APSSDC

Andhra Pradesh State Skill Development Corporation

(https://apssdc.in)

Data Manipulation with NumPy

Data Manipulation with NumPy

1. Numpy Basics
2. Math
3. Random
4. Indexing
5. Filtering
6. Statistics
7. Aggregation
8. Saving/Retriving Data

In [1]:

```
import numpy as np
```

In [2]:

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

In [3]:

```
arr
```

Out[3]:

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

```
for i in range(initiliztion = 0, condition, step = +1)
```

In [6]:

```
arr2 = np.arange(100)
print(arr2)
arr3 = np.arange(5, 101, 5)
print(arr3)

arr4 = np.arange(1, 10, 0.5)
print(arr4)
```

```
[ 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 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
 96 97 98 99]
[  5  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85  90
  95 100]
[1.  1.5 2.  2.5 3.  3.5 4.  4.5 5.  5.5 6.  6.5 7.  7.5 8.  8.5 9.  9.5]
```

np.linspace -> used for creating n numbers between a,b with equal spacing/difference

In [7]:

```
np.linspace(1,10,100)
```

Out[7]:

```
array([ 1.        ,  1.09090909,  1.18181818,  1.27272727,  1.36363636,
        1.45454545,  1.54545455,  1.63636364,  1.72727273,  1.81818182,
        1.90909091,  2.        ,  2.09090909,  2.18181818,  2.27272727,
        2.36363636,  2.45454545,  2.54545455,  2.63636364,  2.72727273,
        2.81818182,  2.90909091,  3.        ,  3.09090909,  3.18181818,
        3.27272727,  3.36363636,  3.45454545,  3.54545455,  3.63636364,
        3.72727273,  3.81818182,  3.90909091,  4.        ,  4.09090909,
        4.18181818,  4.27272727,  4.36363636,  4.45454545,  4.54545455,
        4.63636364,  4.72727273,  4.81818182,  4.90909091,  5.        ,
        5.09090909,  5.18181818,  5.27272727,  5.36363636,  5.45454545,
        5.54545455,  5.63636364,  5.72727273,  5.81818182,  5.90909091,
        6.        ,  6.09090909,  6.18181818,  6.27272727,  6.36363636,
        6.45454545,  6.54545455,  6.63636364,  6.72727273,  6.81818182,
        6.90909091,  7.        ,  7.09090909,  7.18181818,  7.27272727,
        7.36363636,  7.45454545,  7.54545455,  7.63636364,  7.72727273,
        7.81818182,  7.90909091,  8.        ,  8.09090909,  8.18181818,
        8.27272727,  8.36363636,  8.45454545,  8.54545455,  8.63636364,
        8.72727273,  8.81818182,  8.90909091,  9.        ,  9.09090909,
        9.18181818,  9.27272727,  9.36363636,  9.45454545,  9.54545455,
        9.63636364,  9.72727273,  9.81818182,  9.90909091, 10.        ])
```

In [10]:

```
np.zeros(10)
```

Out[10]:

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

In [12]:

```
np.full(15, 10)
```

Out[12]:

```
array([10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10])
```

In [3]:

```
arr = np.arange(0, 25).reshape(5, 5)
print(arr)
```

```
[[ 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]]
```

In [16]:

```
np.arange(0, 50).reshape(-1, 10)
```

Out[16]:

```
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, 25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47, 48, 49]])
```

In [2]:

```
np.arange(0, 50).reshape(10, -1)
```

Out[2]:

```
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],
       [25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34],
       [35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44],
       [45, 46, 47, 48, 49]])
```

In [19]:

```
arr.shape
```

Out[19]:

```
(5, 5)
```

In [20]:

```
arr
```

Out[20]:

```
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]])
```

In [21]:

```
np.hsplit(arr, 5)
```

Out[21]:

```
[array([[ 0],
        [ 5],
        [10],
        [15],
        [20]]),
 array([[ 1],
        [ 6],
        [11],
        [16],
        [21]]),
 array([[ 2],
        [ 7],
        [12],
        [17],
        [22]]),
 array([[ 3],
        [ 8],
        [13],
        [18],
        [23]]),
 array([[ 4],
        [ 9],
        [14],
        [19],
        [24]])]
```

In [24]:

```
np.hsplit(arr, 5)[0]
```

Out[24]:

```
array([[ 0],
       [ 5],
       [10],
       [15],
       [20]])
```

In [25]:

```
type(np.hsplit(arr, 5))
```

Out[25]:

```
list
```

In [26]:

```
np.vsplit(arr, 5)
```

Out[26]:

```
[array([[0, 1, 2, 3, 4]]),
 array([[5, 6, 7, 8, 9]]),
 array([[10, 11, 12, 13, 14]]),
 array([[15, 16, 17, 18, 19]]),
 array([[20, 21, 22, 23, 24]])]
```

In [30]:

```
arr.diagonal()
```

Out[30]:

```
array([ 0,  6, 12, 18, 24])
```

In [4]:

```
arr2 = np.arange(50, 75).reshape(5, 5)
```

In [5]:

```
np.hstack((arr, arr2))
```

Out[5]:

```
array([[ 0,  1,  2,  3,  4, 50, 51, 52, 53, 54],
       [ 5,  6,  7,  8,  9, 55, 56, 57, 58, 59],
       [10, 11, 12, 13, 14, 60, 61, 62, 63, 64],
       [15, 16, 17, 18, 19, 65, 66, 67, 68, 69],
       [20, 21, 22, 23, 24, 70, 71, 72, 73, 74]])
```

In [6]:

```
np.vstack((arr, arr2))
```

Out[6]:

```
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],
       [50, 51, 52, 53, 54],
       [55, 56, 57, 58, 59],
       [60, 61, 62, 63, 64],
       [65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74]])
```

In [8]:

```
arr = np.arange(1, 10)
arr2 = np.arange(10, 20)
```

In [9]:

```
mark = [0, 5, 8, 9, 9, 8, 5.5]
```

In [10]:

```
for i in range(len(mark)):
    mark[i] = mark[i] + 1

mark
```

Out[10]:

```
[1, 6, 9, 10, 10, 9, 6.5]
```

Vectorization

In [11]:

```
marks = np.array([0, 5, 8, 9, 9, 8, 5.5])

marks
```

Out[11]:

```
array([0. , 5. , 8. , 9. , 9. , 8. , 5.5])
```

In [12]:

```
marks + 1
```

Out[12]:

```
array([ 1. ,  6. ,  9. , 10. , 10. ,  9. ,  6.5])
```

In [13]:

```
marks
```

Out[13]:

```
array([0. , 5. , 8. , 9. , 9. , 8. , 5.5])
```

In [14]:

```
marks = marks + 1
```

In [15]:

```
marks
```

Out[15]:

```
array([ 1. ,  6. ,  9. , 10. , 10. ,  9. ,  6.5])
```

In [16]:

```
marks * 2
```

Out[16]:

```
array([ 2., 12., 18., 20., 20., 18., 13.])
```

In [17]:

```
marks > 5
```

Out[17]:

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

Math

In [18]:

```
degrees = np.arange(0, 361, 15)
degrees
```

Out[18]:

```
array([  0,  15,  30,  45,  60,  75,  90, 105, 120, 135, 150, 165, 180,
       195, 210, 225, 240, 255, 270, 285, 300, 315, 330, 345, 360])
```

In [19]:

```
np.sin(degrees)
```

Out[19]:

```
array([ 0.        ,  0.65028784, -0.98803162,  0.85090352, -0.30481062,
       -0.38778164,  0.89399666, -0.97053528,  0.58061118,  0.08836869,
       -0.71487643,  0.99779728, -0.80115264,  0.21945467,  0.46771852,
       -0.93009488,  0.94544515, -0.50639163, -0.17604595,  0.77387159,
       -0.99975584,  0.74513326, -0.13238163, -0.54399582,  0.95891572])
```

In [20]:

```
rad = np.deg2rad(degrees)
rad
```

Out[20]:

```
array([0.        , 0.26179939, 0.52359878, 0.78539816, 1.04719755,
       1.30899694, 1.57079633, 1.83259571, 2.0943951 , 2.35619449,
       2.61799388, 2.87979327, 3.14159265, 3.40339204, 3.66519143,
       3.92699082, 4.1887902 , 4.45058959, 4.71238898, 4.97418837,
       5.23598776, 5.49778714, 5.75958653, 6.02138592, 6.28318531])
```

In [21]:

```
np.sin(rad)
```

Out[21]:

```
array([ 0.00000000e+00,  2.58819045e-01,  5.00000000e-01,  7.07106781e-01,
        8.66025404e-01,  9.65925826e-01,  1.00000000e+00,  9.65925826e-01,
        8.66025404e-01,  7.07106781e-01,  5.00000000e-01,  2.58819045e-01,
        1.22464680e-16, -2.58819045e-01, -5.00000000e-01, -7.07106781e-01,
       -8.66025404e-01, -9.65925826e-01, -1.00000000e+00, -9.65925826e-01,
       -8.66025404e-01, -7.07106781e-01, -5.00000000e-01, -2.58819045e-01,
       -2.44929360e-16])
```

In [22]:

```
np.cos(rad)
```

Out[22]:

```
array([ 1.00000000e+00,  9.65925826e-01,  8.66025404e-01,  7.07106781e-01,
        5.00000000e-01,  2.58819045e-01,  6.12323400e-17, -2.58819045e-01,
       -5.00000000e-01, -7.07106781e-01, -8.66025404e-01, -9.65925826e-01,
       -1.00000000e+00, -9.65925826e-01, -8.66025404e-01, -7.07106781e-01,
       -5.00000000e-01, -2.58819045e-01, -1.83697020e-16,  2.58819045e-01,
        5.00000000e-01,  7.07106781e-01,  8.66025404e-01,  9.65925826e-01,
        1.00000000e+00])
```

In [23]:

```
np.tan(rad)
```

Out[23]:

```
array([ 0.00000000e+00,  2.67949192e-01,  5.77350269e-01,  1.00000000e+00,
        1.73205081e+00,  3.73205081e+00,  1.63312394e+16, -3.73205081e+00,
       -1.73205081e+00, -1.00000000e+00, -5.77350269e-01, -2.67949192e-01,
       -1.22464680e-16,  2.67949192e-01,  5.77350269e-01,  1.00000000e+00,
        1.73205081e+00,  3.73205081e+00,  5.44374645e+15, -3.73205081e+00,
       -1.73205081e+00, -1.00000000e+00, -5.77350269e-01, -2.67949192e-01,
       -2.44929360e-16])
```

In [24]:

```
np.exp(np.arange(0, 10))
```

Out[24]:

```
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])
```

In [25]:

```
np.log(np.arange(0, 10))
```

```
<ipython-input-25-c51cba2d2baf>:1: RuntimeWarning: divide by zero encountere
d in log
  np.log(np.arange(0, 10))
```

Out[25]:

```
array([      -inf, 0.        , 0.69314718, 1.09861229, 1.38629436,
       1.60943791, 1.79175947, 1.94591015, 2.07944154, 2.19722458])
```

In [26]:

```
np.dot(np.arange(0, 10), np.arange(0, 10))
```

Out[26]:

```
285
```

In [30]:

```
np.matmul(np.arange(0, 10).reshape(5, 2), np.arange(0, 10).reshape(2, 5))
```

Out[30]:

```
array([[  5,   6,   7,   8,   9],
       [ 15,  20,  25,  30,  35],
       [ 25,  34,  43,  52,  61],
       [ 35,  48,  61,  74,  87],
       [ 45,  62,  79,  96, 113]])
```

In [32]:

```
np.sum(np.arange(0, 10))
```

Out[32]:

```
45
```

Random

In [111]:

```
np.random.random()
```

Out[111]:

```
0.07273721088434781
```

In [38]:

```
np.random.randint(1, 100)
```

Out[38]:

```
62
```

In [43]:

```
np.random.rand(1, 10) #Uniformly distributed data between 0, 1
```

Out[43]:

```
array([[0.62616541, 0.71234335, 0.52992766, 0.37617105, 0.77755274,
        0.99232841, 0.50305619, 0.06935285, 0.63113231, 0.01321219]])
```

In [46]:

```
np.random.seed(13)

np.random.rand(1, 10)
```

Out[46]:

```
array([[0.77770241, 0.23754122, 0.82427853, 0.9657492 , 0.97260111,
        0.45344925, 0.60904246, 0.77552651, 0.64161334, 0.72201823]])
```

In [48]:

```
np.random.randn(1, 10) # Standard Nominal Mean = 0, std = 1
```

Out[48]:

```
array([[ 0.60628866, -0.02677165, -0.98416078,  1.19070527,  0.95283061,
        -1.08718159, -0.14521133,  0.23785784, -1.63909341, -0.27813452]])
```

Accessing of our data using indexing

In [52]:

```
a2 = np.arange(0, 10).reshape(2,5)

a2
```

Out[52]:

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

In [54]:

```
a2[0]
```

Out[54]:

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

In [55]:

```
a2[0][0]
```

Out[55]:

```
0
```

In [56]:

```
a2[0][2:4]
```

Out[56]:

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

In [60]:

```
a2[:, 2:4]
```

Out[60]:

```
array([[2, 3],
       [7, 8]])
```

In [61]:

```
a2[:, -1]
```

Out[61]:

```
array([4, 9])
```

In [62]:

```
a2[:, -2:]
```

Out[62]:

```
array([[3, 4],
       [8, 9]])
```

Filtering

In [64]:

```
a3 = np.arange(50, 100)
a3
```

Out[64]:

```
array([50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66,
       67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83,
       84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
```

In [65]:

```
a3 % 2 == 0
```

Out[65]:

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

In [66]:

```
a3[a3 % 2 == 0]
```

Out[66]:

```
array([50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82,
       84, 86, 88, 90, 92, 94, 96, 98])
```

In [74]:

```
(a3 > 60)
```

Out[74]:

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

In [75]:

```
a3[(a3 > 60)]
```

Out[75]:

```
array([61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,
       78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94,
       95, 96, 97, 98, 99])
```

In [71]:

```
a3[(a3 > 60) & (a3 <= 90) & (a3 % 2 == 0)]
```

Out[71]:

```
array([62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90])
```

In [72]:

```
a3[~ (a3 > 60) & (a3 <= 90) & (a3 % 2 == 0)]
```

Out[72]:

```
array([50, 52, 54, 56, 58, 60])
```

In [73]:

```
a3[~ ((a3 > 60) & (a3 <= 90) & (a3 % 2 == 0))]
```

Out[73]:

```
array([50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 63, 65, 67, 69, 71,
       73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 92, 93, 94, 95, 96, 97, 98,
       99])
```

In [76]:

```
a2[(a3 > 60)]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-76-c96338d291e5> in <module>
----> 1 a2[(a3 > 60)]

IndexError: boolean index did not match indexed array along dimension 0; dim
ension is 2 but corresponding boolean dimension is 50
```

In [77]:

```
a3[~ ((a3 > 60) & (a3 <= 90) & (a3 % 2 == 0))][:5]
```

Out[77]:

```
array([50, 51, 52, 53, 54])
```

Statistics

In [78]:

```
a3
```

Out[78]:

```
array([50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66,
       67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83,
       84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
```

In [79]:

```
a3.mean()
```

Out[79]:

```
74.5
```

In [80]:

```
np.median(a3)
```

Out[80]:

```
74.5
```

In [82]:

```
np.percentile(a3, 25)
```

Out[82]:

```
62.25
```

In [83]:

```
np.percentile(a3, 50)
```

Out[83]:

```
74.5
```

In [84]:

```
np.percentile(a3, 75)
```

Out[84]:

```
86.75
```

In [85]:

```
np.percentile(a3, 100), np.percentile(a3, 0)
```

Out[85]:

```
(99.0, 50.0)
```

In [88]:

```
a3.std()
```

Out[88]:

```
14.430869689661812
```

Agg

In [86]:

```
a3
```

Out[86]:

```
array([50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66,
       67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83,
       84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
```

In [87]:

```
a3.sum()
```

Out[87]:

```
3725
```

In [91]:

```
np.prod(a3)
```

Out[91]:

```
0
```

In [93]:

```
a3.max()
```

Out[93]:

```
99
```

In [94]:

```
a3.min()
```

Out[94]:

```
50
```

In [95]:

```
a3.argmax()
```

Out[95]:

```
49
```

In [96]:

```
a3.argmin()
```

Out[96]:

```
0
```

In [97]:

```
a3.cumsum()
```

Out[97]:

```
array([  50,  101,  153,  206,  260,  315,  371,  428,  486,  545,  605,
        666,  728,  791,  855,  920,  986, 1053, 1121, 1190, 1260, 1331,
       1403, 1476, 1550, 1625, 1701, 1778, 1856, 1935, 2015, 2096, 2178,
       2261, 2345, 2430, 2516, 2603, 2691, 2780, 2870, 2961, 3053, 3146,
       3240, 3335, 3431, 3528, 3626, 3725], dtype=int32)
```

```
a3.cumprod()
```

In [100]:

```
np.save("a3.npy", a3)
```

In [101]:

```
np.load('a3.npy')
```

Out[101]:

```
array([50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66,
       67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83,
       84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
```

In [102]:

```
len(dir(np)), len(dir(a3))
```

Out[102]:

```
(619, 162)
```

Pandas

- Creating Data
- Import/Exort data from one format to other - .CSV -> .XLSX, DB -> .XLSX, HTML -> CSV, XLSX, DB, JSON, Pickle, HDF5
- Cleaning of data
- Analysis on data
- Visualization

In [ ]:

```
pip install pandas
```

In [103]:

```
import pandas as pd
```

In [105]:

```
s1 = pd.Series([1, 2, 3, 4, 6])
```

In [106]:

```
type(s1)
```

Out[106]:

```
pandas.core.series.Series
```

In [107]:

```
s1.dtypes
```

Out[107]:

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

In [108]:

```
s2 = pd.Series([1, 2, 3, 4, 6], dtype = "int32")

s2.dtypes
```

Out[108]:

```
dtype('int32')
```