1) Numpy Arrays

Array Basics

```
# 1D array
```

```
import numpy as np
```

```
my_list = [1,2,3,4,5]
print(my_list)
```

```
     [1, 2, 3, 4, 5]
```

```
np.array(my_list)
```

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

```
arr = np.array(my_list)
```

```
arr
```

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

```
# 2D array
```

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

```
my_mat
```

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

```
np.array(my_mat)
```

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

NumPy arrays using built in methods

```
import numpy as np
```

```
# 1) arange
```

```
np.arange(0,11)
```

```
     array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

```
np.arange(0,11,2)
```

```
     array([ 0,  2,  4,  6,  8, 10])
```

```
# 2) zeros
```

```
np.zeros(3) # 1D array
```

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

```
np.zeros((3,4))
```

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

```
# 3) ones
```

```
np.ones(4)
```

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

```
np.ones((4,5))
```

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

```
# 4) linspace
```

```
np.linspace(0,5,10)
```

```
array([0.        , 0.55555556, 1.11111111, 1.66666667, 2.22222222,
       2.77777778, 3.33333333, 3.88888889, 4.44444444, 5.        ])
```

```
np.linspace(0,5,15)
```

```
array([0.        , 0.35714286, 0.71428571, 1.07142857, 1.42857143,
       1.78571429, 2.14285714, 2.5       , 2.85714286, 3.21428571,
       3.57142857, 3.92857143, 4.28571429, 4.64285714, 5.        ])
```

```
# 5) eye
```

```
np.eye(5)
```

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

```
# 6) arrays of random numbers (uniform distribution between 0 to 1)
```

```
np.random.rand(5)
```

```
array([0.26475666, 0.48708223, 0.14965471, 0.39197214, 0.23405896])
```

```
np.random.rand(4,4)
```

```
array([[0.99940642, 0.50532145, 0.00683209, 0.73367479],
       [0.01149229, 0.1850161 , 0.59158927, 0.60638651],
       [0.97326936, 0.16573969, 0.83455322, 0.30367799],
       [0.27740123, 0.93981466, 0.5485414 , 0.41814832]])
```

```
# 7) Standard normal distribution or gaussion distribution
```

```
np.random.randn(4) # 1D array
```

```
array([-0.42419987,  2.52306948,  0.79784337, -0.09758585])
```

```
np.random.randn(4,5)
```

```
array([[ 0.36147993,  0.50700741, -0.22990934,  1.94422456, -0.84571415],
       [-0.4611244 , -0.14122759,  0.0765803 ,  0.39646086, -0.73749054],
       [-2.74393596, -0.85946341,  0.29097433, -0.21048446, -0.49858556],
       [ 0.73248134, -0.35512905, -1.1080982 ,  0.12139467, -1.35670216]])
```

```
# 8) randint
```

```
np.random.randint(0,100)
```

```
57
```

```
np.random.randint(0,100,10)
```

```
array([92, 48,  6, 57, 98, 29, 33, 20, 36, 43])
```

Useful attributes and methods

```
arr = np.arange(25)
```

```
arr
```

```
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])
```

```
# reshape method
```

```
arr.reshape(5,5)
```

```
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]])
```

```
arr.reshape(5,10)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-37-67c8f5e0ef4f> in <cell line: 1>()
----> 1 arr.reshape(5,10)

ValueError: cannot reshape array of size 25 into shape (5,10)
```

EXPLAIN ERROR

```
# 2) max, min
```

```
ran_arr = np.random.randint(0,50,10)
```

```
ran_arr
```

```
array([34, 25,  7, 41,  7, 23, 48, 19, 28,  1])
```

```
ran_arr.max()
```

```
48
```

```
ran_arr.min()
```

```
1
```

```
ran_arr.argmax()
```

```
6
```

```
ran_arr.argmin()
```

```
9
```

```
# 3) shape
```

```
arr.shape
```

```
(25,)
```

```
b = arr.reshape(5,5)
```

```
b.shape
```

```
(5, 5)
```

```
# 4) dtype
```

```
arr.dtype
```

```
dtype('int64')
```

2) Numpy Indexing and Selection

Indexing and Selection Basics

```
import numpy as np
```

```
arr = np.arange(10,21)
```

```
arr
```

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20])
```

```
arr[5]
```

```
15
```

```
arr[0:5]
```

```
array([10, 11, 12, 13, 14])
```

```
arr[2:]
```

```
array([12, 13, 14, 15, 16, 17, 18, 19, 20])
```

```
arr[:6]
```

```
array([10, 11, 12, 13, 14, 15])
```

broadcasting the values

```
arr
```

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20])
```

```
arr[0:5] = 100
```

```
arr
```

```
array([100, 100, 100, 100, 100,  15,  16,  17,  18,  19,  20])
```

```
arr = np.arange(10,21)
```

```
arr
```

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20])
```

```
slice_arr = arr[0:6]
```

```
slice_arr
```

```
array([10, 11, 12, 13, 14, 15])
```

```
slice_arr[:] = 99
```

```
slice_arr
```

```
array([99, 99, 99, 99, 99, 99])
```

```
arr
```

```
array([99, 99, 99, 99, 99, 99, 16, 17, 18, 19, 20])
```

```
arr_cp = arr.copy()
```

```
arr_cp
```

```
array([99, 99, 99, 99, 99, 99, 16, 17, 18, 19, 20])
```

```
arr[:] = 100
```

```
arr
```

```
array([100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100])
```

```
arr_cp
```

```
array([99, 99, 99, 99, 99, 99, 16, 17, 18, 19, 20])
```

```
arr = np.arange(0,11)
```

```
arr
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

```
slice_2 = arr[0:5]
```

```
slice_2
```

```
array([0, 1, 2, 3, 4])
```

```
slice_2[:] = 100
```

```
slice_2
```

```
array([100, 100, 100, 100, 100])
```

```
arr
```

```
array([100, 100, 100, 100, 100,   5,   6,   7,   8,   9,  10])
```

```
arr_cp2 = arr.copy()
```

```
arr_cp2[:] = 200
```

```
arr_cp2
```

```
array([200, 200, 200, 200, 200, 200, 200, 200, 200, 200, 200])
```

```
arr
```

```
array([100, 100, 100, 100, 100,   5,   6,   7,   8,   9,  10])
```

Indexing a 2D array(Matrix)

```
import numpy as np
```

```
arr_2d = np.array([[5,10,15],[20,25,30],[35,40,45]])
```

```
arr_2d
```

```
array([[ 5, 10, 15],
       [20, 25, 30],
       [35, 40, 45]])
```

```
arr_2d[0,0]
```

```
5
```

```
arr_2d[1,0]
```

```
20
```

```
arr_2d[0:2, 1:3]
```

```
array([[10, 15],
       [25, 30]])
```

conditional selection

```
arr = np.arange(0,11)
```

```
arr
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

```
arr > 5
```

```
array([False, False, False, False, False, False,  True,  True,  True,
        True,  True])
```

```
b = arr > 5
```

```
b
```

```
array([False, False, False, False, False, False,  True,  True,  True,
        True,  True])
```

```
arr[b]
```

```
array([ 6,  7,  8,  9, 10])
```

```
arr[arr < 8]
```

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

3) Numpy Operations

```
import numpy as np
```

```
# array with array
```

```
arr = np.arange(0,11)
print(arr)

     [ 0  1  2  3  4  5  6  7  8  9 10]
```

```
arr + arr

     array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20])
```

```
arr - arr

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

```
arr * arr

     array([  0,   1,   4,   9,  16,  25,  36,  49,  64,  81, 100])
```

```
arr / arr

     <ipython-input-103-7f952cd3e0ce>:1: RuntimeWarning: invalid value encountered in divide
       arr / arr
     array([nan,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.])
```

```
# array with scalar
```

```
arr + 100

     array([100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110])
```

```
arr - 50

     array([-50, -49, -48, -47, -46, -45, -44, -43, -42, -41, -40])
```

```
arr *2

     array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20])
```

```
arr/2

     array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. ])
```

```
1/0

     ---------------------------------------------------------------------------
     ZeroDivisionError                         Traceback (most recent call last)
     <ipython-input-110-9e1622b385b6> in <cell line: 1>()
     ----> 1 1/0

     ZeroDivisionError: division by zero
```

EXPLAIN ERROR

```
1/arr

     <ipython-input-111-016353831300>:1: RuntimeWarning: divide by zero encountered in divide
       1/arr
     array([       inf, 1.        , 0.5       , 0.33333333, 0.25      ,
            0.2       , 0.16666667, 0.14285714, 0.125     , 0.11111111,
            0.1       ])
```

```
arr**3

     array([   0,    1,    8,   27,   64,  125,  216,  343,  512,  729, 1000])
```

```
# universal array functions
```

```
np.sqrt(arr)
```

```
array([0.        , 1.        , 1.41421356, 1.73205081, 2.        ,
       2.23606798, 2.44948974, 2.64575131, 2.82842712, 3.        ,
       3.16227766])
```

```
np.exp(arr)
```

```
array([1.00000000e+00, 2.71828183e+00, 7.38905610e+00, 2.00855369e+01,
       5.45981500e+01, 1.48413159e+02, 4.03428793e+02, 1.09663316e+03,
       2.98095799e+03, 8.10308393e+03, 2.20264658e+04])
```

```
np.max(arr)
```

```
10
```

```
np.min(arr)
```

```
0
```

```
np.sin(arr)
```

```
array([ 0.        ,  0.84147098,  0.90929743,  0.14112001, -0.7568025 ,
       -0.95892427, -0.2794155 ,  0.6569866 ,  0.98935825,  0.41211849,
       -0.54402111])
```

```
np.log(arr)
```

```
<ipython-input-119-a67b4ae04e95>:1: RuntimeWarning: divide by zero encountered in log
  np.log(arr)
array([      -inf, 0.        , 0.69314718, 1.09861229, 1.38629436,
       1.60943791, 1.79175947, 1.94591015, 2.07944154, 2.19722458,
       2.30258509])
```

Start coding or generate with AI.