NumPy Cheat Sheet

Import

```
import numpy as np
```

Creating arrays and matrices

```
np.array([1,2,3])                          (1×3)-Array
np.array([[6,5,4],[3,2,1]]                 (2×3)-Array
np.array([[[8],[7]],[[2],[3]],[[5],[4]]])  (3×2×1)-Array
```

```
np.zeros((3,2),dtype=int)   ⇒ array([[0,0],[0,0],[0,0]])
np.ones((2,2),dtype=float)  ⇒ array([[1.,1.],[1.,1.]])
np.arange(10, 30, 5)        ⇒ array([10, 15, 20, 25])
np.linspace(0, 0.6, 4)      ⇒ array([0.0, 0.2, 0.4, 0.6])
np.full((2,2), 3)           ⇒ array([[3, 3], [3, 3])
np.eye(3)                   ⇒ array([[1,0,0], [0,1,0], [0,0,1]])
```

Array inspection

```
a = np.array([[3,5,7], [4,1,9]])
a.shape                          ⇒ (2,3)
a.ndim                           ⇒ 2
a.size                           ⇒ 6
a.dtype                          ⇒ 'int64'
```

Array manipulation

```
a = np.array([1, 2, 3, 4, 5, 6])
a.resize(3, 2)                   ⇒ array([[1,2],[3,4],[5,6]])
b = np.transpose(a)              ⇒ array([[1,3,5],[2,4,6]])
c = np.ravel(a)                  ⇒ array([1,3,5,2,4,6])
c.resize(6,1)                    ⇒ array([[1],[3],[5],[2],[4],[6]])
d = np.array([[1,2],[3,4]])
e = np.array([[5],[6]])
f = np.hstack((d,e))             ⇒ array([[1,2,5], [3,4,6]])
g = np.array([7,8,9])
h = np.vstack((f,g))             ⇒ array([[1,2,5],[3,4,6],[7,8,9]])
i = np.append([1, 2, 3],[4, 5])  ⇒ array([1,2,3,4,5])
j = np.delete(i, [1,3])          ⇒ array([1,3,5])
k = np.insert(j, 2, 7)           ⇒ array([1,3,7,5])
```

Statistics

```
data = np.array([[1, 1, 7], [4, 6, 2]])
np.mean(data)                    ⇒ 3.5
np.mean(data, axis=0)            ⇒ array([2.5, 3.5, 4.5])
np.mean(data, axis=1))           ⇒ array([3.0, 4.0])
np.median(data) n                ⇒ 3.0
np.var(data, ddof=0)             ⇒ 5.583... [delta degrees of freedom]
np.var(data, ddof=1)             ⇒ 6.7
np.std(data, ddof=0)             ⇒ 2.362...
np.std(data, ddof=1)             ⇒ 2.588...
np.max([[1,4,3],[2,5,6]])        ⇒ 6
np.max([[1,4,3],[2,5,6]], axis=0)  ⇒ [2 5 6]
np.max([[1,4,3],[2,5,6]], axis=1)  ⇒ [4 6]
```

Random numbers

```
rng = np.random.default_rng(123)        Random number generator (seed=123)
rng.random(size=2)                      ⇒ [0.68235186 0.05382102]
rng.standard_normal(size=2)             ⇒ [1.28792526 0.19397442]
rng.integers(low=0, high=3, size=10)    ⇒ [1 0 1 2 1 2 1 0 2 2]
```

Input/Output

```
a = np.array([[3,5,7], [4,1,6]])
np.save('array_a.npy', a)
b = np.load('array_a.npy')   ⇒ array([[3,5,7], [4,1,6]])

c = np.array([[8,2,4], [5,1,9]])
np.savetxt('array_c.csv', c, delimiter=';')
d = np.loadtxt('array_c.csv', delimiter=';')   ⇒ array([[8,2,4], [5,1,9]])
```

Mathematical functions

```
np.sin([0, np.pi/2, np.pi])    ⇒ [0.0 1.0 0.0] analog
np.cos([0, np.pi/2, np.pi])    ⇒ [ 1.0 0.0 -1.0]
np.tan([0, np.pi/2, np.pi])    ⇒ [ 0.0 1.63312394e+16 0.0]
np.degrees(np.arcsin(0.5))     ⇒ 30.000000000000004
np.degrees(np.arctan(-1))      ⇒ -45.0
np.round([1/3, 2/3, 5/9], 2)   ⇒ [0.33 0.67 0.56]
np.floor(np.sqrt([2, 5, 8]))   ⇒ [1. 2. 2.]
np.ceil(np.sqrt([2, 5, 8]))    ⇒ [2. 3. 3.]
np.exp(1)                      ⇒ 2.718281828459045
np.log(np.exp([5, 0, 7]))      ⇒ [5. 0. 7.]
np.power([1,2,3], 2)           ⇒ [1 4 9]
np.power([1,2,3], [4, 3, 2])   ⇒ [1 8 9]
np.mod([5,8,7,1], 2)           ⇒ [1 0 1 1]
np.mod([5,4,7,8], [3,2,3,2])   ⇒ [2 0 1 0]
```

Linear algbra

```
import numpy.linalg as linalg
u = np.array([2,8])
v = np.array([1,5])
np.dot(u,v)                    ⇒ 42
np.inner(u,v)                  ⇒ 42
np.outer(u,v)                  ⇒ array([[2,10], [8,40]])
A = np.array([[1,2],[3,4]])
B = np.array([[2,3],[3,2]])
10 * A                         ⇒ array([[10,20],[30,40]])
A + B                          ⇒ array([[3,5],[6,6]])
A - B                          ⇒ array([[-1,-1],[0,2]])
A * B                          ⇒ array([[2,6],[9,8]]) (elementwise multiplication)
A @ B                          ⇒ array([[8,7],[18,17]]) (matrix multiplication)
linalg.det(A)                  ⇒ -2.0
linalg.solve(A, u)             ⇒ array([4,-1])
linalg.inv(A)                  ⇒ array([[-2.0,1.0],[1.5,-0.5]])
linalg.eig(B)                  ⇒ EigResult(eigenvalues=array([5.0,-1.0], ...
                                   eigenvectors=array([[...],[...]]))
```

https://numpy.org/doc/stable/