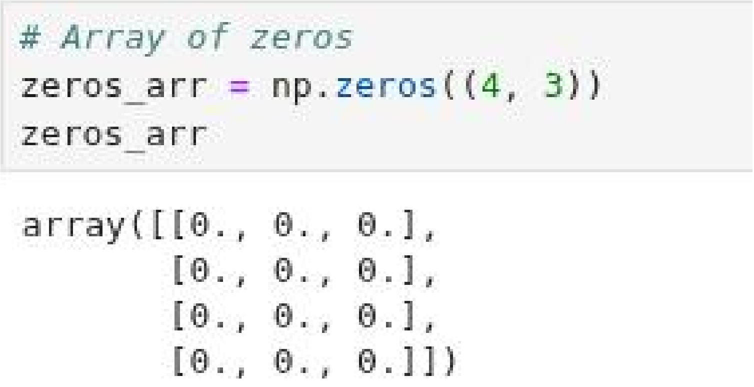
```python
# Multidimensional array
multi_arr = np.array([[1, 2, 3], [4, 5, 6]])
multi_arr
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```



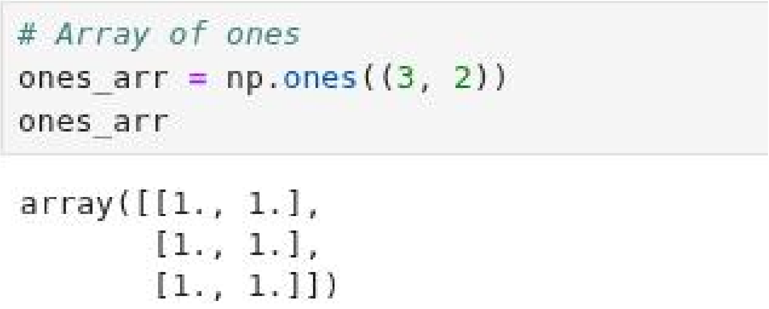
```python
# Range of values
range_arr = np.arange(10)
range_arr
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```



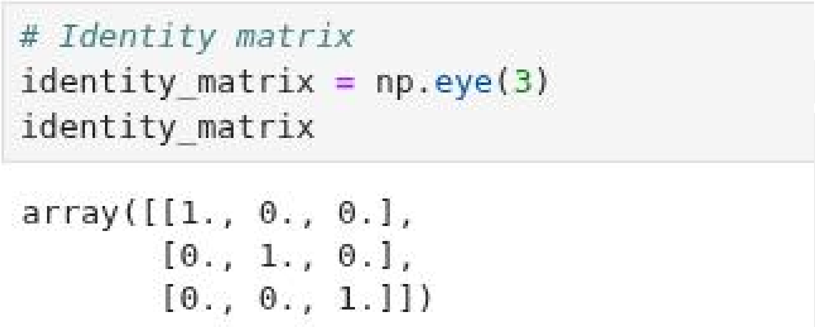
```python
# Array of zeros
zeros_arr = np.zeros((4, 3))
zeros_arr
```

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```



```python
# Array of ones
ones_arr = np.ones((3, 2))
ones_arr
```

```
array([[1., 1.],
       [1., 1.],
       [1., 1.]])
```

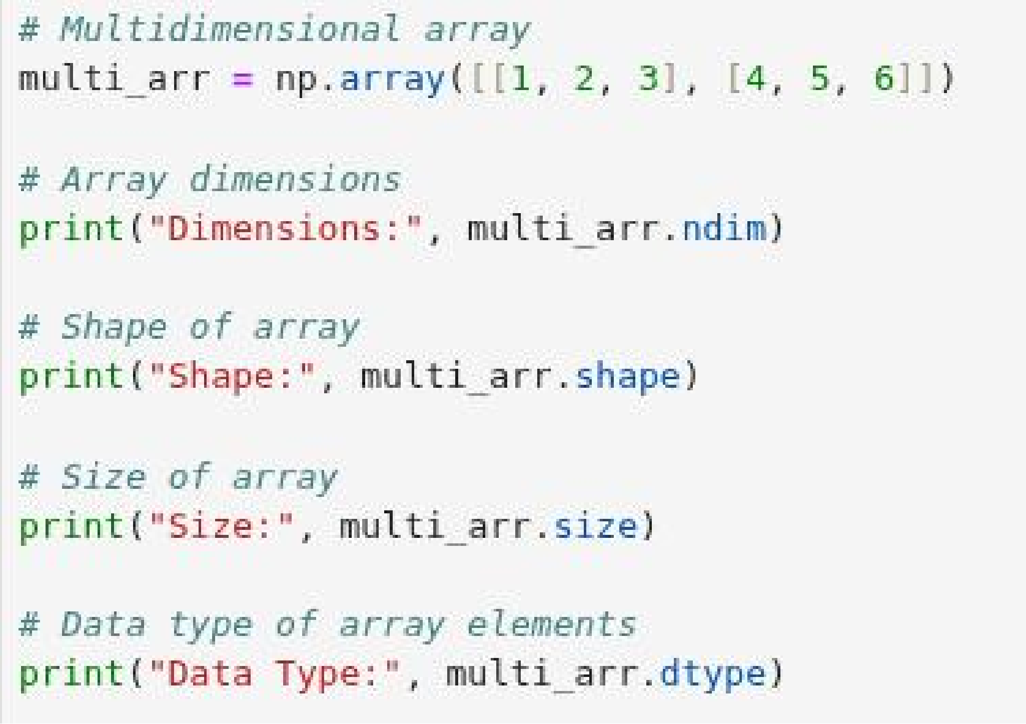


```python
# Identity matrix
identity_matrix = np.eye(3)
identity_matrix
```

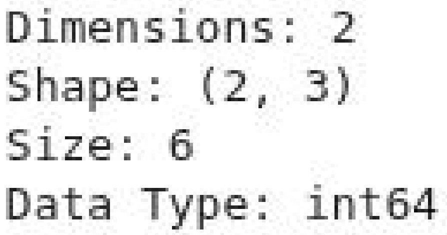
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Numpy Arrays

- Array attributes



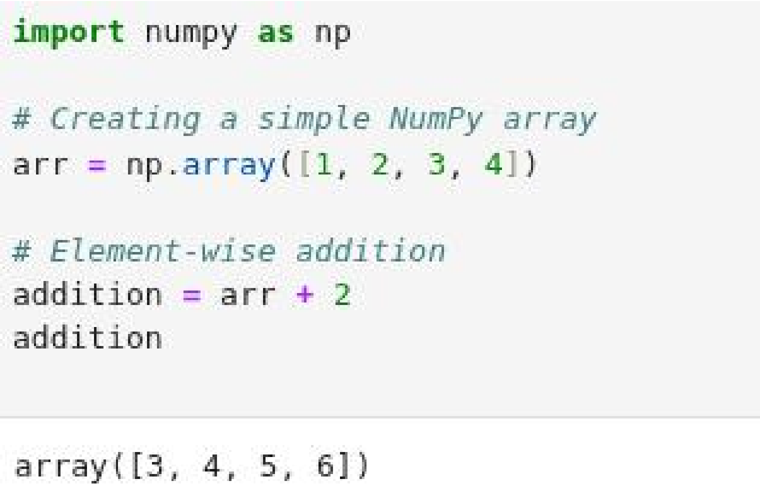
```
# Multidimensional array
multi_arr = np.array([[1, 2, 3], [4, 5, 6]])

# Array dimensions
print("Dimensions:", multi_arr.ndim)

# Shape of array
print("Shape:", multi_arr.shape)

# Size of array
print("Size:", multi_arr.size)

# Data type of array elements
print("Data Type:", multi_arr.dtype)
```



```
Dimensions: 2
Shape: (2, 3)
Size: 6
Data Type: int64
```

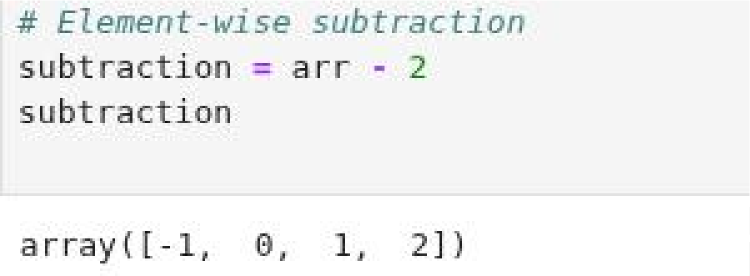
Numpy Operations

- Arithmetic operations



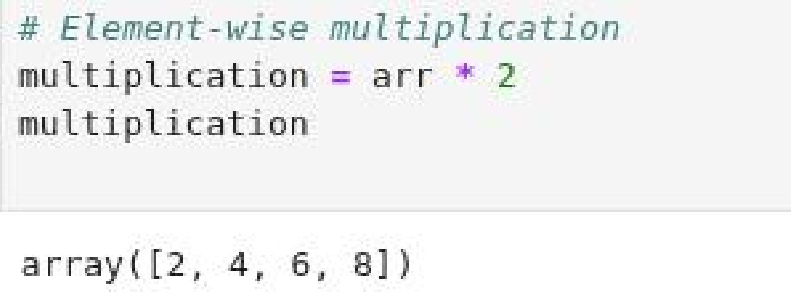
```
import numpy as np

# Creating a simple NumPy array
arr = np.array([1, 2, 3, 4])

# Element-wise addition
addition = arr + 2
addition
```

```
array([3, 4, 5, 6])
```



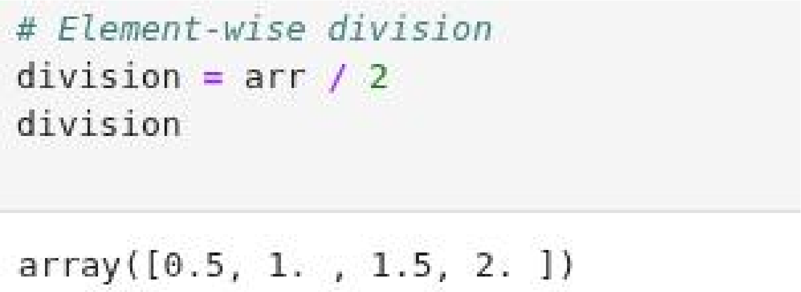
```
# Element-wise subtraction
subtraction = arr - 2
subtraction
```

```
array([-1,  0,  1,  2])
```



```
# Element-wise multiplication
multiplication = arr * 2
multiplication
```

```
array([2, 4, 6, 8])
```

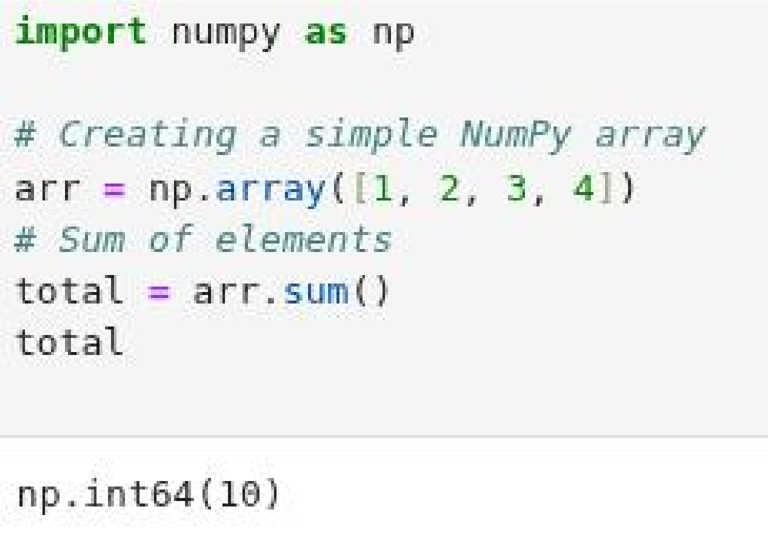


```
# Element-wise division
division = arr / 2
division
```

```
array([0.5, 1. , 1.5, 2. ])
```

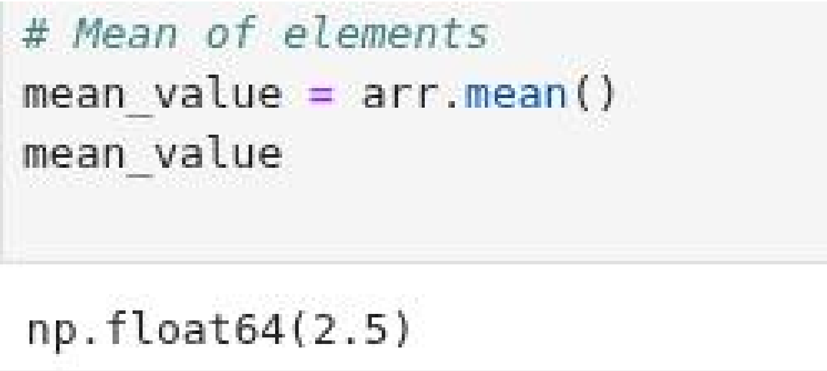
Numpy Operations

- Statistical operations



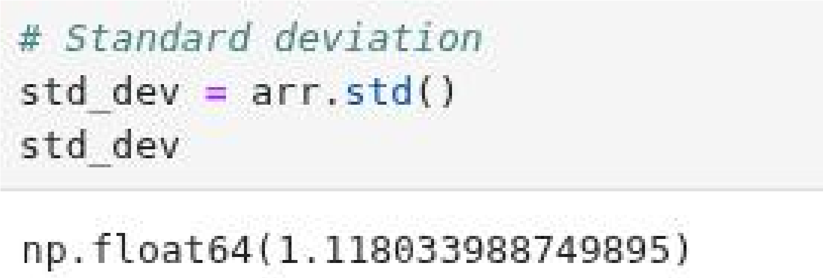
```
import numpy as np

# Creating a simple NumPy array
arr = np.array([1, 2, 3, 4])
# Sum of elements
total = arr.sum()
total
```

```
np.int64(10)
```



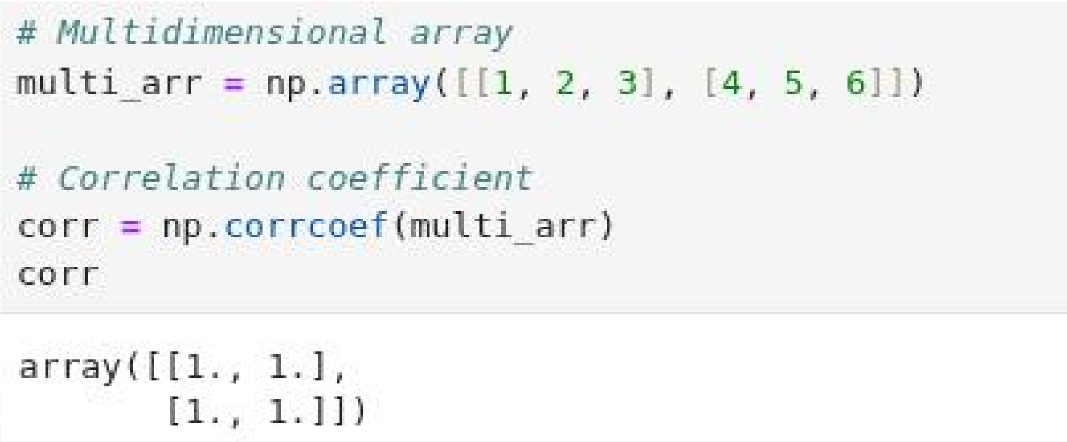
```
# Mean of elements
mean_value = arr.mean()
mean_value
```

```
np.float64(2.5)
```



```
# Standard deviation
std_dev = arr.std()
std_dev
```

```
np.float64(1.118033988749895)
```

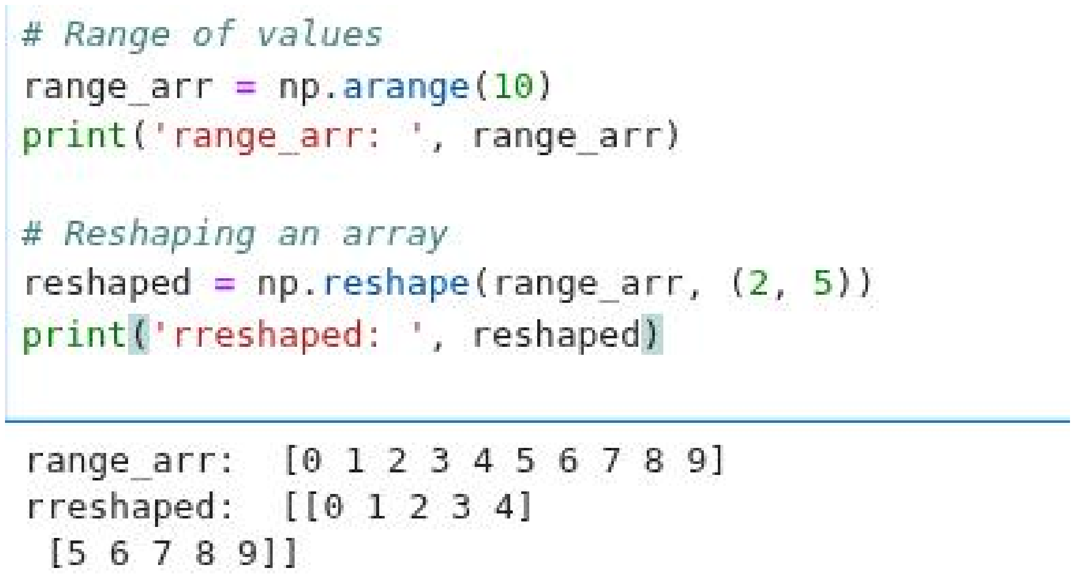


```
# Multidimensional array
multi_arr = np.array([[1, 2, 3], [4, 5, 6]])

# Correlation coefficient
corr = np.corrcoef(multi_arr)
corr
```

```
array([[1., 1.],
       [1., 1.]])
```

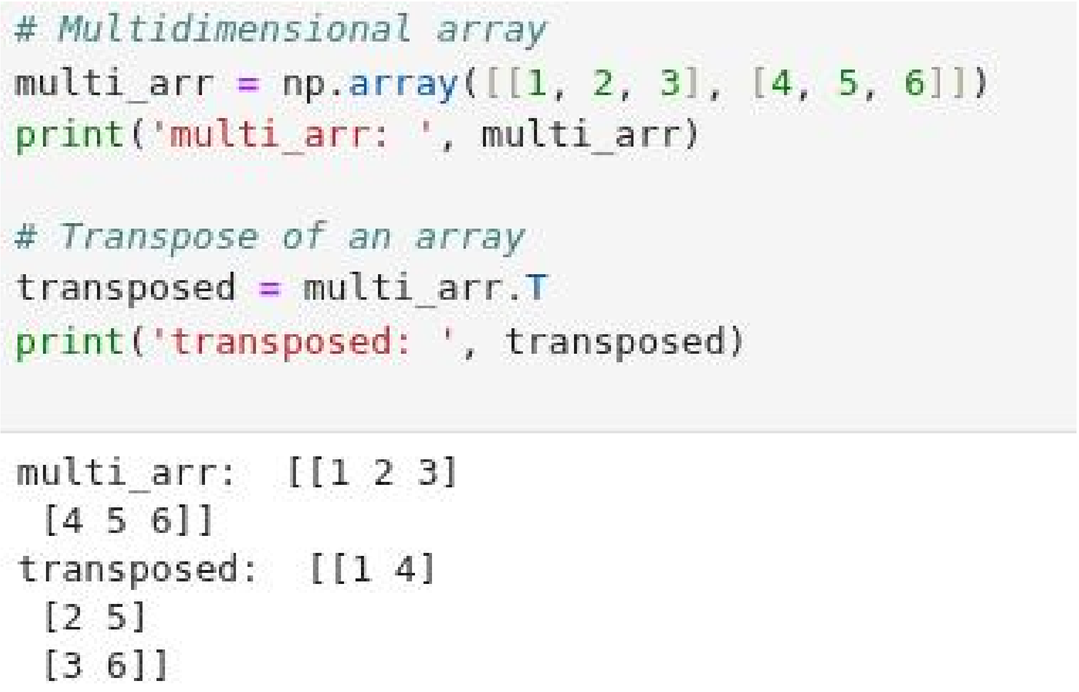
Advanced Operations

- Reshaping and transposing



```
# Range of values
range_arr = np.arange(10)
print('range_arr: ', range_arr)

# Reshaping an array
reshaped = np.reshape(range_arr, (2, 5))
print('rreshaped: ', reshaped)
```

```
range_arr:  [0 1 2 3 4 5 6 7 8 9]
rreshaped:  [[0 1 2 3 4]
 [5 6 7 8 9]]
```

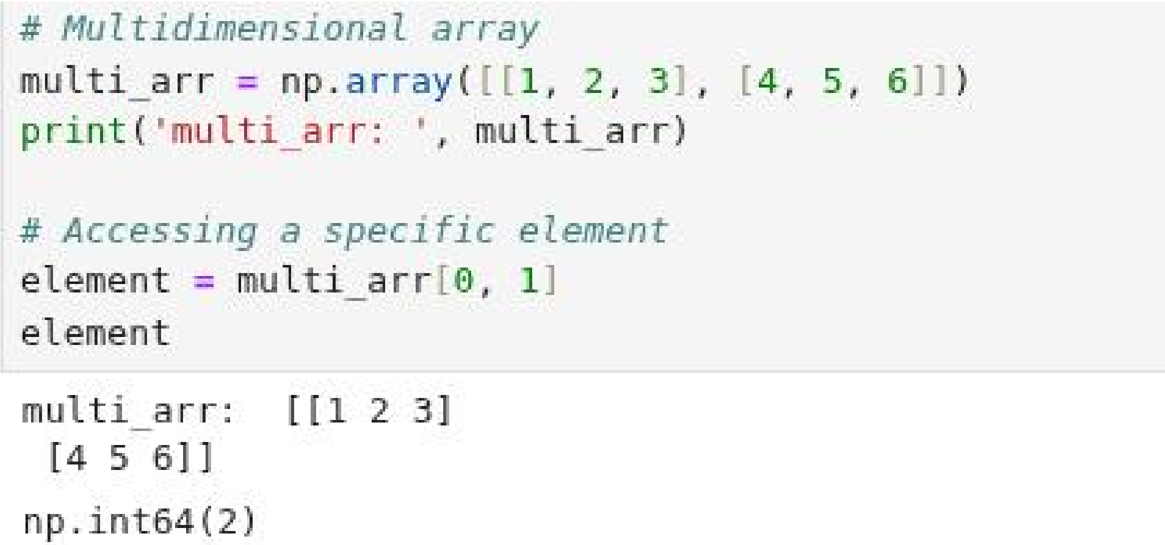


```
# Multidimensional array
multi_arr = np.array([[1, 2, 3], [4, 5, 6]])
print('multi_arr: ', multi_arr)

# Transpose of an array
transposed = multi_arr.T
print('transposed: ', transposed)
```

```
multi_arr:  [[1 2 3]
 [4 5 6]]
transposed:  [[1 4]
 [2 5]
 [3 6]]
```

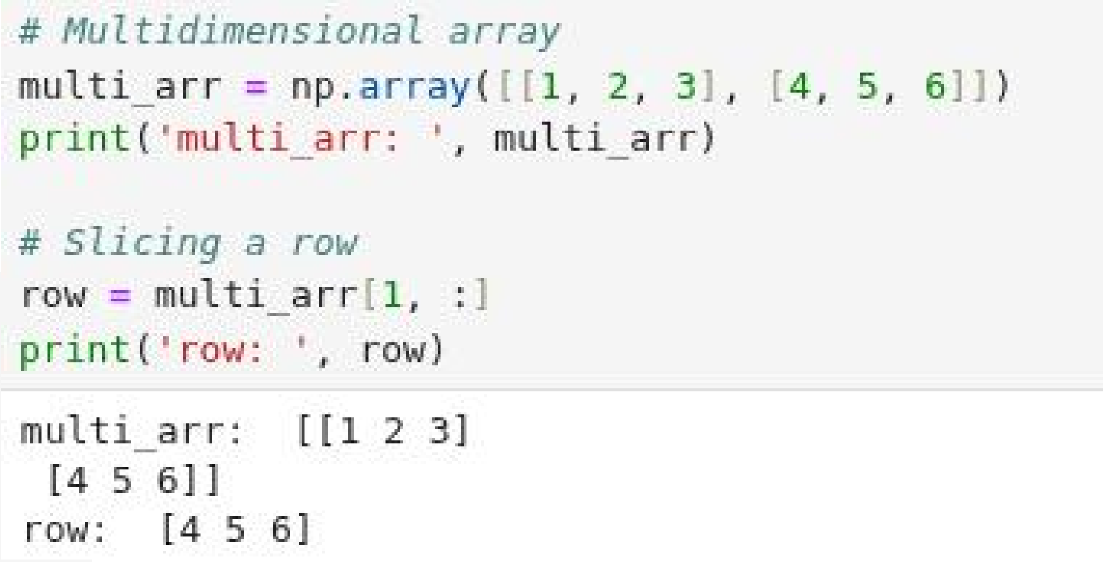
Advanced Operations

- Indexing and slicing



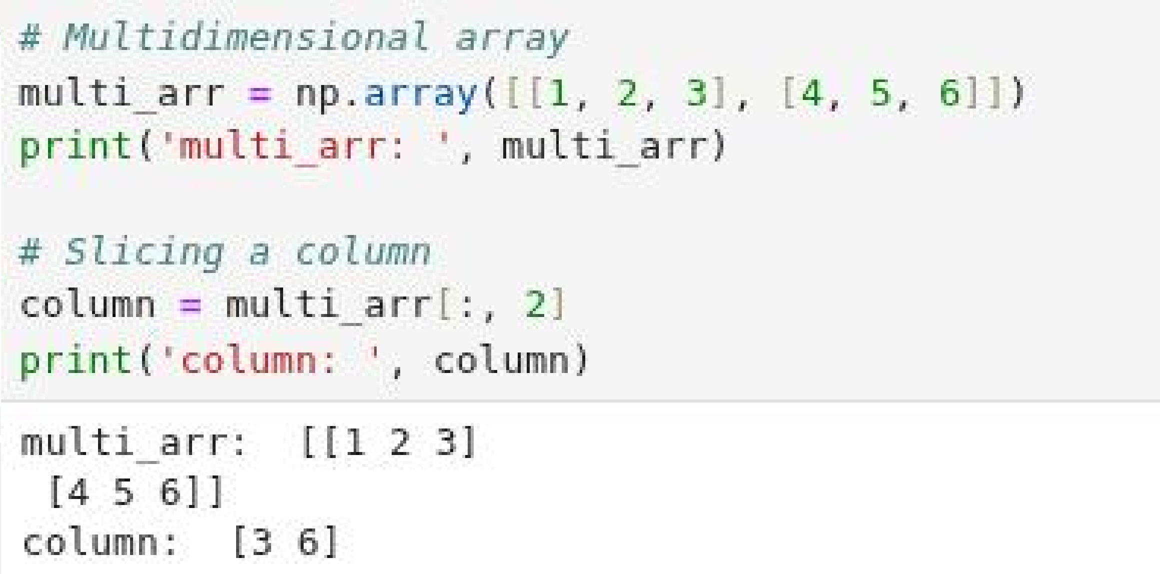
```
# Multidimensional array
multi_arr = np.array([[1, 2, 3], [4, 5, 6]])
print('multi_arr: ', multi_arr)

# Accessing a specific element
element = multi_arr[0, 1]
element
```

```
multi_arr:  [[1 2 3]
 [4 5 6]]
np.int64(2)
```



```
# Multidimensional array
multi_arr = np.array([[1, 2, 3], [4, 5, 6]])
print('multi_arr: ', multi_arr)

# Slicing a row
row = multi_arr[1, :]
print('row: ', row)
```

```
multi_arr:  [[1 2 3]
 [4 5 6]]
row:  [4 5 6]
```

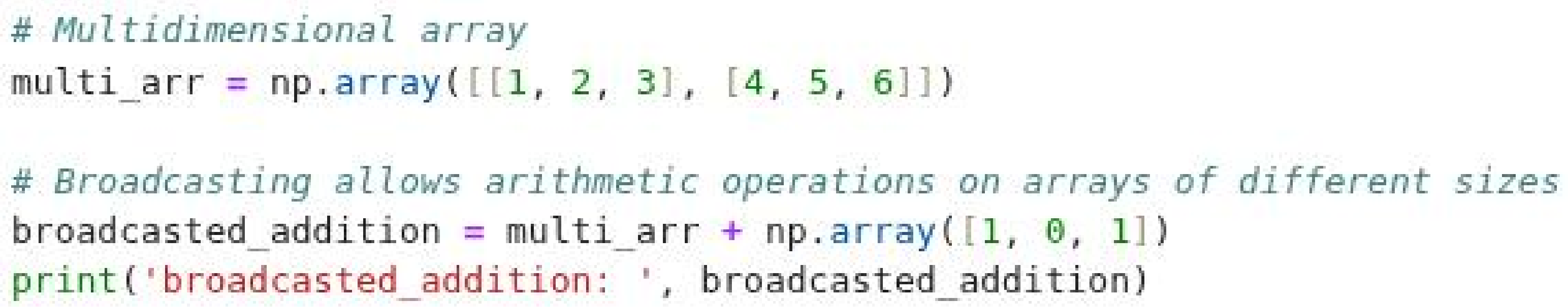


```
# Multidimensional array
multi_arr = np.array([[1, 2, 3], [4, 5, 6]])
print('multi_arr: ', multi_arr)

# Slicing a column
column = multi_arr[:, 2]
print('column: ', column)
```

```
multi_arr:  [[1 2 3]
 [4 5 6]]
column:  [3 6]
```

Advanced Operations

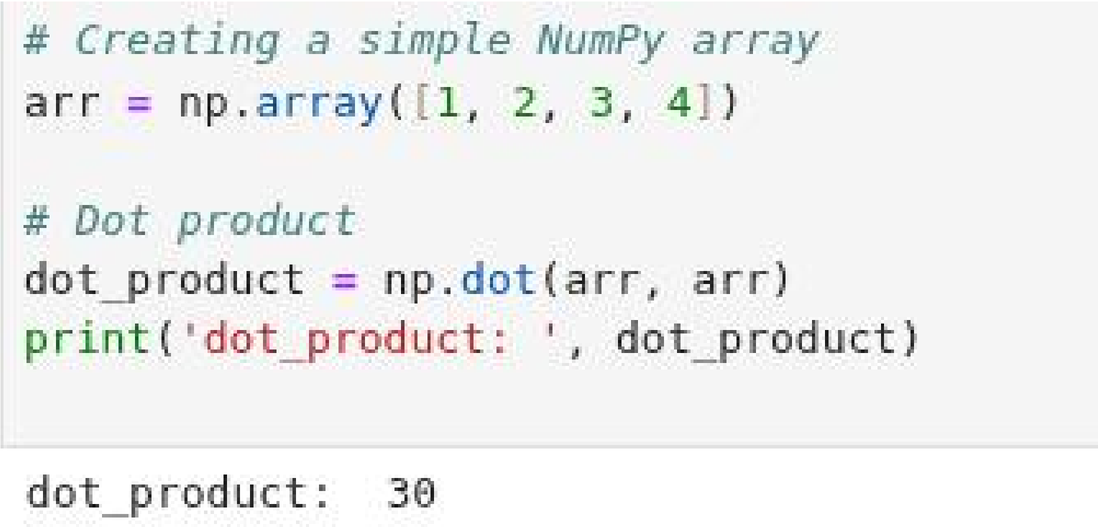
- Broadcasting



```
# Multidimensional array
multi_arr = np.array([[1, 2, 3], [4, 5, 6]])

# Broadcasting allows arithmetic operations on arrays of different sizes
broadcasted_addition = multi_arr + np.array([1, 0, 1])
print('broadcasted_addition: ', broadcasted_addition)
```

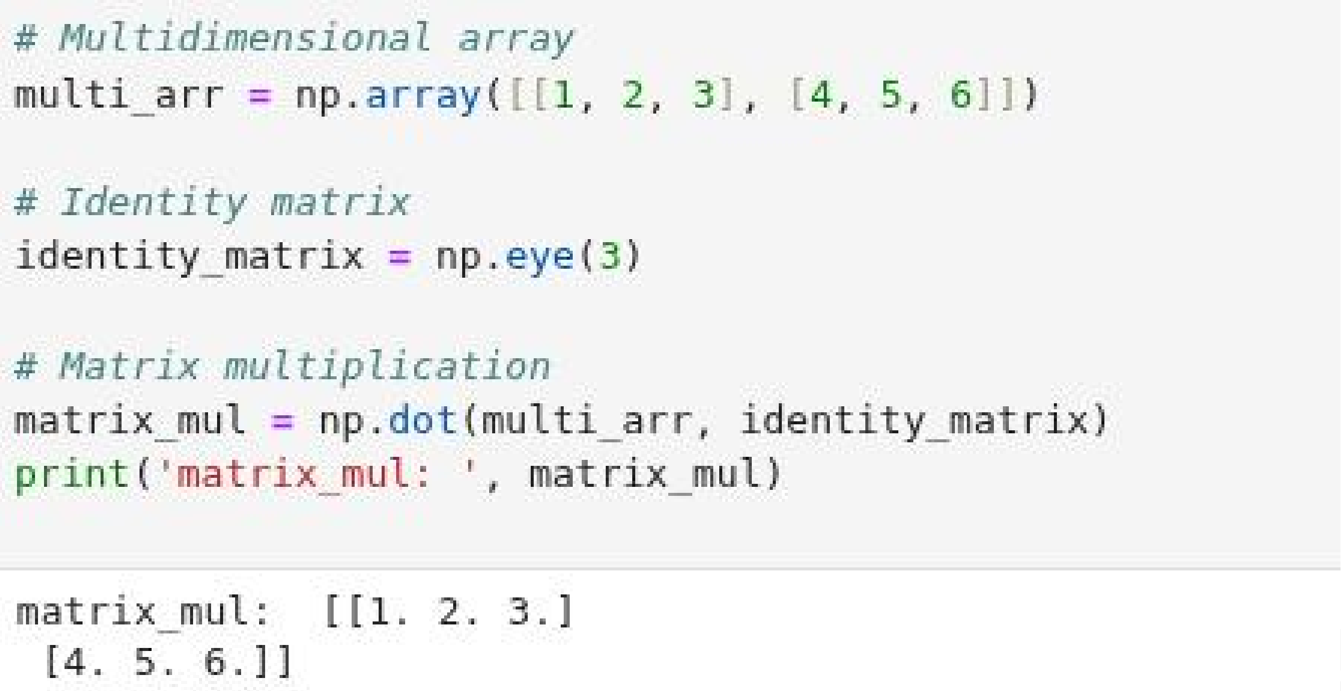
Linear Algebra

- Matrix operations



```
# Creating a simple NumPy array
arr = np.array([1, 2, 3, 4])

# Dot product
dot_product = np.dot(arr, arr)
print('dot_product: ', dot_product)
```

```
dot_product:  30
```

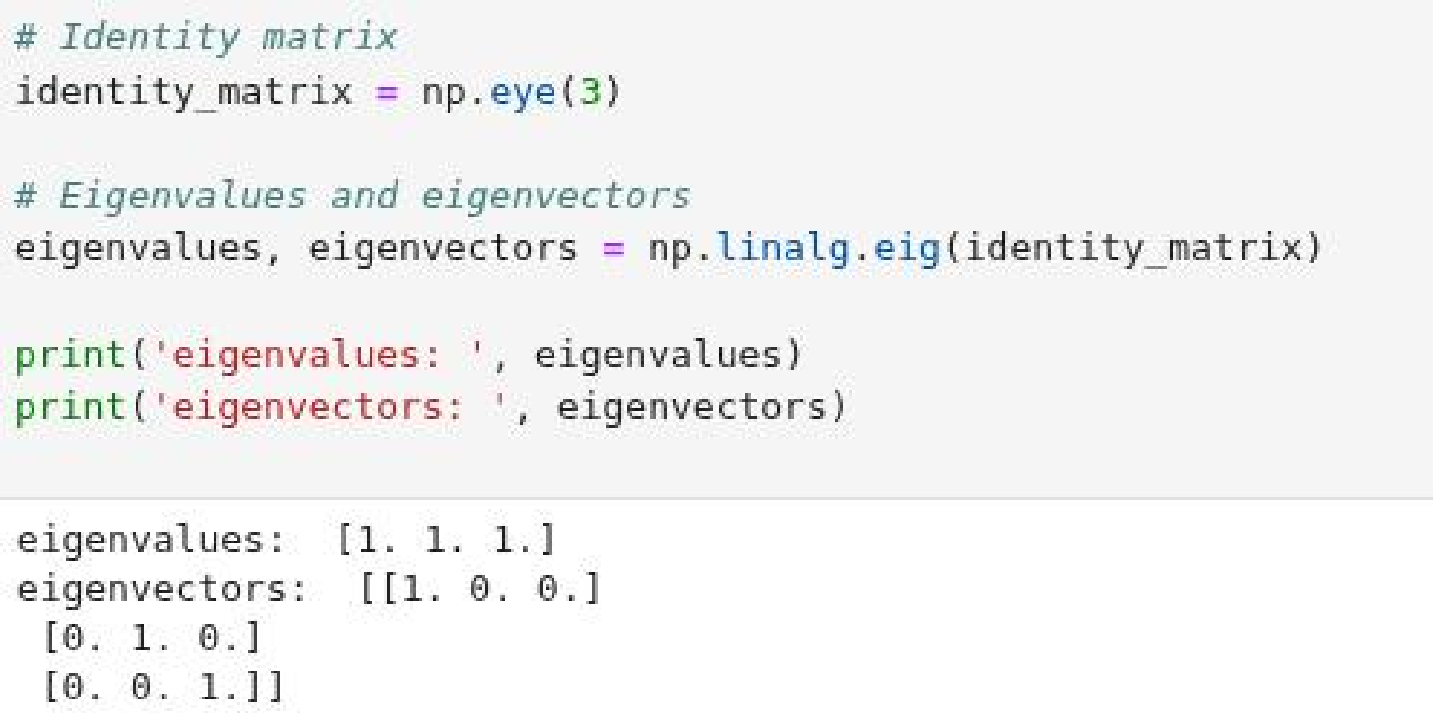


```
# Multidimensional array
multi_arr = np.array([[1, 2, 3], [4, 5, 6]])

# Identity matrix
identity_matrix = np.eye(3)

# Matrix multiplication
matrix_mul = np.dot(multi_arr, identity_matrix)
print('matrix_mul: ', matrix_mul)
```

```
matrix_mul:  [[1. 2. 3.]
 [4. 5. 6.]]
```

Linear Algebra

- Eigenvalues and Eigenvectors



```
# Identity matrix
identity_matrix = np.eye(3)

# Eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(identity_matrix)

print('eigenvalues: ', eigenvalues)
print('eigenvectors: ', eigenvectors)
```

```
eigenvalues:  [1. 1. 1.]
eigenvectors:  [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```