

14.2_Numpy(1)

April 19, 2023

1 Introduction to Python for Open Source Geocomputation



- Instructor: Dr. Wei Kang
- Class Location and Time: ENV 336, Mon & Wed 12:30 pm - 1:50 pm

Content:

- Numpy
- A new data type: `numpy.array`
 - How to create an array
 - Array operations

2 What is Numpy?

- The fundamental package for scientific computing with Python
- Nearly every scientist working in Python draws on the power of NumPy.
- NumPy brings the **computational power** of languages like C and Fortran to Python, a language much easier to learn and use. With this power comes **simplicity: a solution in NumPy is often clear and elegant.**
- Essential in many different realms:
 - NumPy lies at the core of a rich ecosystem of **data science** libraries
 - NumPy forms the basis of powerful **machine learning** libraries like [scikit-learn](#), [SciPy](#), [TensorFlow](#), and [PyTorch](#)
 - NumPy is an essential component in the burgeoning Python **visualization landscape**, which includes Matplotlib, Seaborn, Plotly, Altair, Bokeh, Holoviz, Vispy, Napari, and PyVista, to name a few.

2.1 What makes Numpy so important?

arrays: A very powerful data type essential to numerical computing: * sequences of data all of the *same type* * behave a lot like lists, except for the constraint in the type of their elements. * There is a huge efficiency advantage when you know that **all elements of a sequence are of the same type**—so equivalent methods for arrays execute a lot **faster** than those for lists.

2.2 Numpy Array (or ndarray)

- homogeneous multidimensional array
 - a table of elements (usually numbers), all of the same type, indexed by a tuple of non-negative integers
 - * for the data types accepted in Numpy. Read the [docs: Data type objects](#).
 - dimensions are called *axes*
- An Example: points' coordinates
 - one single point: one-dimensional array: `np.array([1,2])`
 - two or more points: two-dimensional array:
 - * two points: `np.array([[1,2], [3,4]])`
 - * five points: `np.array([[1,2], [3,4], [5,6], [7,8], [9,10]])`

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

```
[2]: a1 = np.array([1,2])
a1
```

```
[2]: array([1, 2])
```

```
[3]: a2 = np.array([[1,2], [3,4], [5,6], [7,8], [9,10]])
a2
```

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

2.2.1 Motivation (1): What can a Numpy array used for?

- An array can contain:
 - values of an experiment/simulation at discrete time steps, e.g., income, air pollution, crime rate, animal/plant occurrence
 - pixels of an image, grey-level or colour
 - signal recorded by a measurement device, e.g. sound wave
 - 3-D data measured at different X-Y-Z positions, e.g. MRI scan, digital elevation model

2.2.2 Motivation (2): Efficiency of Numpy array - an example

- Problem description: Write a python program that calculate the square of each number in a list, such that $x_i = i^2$, for $0 \leq i < n$.

Two data types: * Python built-in type: list * Numpy array

We use `%timeit` to calculate the time execution of a Python statement or expression.

```
[4]: L = list(range(1000)) #produce a list of integers from 0 to 999  
L
```

```
[4]: [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]
```

```
[5]: %timeit -n 1000 [i**2 for i in L]
```

182 μ s \pm 3.77 μ s per loop (mean \pm std. dev. of 7 runs, 1,000 loops each)

```
[6]: import numpy as np  
a = np.arange(1000) #produce an array of integers from 0 to 999  
a
```

```
[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, 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])
```

```
[7]: %timeit -n 1000 a**2
```

1.1 μ s \pm 439 ns per loop (mean \pm std. dev. of 7 runs, 1,000 loops each)

3 Importing Numpy

```
import numpy as np
```

```
[8]: import numpy as np
```

```
[9]: dir(np) #function dir gives you the package's attributes and functions.
```

```
[9]: ['ALLOW_THREADS',
      'AxisError',
      'BUFSIZE',
      'CLIP',
      'ComplexWarning',
      'DataSource',
      'ERR_CALL',
      'ERR_DEFAULT',
      'ERR_IGNORE',
      'ERR_LOG',
      'ERR_PRINT',
      'ERR_RAISE',
      'ERR_WARN',
      'FLOATING_POINT_SUPPORT',
      'FPE_DIVIDEBYZERO',
      'FPE_INVALID',
      'FPE_OVERFLOW',
      'FPE_UNDERFLOW',
      'False_',
      'Inf',
      'Infinity',
      'MAXDIMS',
      'MAY_SHARE_BOUNDS',
```

```
'MAY_SHARE_EXACT',
'ModuleDeprecationWarning',
'NaN',
'NINF',
'NZERO',
'NaN',
'PINF',
'PZERO',
'RAISE',
'RankWarning',
'SHIFT_DIVIDEBYZERO',
'SHIFT_INVALID',
'SHIFT_OVERFLOW',
'SHIFT_UNDERFLOW',
'ScalarType',
'Tester',
'TooHardError',
'True_',
'UFUNC_BUFSIZE_DEFAULT',
'UFUNC_PYVALS_NAME',
'VisibleDeprecationWarning',
'WRAP',
'_CopyMode',
'_NoValue',
'_UFUNC_API',
'__NUMPY_SETUP__',
'__all__',
'__builtins__',
'__cached__',
'__config__',
'__deprecated_attrs__',
'__dir__',
'__doc__',
'__expired_functions__',
'__file__',
'__getattr__',
'__git_version__',
'__loader__',
'__name__',
'__package__',
'__path__',
'__spec__',
'__version__',
'_add_newdoc_ufunc',
'_distributor_init',
'_financial_names',
'_globals',
```

'_mat',
'_pyinstaller_hooks_dir',
'_pytesttester',
'_version',
'abs',
'absolute',
'add',
'add_docstring',
'add_newdoc',
'add_newdoc_ufunc',
'all',
'allclose',
'alltrue',
'amax',
'amin',
'angle',
'any',
'append',
'apply_along_axis',
'apply_over_axes',
'arange',
'arccos',
'arccosh',
'arcsin',
'arcsinh',
'arctan',
'arctan2',
'arctanh',
'argmax',
'argmin',
'argpartition',
'argsort',
'argwhere',
'around',
'array',
'array2string',
'array_equal',
'array_equiv',
'array_repr',
'array_split',
'array_str',
'asanyarray',
'asarray',
'asarray_chkfinite',
'ascontiguousarray',
'asfarray',
'asfortranarray',

'asmatrix',
'atleast_1d',
'atleast_2d',
'atleast_3d',
'average',
'bartlett',
'base_repr',
'binary_repr',
'bincount',
'bitwise_and',
'bitwise_not',
'bitwise_or',
'bitwise_xor',
'blackman',
'block',
'bmat',
'bool8',
'bool_',
'broadcast',
'broadcast_arrays',
'broadcast_shapes',
'broadcast_to',
'busday_count',
'busday_offset',
'busdaycalendar',
'byte',
'byte_bounds',
'bytes0',
'bytes_',
'c_',
'can_cast',
'cast',
'cbrt',
'cdouble',
'ceil',
'cfloat',
'char',
'character',
'chararray',
'choose',
'clip',
'clongdouble',
'clongfloat',
'column_stack',
'common_type',
'compare_chararrays',
'compat',

'complex128',
'complex256',
'complex64',
'complex_',
'complexfloating',
'compress',
'concatenate',
'conj',
'conjugate',
'convolve',
'copy',
'copysign',
'copyto',
'core',
'corrcoef',
'correlate',
'cos',
'cosh',
'count_nonzero',
'cov',
'cross',
'csingle',
'ctypeslib',
'cumprod',
'cumproduct',
'cumsum',
'datetime64',
'datetime_as_string',
'datetime_data',
'deg2rad',
'degrees',
'delete',
'deprecate',
'deprecate_with_doc',
'diag',
'diag_indices',
'diag_indices_from',
'diagflat',
'diagonal',
'diff',
'digitize',
'disp',
'divide',
'divmod',
'dot',
'double',
'dsplit',

'dstack',
'dtype',
'e',
'ediff1d',
'einsum',
'einsum_path',
'emath',
'empty',
'empty_like',
'equal',
'error_message',
'errstate',
'euler_gamma',
'exp',
'exp2',
'expand_dims',
'expm1',
'extract',
'eye',
'fabs',
'fastCopyAndTranspose',
'fft',
'fill_diagonal',
'find_common_type',
'finfo',
'fix',
'flatiter',
'flatnonzero',
'flexible',
'flip',
'fliplr',
'flipud',
'float128',
'float16',
'float32',
'float64',
'float_',
'float_power',
'floating',
'floor',
'floor_divide',
'fmax',
'fmin',
'fmod',
'format_float_positional',
'format_float_scientific',
'format_parser',

'frexp',
'from_dlpack',
'frombuffer',
'fromfile',
'fromfunction',
'fromiter',
'frompyfunc',
'fromregex',
'fromstring',
'full',
'full_like',
'gcd',
'generic',
'genfromtxt',
'geospace',
'get_array_wrap',
'get_include',
'get_printoptions',
'getbufsize',
'geterr',
'geterrcall',
'geterrobj',
'gradient',
'greater',
'greater_equal',
'half',
'hamming',
'hanning',
'heaviside',
'histogram',
'histogram2d',
'histogram_bin_edges',
'histogramdd',
'hsplit',
'hstack',
'hypot',
'i0',
'identity',
'iinfo',
'imag',
'in1d',
'index_exp',
'indices',
'inexact',
'inf',
'info',
'infty',

'inner',
'insert',
'int0',
'int16',
'int32',
'int64',
'int8',
'int_',
'intc',
'integer',
'interp',
'intersect1d',
'intp',
'invert',
'is_busday',
'isclose',
'iscomplex',
'iscomplexobj',
'isfinite',
'isfortran',
'isin',
'isinf',
'isnan',
'isnat',
'isneginf',
'isposinf',
'isreal',
'isrealobj',
'isscalar',
'issctype',
'issubclass_',
'issubdtype',
'issubdtype',
'iterable',
'ix_',
'kaiser',
'kron',
'lcm',
'ldexp',
'left_shift',
'less',
'less_equal',
'lexsort',
'lib',
'linalg',
'linspace',
'little_endian',

'load',
'loadtxt',
'log',
'log10',
'log1p',
'log2',
'logaddexp',
'logaddexp2',
'logical_and',
'logical_not',
'logical_or',
'logical_xor',
'logspace',
'longcomplex',
'longdouble',
'longfloat',
'longlong',
'lookfor',
'ma',
'mask_indices',
'mat',
'math',
'matmul',
'matrix',
'matrixlib',
'max',
'maximum',
'maximum_sctype',
'may_share_memory',
'mean',
'median',
'memmap',
'meshgrid',
'mgrid',
'min',
'min_scalar_type',
'minimum',
'mintypecode',
'mod',
'modf',
'moveaxis',
'msort',
'multiply',
'nan',
'nan_to_num',
'nanargmax',
'nanargmin',

'nancumprod',
'nancumsum',
'nanmax',
'nanmean',
'nanmedian',
'nanmin',
'nanpercentile',
'nanprod',
'nanquantile',
'nanstd',
'nansum',
'nanvar',
'nbytes',
'ndarray',
'ndenumerate',
'ndim',
'ndindex',
'nditer',
'negative',
'nested_iters',
'newaxis',
'nextafter',
'nonzero',
'not_equal',
'numarray',
'number',
'obj2sctype',
'object0',
'object_',
'ogrid',
'oldnumeric',
'ones',
'ones_like',
'os',
'outer',
'packbits',
'pad',
'partition',
'percentile',
'pi',
'piecewise',
'place',
'poly',
'poly1d',
'polyadd',
'polyder',
'polydiv',

'polyfit',
'polyint',
'polymul',
'polynomial',
'polysub',
'polyval',
'positive',
'power',
'printoptions',
'prod',
'product',
'promote_types',
'ptp',
'put',
'put_along_axis',
'putmask',
'quantile',
'r_',
'rad2deg',
'radians',
'random',
'ravel',
'ravel_multi_index',
'real',
'real_if_close',
'rec',
'recarray',
'recfromcsv',
'recfromtxt',
'reciprocal',
'record',
'remainder',
'repeat',
'require',
'reshape',
'resize',
'result_type',
'right_shift',
'rint',
'roll',
'rollaxis',
'roots',
'rot90',
'round',
'round_',
'row_stack',
's_',

'safe_eval',
'save',
'savetxt',
'savez',
'savez_compressed',
'sctype2char',
'sctypeDict',
'sctypes',
'searchsorted',
'select',
'set_numeric_ops',
'set_printoptions',
'set_string_function',
'setbufsize',
'setdiff1d',
'seterr',
'seterrcall',
'seterrobj',
'setxor1d',
'shape',
'shares_memory',
'short',
'show_config',
'sign',
'signbit',
'signedinteger',
'sin',
'sinc',
'single',
'singlecomplex',
'sinh',
'size',
'sometrue',
'sort',
'sort_complex',
'source',
'spacing',
'split',
'sqrt',
'square',
'squeeze',
'stack',
'std',
'str0',
'str_',
'string_',
'subtract',

'sum',
'swapaxes',
'sys',
'take',
'take_along_axis',
'tan',
'tanh',
'tensordot',
'test',
'testing',
'tile',
'timedelta64',
'trace',
'tracemalloc_domain',
'transpose',
'trapz',
'tri',
'tril',
'tril_indices',
'tril_indices_from',
'trim_zeros',
'triu',
'triu_indices',
'triu_indices_from',
'true_divide',
'trunc',
'typecodes',
'typename',
'ubyte',
'ufunc',
'uint',
'uint0',
'uint16',
'uint32',
'uint64',
'uint8',
'uintc',
'uintp',
'ulonglong',
'unicode_',
'union1d',
'unique',
'unpackbits',
'unravel_index',
'unsignedinteger',
'unwrap',
'use_hugepage',

```
'ushort',
'vander',
'var',
'vdot',
'vectorize',
'version',
'void',
'void0',
'vsplit',
'vstack',
'w',
'warnings',
'where',
'who',
'zeros',
'zeros_like']
```

3.0.1 Creating a Numpy Array

- create an array from a regular Python list or tuple using the `array` function.

```
np.array(list/tuple)
```

- functions from Numpy to create special arrays
 - `np.arange()`: create evenly spaced values within a given interval.
 - `np.linspace(start, stop, num=50)`: create evenly spaced numbers over a specified interval.
 - `np.ones(shape)`: create new array of given shape and type, filled with ones.
 - `np.zeros(shape)`: create a new array of given shape and type, filled with zeros.
 - `np.eye(N)`: create a 2-D array with ones on the diagonal and zeros elsewhere.

```
[10]: a1 = np.array([1,2])
a1
```

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

```
[11]: type(a1)
```

```
[11]: numpy.ndarray
```

```
[12]: a1.size
```

```
[12]: 2
```

`array.size` gives the number of items in the array.

```
[13]: len(a1)
```

```
[13]: 2
```

`len(array)` gives the same result to `array.size`

```
[14]: a1.ndim
```

```
[14]: 1
```

`array.ndim` gives the number of axes (dimensions) of the array.

```
[15]: a1.shape
```

```
[15]: (2,)
```

`array.shape` gives the dimensions of the array. This is a tuple of integers indicating the **size** of the array in each dimension. For a matrix with *n* rows and *m* columns, shape will be (*n*,*m*). The length of the shape tuple is therefore the number of axes, `ndim`.

```
[16]: a1.dtype
```

```
[16]: dtype('int64')
```

`array.dtype` returns an object describing the type of the elements in the array

```
[17]: a_str = np.array([1.0,2,"1"])
a_str
```

```
[17]: array(['1.0', '2', '1'], dtype='<U32')
```

```
[18]: a_str.dtype #32-character string
```

```
[18]: dtype('<U32')
```

```
[19]: a2 = np.array([[1,2], [3,4]])
a2
```

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

```
[20]: a2.ndim
```

```
[20]: 2
```

```
[21]: a2.size
```

```
[21]: 4
```

```
[22]: len(a2)
```

```
[22]: 2
```


`len(array)` gives the number of rows or the size of the first dimension when encountering a 2-dimensional array

```
[23]: a2.shape
```

```
[23]: (2, 2)
```

```
[24]: a2.dtype
```

```
[24]: dtype('int64')
```

```
[25]: a3 = np.array([[1,2], [3,4], [5,6], [7,8], [9,10]])  
a3
```

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

```
[26]: a3.ndim
```

```
[26]: 2
```

```
[27]: len(a3)
```

```
[27]: 5
```

```
[28]: a3.size
```

```
[28]: 10
```

```
[29]: a3.shape
```

```
[29]: (5, 2)
```

```
[30]: a3.dtype
```

```
[30]: dtype('int64')
```

3.1 Further reading

- read [Numpy tutorial](#) to learn more about numpy functionalities