Introduction to numpy, pandas, and matplotlib

In [1]:
```
import numpy as np

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

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

In [2]:
```
type(array)
```

Out[2]:
```
numpy.ndarray
```

In [3]:
```
for val in array:
    print(val, type(val))
```

```
1 <class 'numpy.int32'>
2 <class 'numpy.int32'>
3 <class 'numpy.int32'>
4 <class 'numpy.int32'>
5 <class 'numpy.int32'>
```

np.arrange(start,finish,interval)

In [4]:
```
range_array = np.arange(0,101)
print(range_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  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  99 100]
```

In [5]:
```
range_array = np.arange(0,101, 3)
print(range_array)
```

```
[ 0  3  6  9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69
 72 75 78 81 84 87 90 93 96 99]
```

In [6]:
```
range_array = np.arange(101, 0, -3)
print(range_array)
```

```
[101  98  95  92  89  86  83  80  77  74  71  68  65  62  59  56  53  50
  47  44  41  38  35  32  29  26  23  20  17  14  11   8   5   2]
```

forming an array with a list of different values

In [8]:
```
int_float_list = [1,2,3,4,5]
print("int_float_list")
for val in int_float_list:
    print(val, type(val))
    mystery_array = np.array(int_float_list)
    print()
    print("mystery array")
    for val in mystery_array:
        print(val,type(val))

float_list = [1.,2.,3.,4.,5.]
```

```
int_float_list
1 <class 'int'>

mystery array
1 <class 'numpy.int32'>
2 <class 'numpy.int32'>
3 <class 'numpy.int32'>
4 <class 'numpy.int32'>
5 <class 'numpy.int32'>
2 <class 'int'>

mystery array
1 <class 'numpy.int32'>
2 <class 'numpy.int32'>
3 <class 'numpy.int32'>
4 <class 'numpy.int32'>
5 <class 'numpy.int32'>
3 <class 'int'>

mystery array
1 <class 'numpy.int32'>
2 <class 'numpy.int32'>
3 <class 'numpy.int32'>
4 <class 'numpy.int32'>
5 <class 'numpy.int32'>
4 <class 'int'>

mystery array
1 <class 'numpy.int32'>
2 <class 'numpy.int32'>
3 <class 'numpy.int32'>
4 <class 'numpy.int32'>
5 <class 'numpy.int32'>
5 <class 'int'>

mystery array
1 <class 'numpy.int32'>
2 <class 'numpy.int32'>
3 <class 'numpy.int32'>
4 <class 'numpy.int32'>
5 <class 'numpy.int32'>
```

In [10]:
```
int_float_list = [1,2,3,4,5]
print("int_float_list")
for val in int_float_list:
    print(val, type(val))
mystery_array = np.array(int_float_list)
print()

print("mystery array")
for val in mystery_array:
        print(val,type(val))
print()

print("float_array")
float_list = [1.,2.,3.,4.,5.]
float_array=np.array(float_list)
for val in float_array:
    print(val,type(val))
```

```
int_float_list
1 <class 'int'>
2 <class 'int'>
3 <class 'int'>
4 <class 'int'>
5 <class 'int'>

mystery array
1 <class 'numpy.int32'>
2 <class 'numpy.int32'>
3 <class 'numpy.int32'>
4 <class 'numpy.int32'>
5 <class 'numpy.int32'>

float_array
1.0 <class 'numpy.float64'>
2.0 <class 'numpy.float64'>
3.0 <class 'numpy.float64'>
4.0 <class 'numpy.float64'>
5.0 <class 'numpy.float64'>
```

Two Dimensional Lists and Arrays

In [11]:
```
two_dim_list = [[1, 2, 3], [2, 3, 4], [5, 6, 7]]
print(two_dim_list)
```

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

In [12]:
```
for val in two_dim_list:
    print(val)
```

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

In [13]:
```
for lst in two_dim_list:
    print(lst)
```

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

In [14]:
```
for lst in two_dim_list:
    print(lst)
    for val in lst:
        print(val)
```

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

transform two dimensional list to two dimensional using np.array(lst)

In [16]:
```
two_dim_array = np.array(two_dim_list)
two_dim_array
```

Out[16]:
```
array([[1, 2, 3],
       [2, 3, 4],
       [5, 6, 7]])
```

In [17]:
```
for array in two_dim_array:
    print(array)
    for val in array:
        print(val)
```

```
[1 2 3]
1
2
3
[2 3 4]
2
3
4
[5 6 7]
5
6
7
```

Changing the values of elements in an array

In [19]:
```
array = np.zeros((3,3,3))
array
```

Out[19]:
```
array([[[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]],

       [[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]],

       [[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]]])
```

In [20]:
```
array = np.zeros((3,3,2))
array
```

Out[20]:
```
array([[[0., 0.],
        [0., 0.],
        [0., 0.]],

       [[0., 0.],
        [0., 0.],
        [0., 0.]],

       [[0., 0.],
        [0., 0.],
        [0., 0.]]])
```

In [21]:
```
array = np.zeros((3,3))
array
```

Out[21]:
```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

In [23]:
```
array[0][0] = 1
array[1][1] = 3
array[2][2] = 2

array
```

Out[23]:
```
array([[1., 0., 0.],
       [0., 3., 0.],
       [0., 0., 2.]])
```

np.zeros creates an n-dimensional array (array of arrays) that is comprised of zeros that are floats.

If you want to create an n-dimensional array of zeros that are ints, you could create an n-dimensional list using zeros that are integers

In [24]:
```
zero_lst_of_lsts = [[0,0,0], [0,0,0], [0,0,0]]
np.array(zero_lst_of_lsts)
```

Out[24]:
```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

In [29]:
```
zero_lst_of_lsts = [[0,0,0], [0,0,0], [0,0,0]]
zero_array_of_arrays = np.array(zero_lst_of_lsts)
zero_array_of_arrays
```

Out[29]:
```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

In [30]:
```
zero_array_of_arrays[0][0] = 4
zero_array_of_arrays
```

Out[30]:
```
array([[4, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

Logging Data

In [32]:
```
import numpy as np

np.log(np.e)
```

Out[32]:
```
1.0
```

In [33]:
```
np.log10(10)
```

Out[33]:
```
1.0
```

In [35]:
```
np.log10(100)
```

Out[35]:
```
2.0
```

In [36]:
```
for i in range (1, 1000, 10):
    print(np.log10(i))
```

```
0.0
1.0413926851582251
1.3222192947339193
1.4913616938342726
1.6127838567197355
1.7075701760979363
1.7853298350107671
1.8512583487190752
1.9084850188786497
1.9590413923210936
2.0043213737826426
2.0453229787866576
2.0827853703164503
2.1172712956557644
2.1492191126553797
2.1789769472931693
2.2068258760318495
2.2329961103921536
2.2576785748691846
2.2810333672477277
2.303196057420489
2.3242824552976926
2.3443922736851106
2.3636119798921444
2.3820170425748683
2.399673721481038
2.416640507338281
2.432969290874406
2.44870631990508
2.4638929889859074
2.4785664955938436
2.4927603890268375
2.506505032404872
2.519827993775719
2.5327543789924976
2.545307116465824
2.5575072019056577
2.569373909615046
2.5809249756756194
2.5921767573958667
2.603144372620182
2.6138418218760693
2.6242820958356683
2.6344772701607315
2.6444385894678386
2.6541765418779604
2.663700925389648
2.673020907128896
2.682145076373832
2.6910814921229687
2.699837725867246
2.708420900134713
2.7168377232995247
2.725094521081469
```

```
2.7331972651065692
2.741151598851785
2.7489628612561616
2.756636108245848
2.7641761323903307
2.7715874808812555
2.7788744720027396
2.786041210242554
2.79309160017658
2.8000293592441343
2.8068580295188172
2.813580988568192
2.82020145948564
2.826722520168992
2.833147111912785
2.8394780473741985
2.8457180179666586
2.851869600729766
2.857935264719429
2.8639173769578603
2.869818207979328
2.8756399370041685
2.8813846567705728
2.8870543780509568
2.8926510338773004
2.8981764834976764
2.9036325160842376
2.909020854211156
2.9143431571194407
2.919601023784111
2.924795995797912
2.929929560084588
2.935003151453655
2.9400181550076634
2.9449759084120477
2.949877704036875
2.954724790979063
2.9595183769729982
2.964259630196849
2.9689496809813427
2.973589623427257
2.978180516937414
2.9827233876685453
2.9872192299080047
2.9916690073799486
2.9960736544852753
```

In [37]:
```
for i in range (10, 1000, 10):
    print(i, np.log10(i))
```

```
10 1.0
20 1.3010299956639813
30 1.4771212547196624
40 1.6020599913279623
50 1.6989700043360187
60 1.7781512503836436
70 1.845098040014257
80 1.9030899869919435
90 1.954242509439325
100 2.0
110 2.041392685158225
120 2.0791812460476247
130 2.113943352306837
140 2.146128035678238
150 2.1760912590556813
160 2.2041199826559246
170 2.230448921378274
180 2.255272505103306
190 2.278753600952829
200 2.3010299956639813
210 2.322219294733919
220 2.342422680822206
230 2.361727836017593
240 2.380211241711606
250 2.3979400086720375
260 2.4149733479708178
270 2.4313637641589874
280 2.4471580313422194
290 2.462397997898956
300 2.4771212547196626
310 2.4913616938342726
320 2.505149978319906
330 2.5185139398778875
340 2.531478917042255
350 2.5440680443502757
360 2.5563025007672873
370 2.568201724066995
380 2.57978359661681
390 2.591064607026499
400 2.6020599913279625
410 2.6127838567197355
420 2.6232492903979003
430 2.6334684555795866
440 2.6434526764861874
450 2.6532125137753435
460 2.662757831681574
470 2.6720978579357175
480 2.681241237375587
490 2.690196080028514
500 2.6989700043360187
510 2.7075701760979363
520 2.716003343634799
530 2.724275869600789
540 2.7323937598229686
```

```
550 2.7403626894942437
560 2.7481880270062002
570 2.7558748556724915
580 2.7634279935629373
590 2.7708520116421442
600 2.7781512503836434
610 2.785329835010767
620 2.792391689498254
630 2.7993405494535817
640 2.806179973983887
650 2.8129133566428557
660 2.8195439355418688
670 2.8260748027008264
680 2.832508912706236
690 2.838849090737255
700 2.845098040014257
710 2.8512583487190755
720 2.8573324964312685
730 2.863322860120456
740 2.8692317197309762
750 2.8750612633917
760 2.8808135922807914
770 2.886490725172482
780 2.8920946026904804
790 2.8976270912904414
800 2.9030899869919438
810 2.90848501887865
820 2.9138138523837167
830 2.9190780923760737
840 2.9242792860618816
850 2.929418925714293
860 2.934498451243568
870 2.9395192526186187
880 2.9444826721501687
890 2.949390006644913
900 2.9542425094393248
910 2.9590413923210934
920 2.963787827345555
930 2.9684829485539352
940 2.9731278535996988
950 2.9777236052888476
960 2.9822712330395684
970 2.9867717342662448
980 2.9912260756924947
990 2.99563519459755
```

In [38]:
```
for i in range (10, 1001, 10):
    print(i, np.log10(i))
```

```
10 1.0
20 1.3010299956639813
30 1.4771212547196624
40 1.6020599913279623
50 1.6989700043360187
60 1.7781512503836436
70 1.845098040014257
80 1.9030899869919435
90 1.954242509439325
100 2.0
110 2.041392685158225
120 2.0791812460476247
130 2.113943352306837
140 2.146128035678238
150 2.1760912590556813
160 2.2041199826559246
170 2.230448921378274
180 2.255272505103306
190 2.278753600952829
200 2.3010299956639813
210 2.322219294733919
220 2.342422680822206
230 2.361727836017593
240 2.380211241711606
250 2.3979400086720375
260 2.4149733479708178
270 2.4313637641589874
280 2.4471580313422194
290 2.462397997898956
300 2.4771212547196626
310 2.4913616938342726
320 2.505149978319906
330 2.5185139398778875
340 2.531478917042255
350 2.5440680443502757
360 2.5563025007672873
370 2.568201724066995
380 2.57978359661681
390 2.591064607026499
400 2.6020599913279625
410 2.6127838567197355
420 2.6232492903979003
430 2.6334684555795866
440 2.6434526764861874
450 2.6532125137753435
460 2.662757831681574
470 2.6720978579357175
480 2.681241237375587
490 2.690196080028514
500 2.6989700043360187
510 2.7075701760979363
520 2.716003343634799
530 2.724275869600789
540 2.7323937598229686
```

```
550 2.7403626894942437
560 2.7481880270062002
570 2.7558748556724915
580 2.7634279935629373
590 2.7708520116421442
600 2.7781512503836434
610 2.785329835010767
620 2.792391689498254
630 2.7993405494535817
640 2.806179973983887
650 2.8129133566428557
660 2.8195439355418688
670 2.8260748027008264
680 2.832508912706236
690 2.838849090737255
700 2.845098040014257
710 2.8512583487190755
720 2.8573324964312685
730 2.863322860120456
740 2.8692317197309762
750 2.8750612633917
760 2.8808135922807914
770 2.886490725172482
780 2.8920946026904804
790 2.8976270912904414
800 2.9030899869919438
810 2.90848501887865
820 2.9138138523837167
830 2.9190780923760737
840 2.9242792860618816
850 2.929418925714293
860 2.934498451243568
870 2.9395192526186187
880 2.9444826721501687
890 2.949390006644913
900 2.9542425094393248
910 2.9590413923210934
920 2.963787827345555
930 2.9684829485539352
940 2.9731278535996988
950 2.9777236052888476
960 2.9822712330395684
970 2.9867717342662448
980 2.9912260756924947
990 2.99563519459755
1000 3.0
```