Numpy Lecture - 1 [Airbnb NPS]

In [3]:
```
!pip install numpy
```

```
DEPRECATION: Configuring installation scheme with distutils config files is deprecated and will no longer work in the near future. If you are using a Homebrew or Linuxbrew Python, please see discussion at https://github.com/Homebrew/homebrew-core/issues/76621
Requirement already satisfied: numpy in /usr/local/lib/python3.9/site-packages (1.22.3)
DEPRECATION: Configuring installation scheme with distutils config files is deprecated and will no longer work in the near future. If you are using a Homebrew or Linuxbrew Python, please see discussion at https://github.com/Homebrew/homebrew-core/issues/76621
```

In [4]:
```
import numpy as np
```

In [5]:
```
a = [1, 2, 3, 4, 5]
a = [i**2 for i in a]
print(a)
```

```
[1, 4, 9, 16, 25]
```

In [6]:
```
a = np.array([1, 2, 3, 4, 5])
a**2
```

Out[6]:
```
array([ 1,  4,  9, 16, 25])
```

In [7]:
```
l = [1, 2, 3, 4, 5]
l*2
```

Out[7]:
```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

In [8]:
```
lst = range(100000)
%timeit [i**2 for i in lst]
```

```
43.7 ms ± 4.99 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

In [9]:
```
arr1 = np.array(range(100000))
%timeit arr1**2
```

```
44 µs ± 4.26 µs per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
```

In [10]:
```
#1- D numpy array

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

Out[10]:
```
array([1, 2, 3])
```

In [11]:
```
arr.ndim
```

Out[11]:
```
1
```

In [12]:
```
arr.shape
```

Out[12]:
```
(3,)
```

In [13]:
```
arr2 = np.array([[1, 2, 3], [4, 5, 6]])
arr2.ndim
```

Out[13]:
```
2
```

In [14]:
```
arr2.shape
```

Out[14]:
```
(2, 3)
```

In [15]:
```
arr.size
```

Out[15]:
```
3
```

In [16]:
```
arr2.size
```

Out[16]:
```
6
```

In [17]:
```
len(arr2)
```

Out[17]:
```
2
```

In [18]:
```
# Sequence
```

In [19]:
```
l = range(1, 1000, 0.5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [19], in <cell line: 1>()
----> 1 l = range(1, 1000, 0.5)

TypeError: 'float' object cannot be interpreted as an integer
```

In [20]:
```
np.arange(1000)
```

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,  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, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
       143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
       169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
       195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
       208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
       221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233,
       234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246,
       247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259,
       260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272,
       273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285,
       286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298,
       299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311,
       312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324,
       325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337,
       338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350,
       351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363,
       364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376,
       377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389,
       390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402,
       403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415,
       416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428,
       429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441,
       442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454,
       455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467,
       468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480,
       481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493,
       494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506,
       507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519,
       520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532,
       533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545,
       546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558,
       559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571,
       572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584,
       585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597,
       598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610,
       611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623,
       624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636,
       637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649,
       650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662,
       663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675,
       676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688,
       689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701,
       702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714,
       715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727,
       728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740,
       741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753,
       754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766,
       767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779,
       780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792,
       793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805,
       806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818,
       819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831,
       832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844,
       845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857,
       858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870,
       871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883,
       884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896,
       897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909,
       910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922,
       923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935,
       936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948,
       949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961,
       962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974,
       975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987,
       988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999])
```

In [21]:

```
a = np.arange(1, 10, 0.5)
```

```
print(a)
```

```
[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]
```

In [22]:
```
a = np.array([1, 2, 3, 4.0])
print(a)
```

```
[1. 2. 3. 4.]
```

In [23]:
```
a = np.array([1, 2, 'satish'])
print(a)
```

```
['1' '2' 'satish']
```

In [24]:
```
A = [1, 2, 3, 4]
type(A)
```

Out[24]:
```
list
```

In [25]:
```
type(a)
```

Out[25]:
```
numpy.ndarray
```

In [26]:
```
a.dtype
```

Out[26]:
```
dtype('<U21')
```

In [27]:
```
arr.dtype
```

Out[27]:
```
dtype('int64')
```

In [28]:
```
arr = np.arange(12)
arr[1]
```

Out[28]:
```
1
```

In [29]:
```
arr[12]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [29], in <cell line: 1>()
----> 1 arr[12]

IndexError: index 12 is out of bounds for axis 0 with size 12
```

In [30]:
```
arr.size
```

Out[30]:
```
12
```

In [31]:
```
arr[-1]
```

Out[31]:
```
11
```

In [32]:
```
arr[[1, 2, 3, 2, 2]]
```

Out[32]:
```
array([1, 2, 3, 2, 2])
```

In [33]:
```
arr = np.arange(1, 12)
arr[[1, 2, 3, 2, 2]]
```

Out[33]:
```
array([2, 3, 4, 3, 3])
```

Slicing

In [34]:
```
a = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
a[:5]
```

Out[34]:
```
array([1, 2, 3, 4, 5])
```

In [35]:
```
a[-5:-1]
```

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

In [36]:
```
a[-5:-1:-1]
```

```
array([], dtype=int64)
```

Operations on Numpy Array

In [37]:
```
arr = np.arange(4)
arr
```

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

In [38]:
```
arr+3
```

```
array([3, 4, 5, 6])
```

In [39]:
```
arr*2
```

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

In [41]:
```
a = np.arange(4)
b = np.array([1, 2, 3, 4])
a+ b
```

```
array([1, 3, 5, 7])
```

Universal Functions

In [42]:
```
a = np.array([1, 2, 3, 4])

np.add(a, 2)
```

```
array([3, 4, 5, 6])
```

In [43]:
```
a = np.arange(1, 11)
np.sum(a)
```

```
55
```

In [44]:
```
np.mean(a)
```

```
5.5
```

In [45]:
```
np.min(a)
```

```
1
```

In [46]:
```
np.max(a)
```

```
10
```

In [47]:
```
import numpy as np
a = np.array([1,2,3,4,5,6,7,8])
print(a.ndim, a.shape)
```

```
1 (8,)
```

In [48]:
```
import numpy as np

arr = np.arange(5)
arr[2:4] = 0

print(arr)
```

```
[0 1 0 0 4]
```

In [49]:
```
a = [1,2,3,4,5]
b = [8,7,6]
a[2:] = b[::-1]
print(a)
```

```
[1, 2, 6, 7, 8]
```

In [50]:
```
a, b, c = [1, 2, 3]
print(a, b, c)
```

```
1 2 3
```

In [ ]:
```

```