# NumPy Beginner Cheat Sheet

## 1. Importing NumPy

The standard convention for importing NumPy.

```
import numpy as np
```

## 2. Creating Arrays

Create NumPy arrays (`ndarray`) from lists or using built-in functions.

### From a Python List

Use `np.array()` to convert a list or nested list.

```
# 1D array
a = np.array([1, 2, 3])

# 2D array (matrix)
b = np.array([[1.5, 2, 3], [4, 5, 6]])
```

### Using Built-in Functions

Create arrays with initial placeholders.

```
# Create an array of zeros
c = np.zeros((2, 3)) # a 2x3 matrix of zeros

# Create an array of ones
d = np.ones((3, 2)) # a 3x2 matrix of ones

# Create an array with a range of elements
e = np.arange(10, 25, 5) # array([10, 15, 20])

# Create an array with evenly spaced values
f = np.linspace(0, 2, 9) # 9 numbers from 0 to 2

# Create a 2x2 identity matrix
g = np.eye(2)

# Create an array with random values
h = np.random.rand(2, 2) # 2x2, uniform dist [0,1)
i = np.random.randn(2, 2) # 2x2, standard normal
    dist
```

## 3. Array Attributes

Inspect the size, shape, and data type of an array.

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

# Shape of array (rows, columns)
arr.shape  # Returns: (2, 3)

# Number of dimensions
arr.ndim  # Returns: 2

# Total number of elements
arr.size  # Returns: 6

# Data type of elements
arr.dtype # Returns: dtype('int64')
```

## 4. Indexing & Slicing

Access and subset array elements.

### 1D Arrays

Works just like Python lists.

```
a = np.array([10, 20, 30, 40, 50])

a[0]        # Returns: 10
a[2:4]      # Returns: array([30, 40])
a[-1]       # Returns: 50
```

### 2D Arrays (Matrices)

Use `[row, col]` or `[row][col]`.

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

# Get a single element
b[1, 2]     # Returns: 6

# Get a row
b[0, :]     # Returns: array([1, 2, 3])

# Get a column
b[:, 1]     # Returns: array([2, 5, 8])

# Get a sub-matrix (rows 0-1, cols 1-2)
b[0:2, 1:3] # Returns: array([[2, 3], [5, 6]])
```

### Boolean Indexing

Use a boolean mask to select elements that meet a condition.

```
# Create an array
data = np.array([[1, 2], [3, 4], [5, 6]])

# Create a boolean mask
mask = (data > 3)
# mask is:
# [[False, False],
#  [False,  True],
#  [ True,  True]]

# Apply the mask
data[mask] # Returns: array([4, 5, 6])
```

## 5. Basic Operations

### Element-wise Arithmetic

Operations are applied to each element individually. Arrays must have compatible shapes.

```
x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])

# Element-wise operations
x + y  # or np.add(x, y)
x - y  # or np.subtract(x, y)
x * y  # or np.multiply(x, y)
x / y  # or np.divide(x, y)
x ** 2 # Squaring each element
```

### Matrix Multiplication

Use the `@` operator for dot product.

```
x @ y
# is equivalent to
np.dot(x, y)
```

# NumPy Beginner Cheat Sheet

### Universal Functions (UFuncs)

Functions that operate element-wise on an array.

```
a = np.array([0, np.pi/2, np.pi])

np.sin(a)   # Sine of each element
np.cos(a)   # Cosine of each element
np.sqrt(x)  # Square root of each element
np.exp(x)   # e^(element)
```

## 6. Aggregate Functions

Perform calculations over an entire array or along an axis.

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

m.sum()      # Sum of all elements. Returns: 21
m.min()      # Minimum element. Returns: 1
m.max()      # Maximum element. Returns: 6
m.mean()     # Mean of all elements. Returns: 3.5
m.std()      # Standard deviation.
```

### Operations along an Axis

`axis=0` collapses columns, `axis=1` collapses rows.

```
# Sum along columns (result is 1D array)
m.sum(axis=0) # Returns: array([5, 7, 9])
# Calculation: [1+4, 2+5, 3+6]

# Sum along rows (result is 1D array)
m.sum(axis=1) # Returns: array([6, 15])
# Calculation: [1+2+3, 4+5+6]
```

## 7. Reshaping & Manipulating

Change the shape and structure of arrays.

### Reshaping

Change the shape of an array without changing its data.

```
a = np.arange(6) # array([0, 1, 2, 3, 4, 5])

# Reshape to a 2x3 matrix
b = a.reshape(2, 3)
# [[0, 1, 2],
#  [3, 4, 5]]
```

### Transpose

Flip the axes of an array.

```
# b is from the previous example
b.T # Transpose of b
# [[0, 3],
#  [1, 4],
#  [2, 5]]
```

### Flattening

Collapse an array to one dimension.

```
# b is a 2x3 matrix
b.ravel()   # Returns array([0, 1, 2, 3, 4, 5])
```

### Stacking

Join a sequence of arrays.

```
a = np.array([1, 2])
b = np.array([3, 4])

# Stack vertically (row-wise)
np.vstack((a, b))
# [[1, 2],
#  [3, 4]]

# Stack horizontally (column-wise)
np.hstack((a, b))
# [1, 2, 3, 4]

# General concatenation
np.concatenate((a,b), axis=0) # axis=0 is default
```