Data Science & AI
&
NIC - Param

Python-For Data Science

Numpy

Lecture No.- 01

By- Pankaj Sharma Sir

Recap of Previous Lecture

Topic Trees Part 04

Topics to be Covered

Topic NumPy Part 01

Topic : NumPy

Numpy → H.W

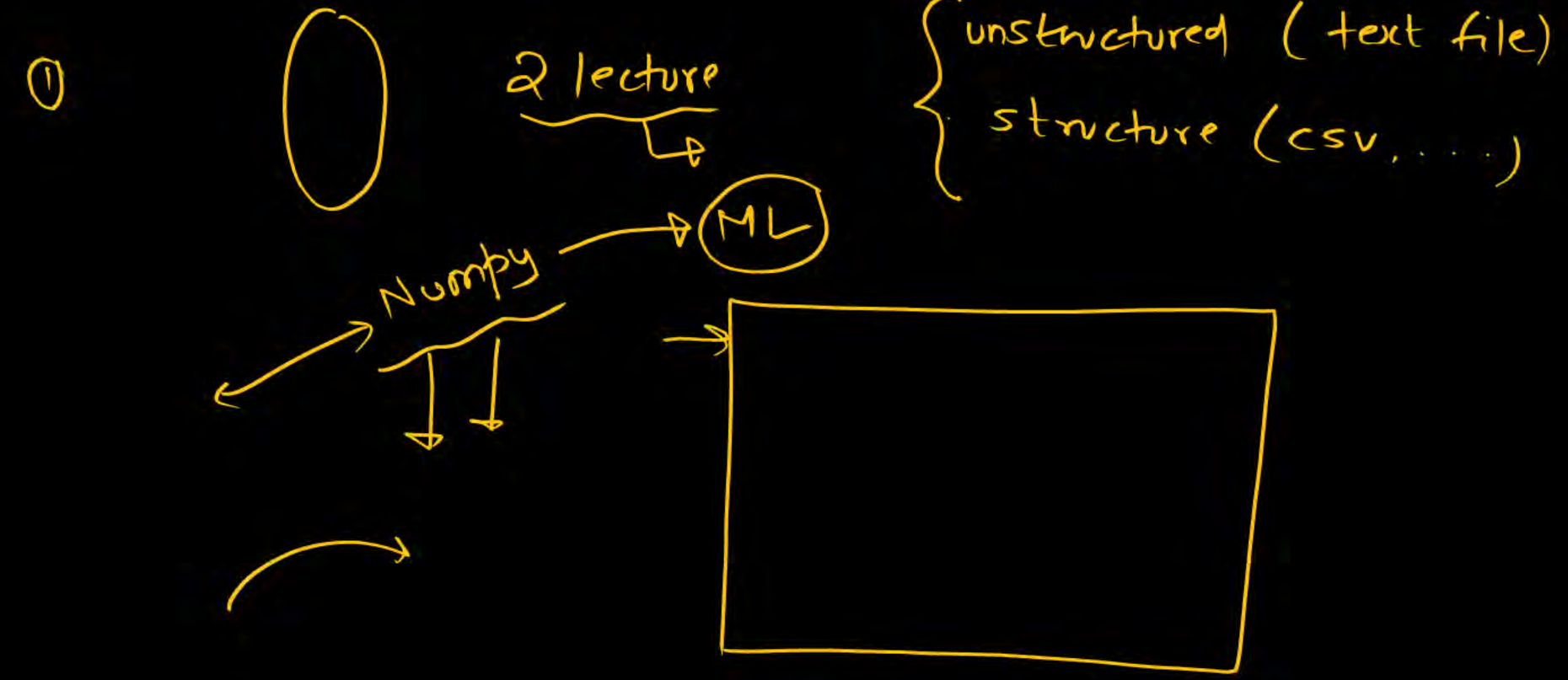
- 1d
- 2d ... nd

1d array

- list in python
- Vs
- Numpy array

Why ?

1) Memory
2) Time
3) Easy to use


1
2 lecture
unstructured (text file)
structure (csv, ...)
ML
Numpy

np.array( ) →

np.ones( ) → All 1s

np.zeros( ) → All 0s

np.full( ) → particular ele

np.empty( ) → random ele.

np.arange( )

np.linspace( )

$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Square matrix

np.identity(3)

$$I = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

np.eye( )

Slicing and indexing

→ broadcasting

In [7]:
```
import numpy as np #aliasing np is an alias for numpy
```

In [8]:
```
l=[i for i in range(1000)]
```

In [9]:
```
n_arr=np.arange(1000)
```

In [10]:
```
print(l)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 2
3, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 4
4, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 6
5, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 8
6, 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 [11]:
```
print(n_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  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 [12]:
```
print(n_arr.itemsize) #size of each element in numpy array
```

```
4
```

In [13]:
```
print(n_arr.size) #size of array ==>total elements in numpy array
```

```
1000
```

In [14]:
```
print(n_arr.itemsize * n_arr.size)
```

```
4000
```

In [15]:
```
import sys
```

In [16]:
```
a=4
print(sys.getsizeof(a))
```

```
28
```

In [17]:
```
print(sys.getsizeof(a) * len(l))
```

```
28000
```

In [18]:
```
#prooved that a numpy array takes less memory as compared to list in python
```

In [19]:
```
#time
```

In [25]:
```
import time
```

In [26]:
```
a=[i for i in range(1000000)]   #list1
b=[2*i for i in range(1000000)] #list2   each of size 10^6
```

In [27]:
```
t1=time.time()
```

In [28]:
```
c=[a[i]-b[i] for  i in range(1000000)] #how much time
```

In [29]:
```
t2=time.time()
```

In [30]:
```
print(t2-t1)  #time in milliseconds ==>to subtract 2 lists of size 10^6
```

```
31.41450047492981
```

In [31]:
```
n1=np.arange(1000000)
n2=np.arange(1000000)
```

In [32]:
```
t3=time.time()
```

In [33]:
```
n3=n1 - n2
```

In [34]:
```
t4=time.time()
```

In [35]:
```
print(t4-t3)
```

```
9.560009717941284
```

In [36]:
```
#prove ===>execution time on operations on numpy array takes less time
#easy
```

In [37]:
```
#how to create a numpy array
#1 ===>already data/sequence/list se array

#np.array(something like array )
```

In [38]:
```
a=[10,20,30]
b=np.array(a)
```

In [39]:
```
print(b)
```

```
[10 20 30]
```

In [40]:
```
print(type(b))
```

```
<class 'numpy.ndarray'>
```

In [41]:
```
x=[1,2,3,5.46,6.47]
```

In [42]:
```
y=np.array(x)
```

In [43]:
```
print(y)#in numpy array all elements are of same type
```

```
[1.   2.   3.   5.46 6.47]
```

In [44]:
```
y=np.array(x,dtype=int)
```

In [45]:
```
y
```

Out[45]:
```
array([1, 2, 3, 5, 6])
```

In [46]:
```
print(y)
```

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

In [47]:
```
l=[1,2,3,4.56,7.890,'pankaj']
narr=np.array(l)
```

In [48]:
```
print(narr)
```

```
['1' '2' '3' '4.56' '7.89' 'pankaj']
```

In [49]:
```
c=[1,2,3,4,5]
narr1=np.array(c,dtype=str)
```

In [50]:
```
print(narr1)
```

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

In [51]:
```
l2=[1,2,3,4,5]
l2*3
```

Out[51]:
```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

In [52]:
```
narr2=np.array(l2*3)
```

In [53]:
```
print(narr2)
```

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

In [54]:
```
#np.ones   ==>numpy array with all elements as 1
x=np.ones(4)
```

In [55]:
```
print(x)
```

```
[1. 1. 1. 1.]
```

In [56]:
```
x=np.ones(4,dtype=int)
print(x)
```

```
[1 1 1 1]
```

In [57]:
```
#np.zeros()==>numpy array with all elements as 0
a=np.zeros(4)
print(a)
```

```
[0. 0. 0. 0.]
```

In [58]:
```
a=np.zeros(4,dtype=int)
print(a)
```

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

In [59]:
```
#2d array
x=np.ones((3,2),dtype=int) #create an array with 3 rows and 2 cols with all elements a
```

In [60]:
```
print(x)
```

```
[[1 1]
 [1 1]
 [1 1]]
```

In [61]:
```
y=np.zeros((2,3),dtype=int)
```

In [62]:
```
print(y)
```

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

In [63]:
```
#how to create an array with some other element
#[6 6 6 6]  ==>all the elements are 6
x=np.full(4,6)  # create a numpy array with 4 elements and each element is 6
```

In [64]:
```
print(x)
```

```
[6 6 6 6]
```

In [65]:
```
y=np.full((2,3),10,dtype=float)
```

In [66]:
```
print(y)
```

```
[[10. 10. 10.]
 [10. 10. 10.]]
```

In [73]:
```
a=np.empty(4,dtype=int)#random values ka array
```

In [74]:
```
print(a)
```

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

In [75]:
```
b=np.arange(8)  # 0 to8 step=1 0,1,2,3,4,5,6,7
```

In [76]:
```
b
```

Out[76]:
```
array([0, 1, 2, 3, 4, 5, 6, 7])
```

In [77]:
```
c=np.arange(20)
c
```

Out[77]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19])
```

In [78]:
```
c=np.arange(3,20)#20 is not included
```

In [79]:
```
c
```

Out[79]:
```
array([ 3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

In [80]:
```
c=np.arange(3,20,3) # 3 ==>3+3==>3+3+3 ==>3+3+3+3 ==>....
```

In [81]:
```
c
```

Out[81]:
```
array([ 3,  6,  9, 12, 15, 18])
```

In [82]:
```
x=np.linspace(3,20)
```

In [83]:
```
x
```

Out[83]:
```
array([ 3.        ,  3.34693878,  3.69387755,  4.04081633,  4.3877551 ,
        4.73469388,  5.08163265,  5.42857143,  5.7755102 ,  6.12244898,
        6.46938776,  6.81632653,  7.16326531,  7.51020408,  7.85714286,
        8.20408163,  8.55102041,  8.89795918,  9.24489796,  9.59183673,
        9.93877551, 10.28571429, 10.63265306, 10.97959184, 11.32653061,
       11.67346939, 12.02040816, 12.36734694, 12.71428571, 13.06122449,
       13.40816327, 13.75510204, 14.10204082, 14.44897959, 14.79591837,
       15.14285714, 15.48979592, 15.83673469, 16.18367347, 16.53061224,
       16.87755102, 17.2244898 , 17.57142857, 17.91836735, 18.26530612,
       18.6122449 , 18.95918367, 19.30612245, 19.65306122, 20.        ])
```

In [84]:
```
x[1]-x[0]
```

Out[84]:
```
0.34693877551020424
```

In [85]:
```
x[2]-x[1]
```

Out[85]:
```
0.3469387755102038
```

In [86]:
```
b=np.linspace(3,20,6)
```

In [87]:
```
b
```

Out[87]:
```
array([ 3. ,  6.4,  9.8, 13.2, 16.6, 20. ])
```

In [88]:
```
b=np.linspace(3,20,6,dtype=int)
```

In [89]:
```
b
```

Out[89]:
```
array([ 3,  6,  9, 13, 16, 20])
```

In [90]:
```
a=np.identity(3)
print(a)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [91]:
```
b=np.eye(3)
```

In [92]:
```
b
```

Out[92]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [93]:
```
c=np.eye(2,3)
```

In [94]:
```
print(c)
```

```
[[1. 0. 0.]
 [0. 1. 0.]]
```

In [95]:
```
#Hw ==>file handling
#text files===>open()
#read()
#readlines()

a=np.random.rand(3)
print(a)
```

```
[0.91672264 0.74853495 0.43946197]
```

In [96]:
```
a=np.random.rand(3,4)
print(a)
```

```
[[0.69053909 0.59038835 0.33236476 0.11798353]
 [0.14475388 0.84757008 0.24313744 0.0716367 ]
 [0.02557186 0.16594273 0.62066217 0.90917017]]
```

In [97]:
```
a=np.random.rand(4,5)
print(a)
```

```
[[0.22932046 0.85995316 0.90868459 0.03965647 0.98493872]
 [0.47152132 0.79556074 0.71598082 0.21158939 0.2563894 ]
 [0.01898268 0.09702994 0.24307797 0.15367706 0.8442878 ]
 [0.67128968 0.17255432 0.62398457 0.33965338 0.2313071 ]]
```

In [ ]:

THANK - YOU