```
## INTRODUCTION TO NUMPY

list = range(1000)
%timeit [i**2 for i in list]
```

```
225 µs ± 2.88 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

```
import numpy as np
```

```
a = np.array (range(1000))
%timeit a**2
```

```
1.05 µs ± 263 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
```

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

```
a.ndim
```

```
1
```

```
a.shape
```

```
(2, 3)
```

```
a.size
```

```
6
```

```
len(a)
```

```
2
```

```
a = np.array ([1,2,3,4,5,6,7,8])
```

```
a[a>2]
```

```
array([3, 4, 5, 6, 7, 8])
```

```
b = np.arange(0,10,0.5)
```

```
print(a)
```

```
[1 2 3 4 5 6 7 8]
```

```
a = range(0,100,0.5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-26-eac9222225c5> in <cell line: 1>()
----> 1 a = range(0,100,0.5)

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

SEARCH STACK OVERFLOW

```
a = np.arange(0,100,1.5)
```

```
print(a)
```

```
[ 0.   0.5  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 10.  10.5 11.  11.5 12.  12.5 13.  13.5
 14.  14.5 15.  15.5 16.  16.5 17.  17.5 18.  18.5 19.  19.5 20.  20.5
 21.  21.5 22.  22.5 23.  23.5 24.  24.5 25.  25.5 26.  26.5 27.  27.5
 28.  28.5 29.  29.5 30.  30.5 31.  31.5 32.  32.5 33.  33.5 34.  34.5
 35.  35.5 36.  36.5 37.  37.5 38.  38.5 39.  39.5 40.  40.5 41.  41.5
 42.  42.5 43.  43.5 44.  44.5 45.  45.5 46.  46.5 47.  47.5 48.  48.5
 49.  49.5 50.  50.5 51.  51.5 52.  52.5 53.  53.5 54.  54.5 55.  55.5
 56.  56.5 57.  57.5 58.  58.5 59.  59.5 60.  60.5 61.  61.5 62.  62.5
 63.  63.5 64.  64.5 65.  65.5 66.  66.5 67.  67.5 68.  68.5 69.  69.5
 70.  70.5 71.  71.5 72.  72.5 73.  73.5 74.  74.5 75.  75.5 76.  76.5
```

```
 77.  77.5 78.  78.5 79.  79.5 80.  80.5 81.  81.5 82.  82.5 83.  83.5
 84.  84.5 85.  85.5 86.  86.5 87.  87.5 88.  88.5 89.  89.5 90.  90.5
 91.  91.5 92.  92.5 93.  93.5 94.  94.5 95.  95.5 96.  96.5 97.  97.5
 98.  98.5 99.  99.5]
```

```
a = np.arange(1000)
```

```
(a%2==0) & (a%5==0)
```

```
       False, False, False,  True, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False,  True, False, False, False,
       False, False, False, False, False, False,  True, False, False,
       False, False, False, False, False, False, False,  True, False,
       False, False, False, False, False, False, False, False,  True,
       False, False, False, False, False, False, False, False, False,
        True, False, False, False, False, False, False, False, False,
       False,  True, False, False, False, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False,  True, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False,  True, False, False, False,
       False, False, False, False, False, False,  True, False, False,
       False, False, False, False, False, False, False,  True, False,
       False, False, False, False, False, False, False, False,  True,
       False, False, False, False, False, False, False, False, False,
        True, False, False, False, False, False, False, False, False,
       False,  True, False, False, False, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False,  True, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False,  True, False, False, False,
       False, False, False, False, False, False,  True, False, False,
       False, False, False, False, False, False, False,  True, False,
       False, False, False, False, False, False, False, False,  True,
       False, False, False, False, False, False, False, False, False,
        True, False, False, False, False, False, False, False, False,
       False,  True, False, False, False, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False,  True, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False,  True, False, False, False,
       False, False, False, False, False, False,  True, False, False,
       False, False, False, False, False, False, False,  True, False,
       False, False, False, False, False, False, False, False,  True,
       False, False, False, False, False, False, False, False, False,
        True, False, False, False, False, False, False, False, False,
       False,  True, False, False, False, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False,  True, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False,  True, False, False, False,
       False, False, False, False, False, False,  True, False, False,
       False, False, False, False, False, False, False,  True, False,
       False, False, False, False, False, False, False, False,  True,
       False, False, False, False, False, False, False, False, False,
        True, False, False, False, False, False, False, False, False,
       False,  True, False, False, False, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False,  True, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False,  True, False, False, False,
       False, False, False, False, False, False,  True, False, False,
       False, False, False, False, False, False, False,  True, False,
       False, False, False, False, False, False, False, False,  True,
       False, False, False, False, False, False, False, False, False,
        True, False, False, False, False, False, False, False, False,
       False,  True, False, False, False, False, False, False, False,
```

```
a[(a%2==0) |(a%5==0)]
```

```
array([  0,   2,   4,   5,   6,   8,  10,  12,  14,  15,  16,  18,  20,
        22,  24,  25,  26,  28,  30,  32,  34,  35,  36,  38,  40,  42,
        44,  45,  46,  48,  50,  52,  54,  55,  56,  58,  60,  62,  64,
        65,  66,  68,  70,  72,  74,  75,  76,  78,  80,  82,  84,  85,
        86,  88,  90,  92,  94,  95,  96,  98, 100, 102, 104, 105, 106,
       108, 110, 112, 114, 115, 116, 118, 120, 122, 124, 125, 126, 128,
       130, 132, 134, 135, 136, 138, 140, 142, 144, 145, 146, 148, 150,
       152, 154, 155, 156, 158, 160, 162, 164, 165, 166, 168, 170, 172,
       174, 175, 176, 178, 180, 182, 184, 185, 186, 188, 190, 192, 194,
       195, 196, 198, 200, 202, 204, 205, 206, 208, 210, 212, 214, 215,
       216, 218, 220, 222, 224, 225, 226, 228, 230, 232, 234, 235, 236,
       238, 240, 242, 244, 245, 246, 248, 250, 252, 254, 255, 256, 258,
       260, 262, 264, 265, 266, 268, 270, 272, 274, 275, 276, 278, 280,
       282, 284, 285, 286, 288, 290, 292, 294, 295, 296, 298, 300, 302,
       304, 305, 306, 308, 310, 312, 314, 315, 316, 318, 320, 322, 324,
       325, 326, 328, 330, 332, 334, 335, 336, 338, 340, 342, 344, 345,
       346, 348, 350, 352, 354, 355, 356, 358, 360, 362, 364, 365, 366,
       368, 370, 372, 374, 375, 376, 378, 380, 382, 384, 385, 386, 388,
```

```
       390, 392, 394, 395, 396, 398, 400, 402, 404, 405, 406, 408, 410,
       412, 414, 415, 416, 418, 420, 422, 424, 425, 426, 428, 430, 432,
       434, 435, 436, 438, 440, 442, 444, 445, 446, 448, 450, 452, 454,
       455, 456, 458, 460, 462, 464, 465, 466, 468, 470, 472, 474, 475,
       476, 478, 480, 482, 484, 485, 486, 488, 490, 492, 494, 495, 496,
       498, 500, 502, 504, 505, 506, 508, 510, 512, 514, 515, 516, 518,
       520, 522, 524, 525, 526, 528, 530, 532, 534, 535, 536, 538, 540,
       542, 544, 545, 546, 548, 550, 552, 554, 555, 556, 558, 560, 562,
       564, 565, 566, 568, 570, 572, 574, 575, 576, 578, 580, 582, 584,
       585, 586, 588, 590, 592, 594, 595, 596, 598, 600, 602, 604, 605,
       606, 608, 610, 612, 614, 615, 616, 618, 620, 622, 624, 625, 626,
       628, 630, 632, 634, 635, 636, 638, 640, 642, 644, 645, 646, 648,
       650, 652, 654, 655, 656, 658, 660, 662, 664, 665, 666, 668, 670,
       672, 674, 675, 676, 678, 680, 682, 684, 685, 686, 688, 690, 692,
       694, 695, 696, 698, 700, 702, 704, 705, 706, 708, 710, 712, 714,
       715, 716, 718, 720, 722, 724, 725, 726, 728, 730, 732, 734, 735,
       736, 738, 740, 742, 744, 745, 746, 748, 750, 752, 754, 755, 756,
       758, 760, 762, 764, 765, 766, 768, 770, 772, 774, 775, 776, 778,
       780, 782, 784, 785, 786, 788, 790, 792, 794, 795, 796, 798, 800,
       802, 804, 805, 806, 808, 810, 812, 814, 815, 816, 818, 820, 822,
       824, 825, 826, 828, 830, 832, 834, 835, 836, 838, 840, 842, 844,
       845, 846, 848, 850, 852, 854, 855, 856, 858, 860, 862, 864, 865,
       866, 868, 870, 872, 874, 875, 876, 878, 880, 882, 884, 885, 886,
       888, 890, 892, 894, 895, 896, 898, 900, 902, 904, 905, 906, 908,
       910, 912, 914, 915, 916, 918, 920, 922, 924, 925, 926, 928, 930,
       932, 934, 935, 936, 938, 940, 942, 944, 945, 946, 948, 950, 952,
       954, 955, 956, 958, 960, 962, 964, 965, 966, 968, 970, 972, 974,
       975, 976, 978, 980, 982, 984, 985, 986, 988, 990, 992, 994, 995,
       996, 998])
```

```
a = np.arange([100])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-35-bc8c52650056> in <cell line: 1>()
----> 1 a = np.arange([100])

TypeError: unsupported operand type(s) for -: 'list' and 'int'
```

SEARCH STACK OVERFLOW

```
a = np.arange(0,2000,2.8)
```

```
a.shape
```

```
(715,)
```

```
a.size
```

```
715
```

```
a.ndim
```

```
1
```

```
a[a>=10]
```

```
array([  11.2,   14. ,   16.8,   19.6,   22.4,   25.2,   28. ,   30.8,
         33.6,   36.4,   39.2,   42. ,   44.8,   47.6,   50.4,   53.2,
         56. ,   58.8,   61.6,   64.4,   67.2,   70. ,   72.8,   75.6,
         78.4,   81.2,   84. ,   86.8,   89.6,   92.4,   95.2,   98. ,
        100.8,  103.6,  106.4,  109.2,  112. ,  114.8,  117.6,  120.4,
        123.2,  126. ,  128.8,  131.6,  134.4,  137.2,  140. ,  142.8,
        145.6,  148.4,  151.2,  154. ,  156.8,  159.6,  162.4,  165.2,
        168. ,  170.8,  173.6,  176.4,  179.2,  182. ,  184.8,  187.6,
        190.4,  193.2,  196. ,  198.8,  201.6,  204.4,  207.2,  210. ,
        212.8,  215.6,  218.4,  221.2,  224. ,  226.8,  229.6,  232.4,
        235.2,  238. ,  240.8,  243.6,  246.4,  249.2,  252. ,  254.8,
        257.6,  260.4,  263.2,  266. ,  268.8,  271.6,  274.4,  277.2,
        280. ,  282.8,  285.6,  288.4,  291.2,  294. ,  296.8,  299.6,
        302.4,  305.2,  308. ,  310.8,  313.6,  316.4,  319.2,  322. ,
        324.8,  327.6,  330.4,  333.2,  336. ,  338.8,  341.6,  344.4,
        347.2,  350. ,  352.8,  355.6,  358.4,  361.2,  364. ,  366.8,
        369.6,  372.4,  375.2,  378. ,  380.8,  383.6,  386.4,  389.2,
        392. ,  394.8,  397.6,  400.4,  403.2,  406. ,  408.8,  411.6,
        414.4,  417.2,  420. ,  422.8,  425.6,  428.4,  431.2,  434. ,
        436.8,  439.6,  442.4,  445.2,  448. ,  450.8,  453.6,  456.4,
        459.2,  462. ,  464.8,  467.6,  470.4,  473.2,  476. ,  478.8,
        481.6,  484.4,  487.2,  490. ,  492.8,  495.6,  498.4,  501.2,
        504. ,  506.8,  509.6,  512.4,  515.2,  518. ,  520.8,  523.6,
        526.4,  529.2,  532. ,  534.8,  537.6,  540.4,  543.2,  546. ,
        548.8,  551.6,  554.4,  557.2,  560. ,  562.8,  565.6,  568.4,
```

```
       571.2,  574. ,  576.8,  579.6,  582.4,  585.2,  588. ,  590.8,
       593.6,  596.4,  599.2,  602. ,  604.8,  607.6,  610.4,  613.2,
       616. ,  618.8,  621.6,  624.4,  627.2,  630. ,  632.8,  635.6,
       638.4,  641.2,  644. ,  646.8,  649.6,  652.4,  655.2,  658. ,
       660.8,  663.6,  666.4,  669.2,  672. ,  674.8,  677.6,  680.4,
       683.2,  686. ,  688.8,  691.6,  694.4,  697.2,  700. ,  702.8,
       705.6,  708.4,  711.2,  714. ,  716.8,  719.6,  722.4,  725.2,
       728. ,  730.8,  733.6,  736.4,  739.2,  742. ,  744.8,  747.6,
       750.4,  753.2,  756. ,  758.8,  761.6,  764.4,  767.2,  770. ,
       772.8,  775.6,  778.4,  781.2,  784. ,  786.8,  789.6,  792.4,
       795.2,  798. ,  800.8,  803.6,  806.4,  809.2,  812. ,  814.8,
       817.6,  820.4,  823.2,  826. ,  828.8,  831.6,  834.4,  837.2,
       840. ,  842.8,  845.6,  848.4,  851.2,  854. ,  856.8,  859.6,
       862.4,  865.2,  868. ,  870.8,  873.6,  876.4,  879.2,  882. ,
       884.8,  887.6,  890.4,  893.2,  896. ,  898.8,  901.6,  904.4,
       907.2,  910. ,  912.8,  915.6,  918.4,  921.2,  924. ,  926.8,
       929.6,  932.4,  935.2,  938. ,  940.8,  943.6,  946.4,  949.2,
       952. ,  954.8,  957.6,  960.4,  963.2,  966. ,  968.8,  971.6,
       974.4,  977.2,  980. ,  982.8,  985.6,  988.4,  991.2,  994. ,
       996.8,  999.6, 1002.4, 1005.2, 1008. , 1010.8, 1013.6, 1016.4,
      1019.2, 1022. , 1024.8, 1027.6, 1030.4, 1033.2, 1036. , 1038.8,
      1041.6, 1044.4, 1047.2, 1050. , 1052.8, 1055.6, 1058.4, 1061.2,
      1064. , 1066.8, 1069.6, 1072.4, 1075.2, 1078. , 1080.8, 1083.6,
      1086.4, 1089.2, 1092. , 1094.8, 1097.6, 1100.4, 1103.2, 1106. ,
      1108.8, 1111.6, 1114.4, 1117.2, 1120. , 1122.8, 1125.6, 1128.4,
      1131.2, 1134. , 1136.8, 1139.6, 1142.4, 1145.2, 1148. , 1150.8,
      1153.6, 1156.4, 1159.2, 1162. , 1164.8, 1167.6, 1170.4, 1173.2,
      1176. , 1178.8, 1181.6, 1184.4, 1187.2, 1190. , 1192.8, 1195.6,
      1198.4, 1201.2, 1204. , 1206.8, 1209.6, 1212.4, 1215.2, 1218. ,
      1220.8, 1223.6, 1226.4, 1229.2, 1232. , 1234.8, 1237.6, 1240.4,
      1243.2, 1246. , 1248.8, 1251.6, 1254.4, 1257.2, 1260. , 1262.8,
      1265.6, 1268.4, 1271.2, 1274. , 1276.8, 1279.6, 1282.4, 1285.2,
      1288. , 1290.8, 1293.6, 1296.4, 1299.2, 1302. , 1304.8, 1307.6,
```

```
a<>5
```

```
  File "<ipython-input-47-0ea4cb0cd083>", line 1
    a<>5
     ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

```
print(a)
```

```
[ True  True  True  True]
```

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

```
import numpy as np
```

```
a = np.array([1,2,"Sagar",5.0,1==2])
```

```
a
```

```
array(['1', '2', 'Sagar', '5.0', 'False'], dtype='<U32')
```

```
Help on built-in function empty in module numpy:

empty(...)
    empty(shape, dtype=float, order='C', *, like=None)

    Return a new array of given shape and type, without initializing entries.

    Parameters
    ----------
    shape : int or tuple of int
        Shape of the empty array, e.g., ``(2, 3)`` or ``2``.
    dtype : data-type, optional
        Desired output data-type for the array, e.g, `numpy.int8`. Default is
        `numpy.float64`.
    order : {'C', 'F'}, optional, default: 'C'
        Whether to store multi-dimensional data in row-major
        (C-style) or column-major (Fortran-style) order in
        memory.
    like : array_like, optional
```

```
        Reference object to allow the creation of arrays which are not
        NumPy arrays. If an array-like passed in as ``like`` supports
        the ``__array_function__`` protocol, the result will be defined
        by it. In this case, it ensures the creation of an array object
        compatible with that passed in via this argument.

        .. versionadded:: 1.20.0

    Returns
    -------
    out : ndarray
        Array of uninitialized (arbitrary) data of the given shape, dtype, and
        order.  Object arrays will be initialized to None.

    See Also
    --------
    empty_like : Return an empty array with shape and type of input.
    ones : Return a new array setting values to one.
    zeros : Return a new array setting values to zero.
    full : Return a new array of given shape filled with value.


    Notes
    -----
    `empty`, unlike `zeros`, does not set the array values to zero,
    and may therefore be marginally faster.  On the other hand, it requires
    the user to manually set all the values in the array, and should be
    used with caution.

    Examples
    --------
    >>> np.empty([2, 2])
    array([[ -9.74499359e+001,   6.69583040e-309],
           [  2.13182611e-314,   3.06959433e-309]])         #uninitialized

    >>> np.empty([2, 2], dtype=int)
    array([[-1073741821, -1067949133],
           [  496041986,    19249760]])                     #uninitialized
```

```
score = np.loadtxt('survey.txt', dtype='int')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-9-a2f053d74810> in <cell line: 1>()
----> 1 score = np.loadtxt('survey.txt', dtype='int')

                                 3 frames
/usr/local/lib/python3.10/dist-packages/numpy/lib/_datasource.py in open(self, path, mode, encoding, newline)
    531                                      encoding=encoding, newline=newline)
    532         else:
--> 533             raise FileNotFoundError(f"{path} not found.")
    534
    535

FileNotFoundError: survey.txt not found.
```

SEARCH STACK OVERFLOW

```
!gdown 1c0ClC8SrPwJq5rrkyMKyPn80nyHcFikK
```

```
Downloading...
From: https://drive.google.com/uc?id=1c0ClC8SrPwJq5rrkyMKyPn80nyHcFikK
To: /content/survey.txt
100% 2.55k/2.55k [00:00<00:00, 6.25MB/s]
```

```
score = np.loadtxt('survey.txt', dtype='int')
```

```
score[score < 6]
```

```
array([5, 4, 4, 5, 1, 5, 5, 1, 4, 5, 4, 4, 4, 5, 1, 4, 1, 4, 1, 5, 5, 1,
       1, 4, 1, 5, 4, 1, 1, 4, 1, 5, 1, 4, 4, 1, 1, 1, 1, 1, 1, 4, 1,
       1, 5, 5, 5, 4, 4, 1, 4, 1, 4, 1, 5, 1, 1, 5, 4, 4, 4, 4, 1, 4, 5,
       4, 4, 1, 1, 5, 5, 1, 5, 1, 5, 5, 4, 5, 4, 1, 1, 1, 1, 4, 1, 4, 4,
       5, 4, 1, 1, 1, 1, 5, 4, 5, 5, 4, 1, 5, 1, 4, 4, 1, 1, 1, 4, 4, 5,
       5, 4, 5, 5, 5, 1, 4, 1, 5, 5, 1, 5, 1, 1, 5, 5, 4, 4, 1, 4, 4, 4,
       1, 1, 4, 4, 4, 5, 5, 1, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 5, 4, 4,
       5, 1, 4, 5, 5, 5, 1, 5, 4, 1, 1, 5, 5, 5, 4, 5, 4, 4, 1, 4, 4, 4,
       4, 5, 1, 5, 5, 1, 4, 4, 5, 1, 1, 4, 5, 5, 5, 1, 4, 5, 5, 4, 1, 5,
       5, 5, 1, 1, 5, 5, 1, 1, 1, 4, 5, 5, 4, 4, 4, 5, 1, 4, 1, 4, 5, 4,
       5, 5, 1, 5, 1, 5, 5, 1, 4, 5, 5, 4, 1, 5, 1, 4, 1, 4, 1, 1, 1, 1,
       1, 1, 4, 1, 5, 4, 5, 1, 5, 1, 5, 4, 4, 4, 4, 5, 5, 1, 4, 1, 5, 5,
       1, 4, 1, 1, 4, 4, 4, 4, 1, 4, 1, 1, 4, 1, 5, 4, 1, 1, 5, 4, 5, 4,
       4, 4, 1, 5, 5, 1, 4, 5, 4, 4, 4, 1, 4, 1, 4, 4, 4, 5, 1, 1, 1, 4,
       5, 5, 1, 5, 4, 5, 5, 4, 1, 1, 5, 5, 5, 1, 4, 5, 4, 5, 5, 5, 1, 4,
       1, 5])
```

```
a = np.array([1, 2, 3,9])
b = np.array([4, 3, 2, 1])
np.any(a > b)
```

```
True
```

```
# Window Function
```

```
a = np.array([1,,3,,5,6,7,,9,10,11,12])
a = np.arange(0,13,2)
a = a.reshape(4,3)
```

```
a.T
```

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

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

```
a =a.T
```

```
a.shape
```

```
(3, 1)
```

```
a
```

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

```
a.shape
```

```
(3, 1)
```

```
a
```

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

```
a = a.T
a
```

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

```
import numpy as np

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

```
a
```

```
array(['1', '2', '3', '4', '5', '6', '7', '8', '9', 'a'], dtype='<U21')
```

```
a = np.array([1,2,3,4,5,6,7,8,9,5.0])
```

```
a
```

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

```
a = np.array([1.0,2,3,4,5,6.0,7,8,9,"a"])
```

```
a
```

```
array(['1.0', '2', '3', '4', '5', '6.0', '7', '8', '9', 'a'], dtype='<U32')
```

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

```
a
```

```
array(['1', '2', '3', '4', '5', '6', '7', '8', '9', 'False', 'a'],
      dtype='<U21')
```

```
a = np.array ([1,2,3,4,5], dtype = "str")
```

```
a
```

```
array(['1', '2', '3', '4', '5'], dtype='<U1')
```

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

```
b = a.astype('str')
b
```

```
array(['1', '2', '3', '4', '5'], dtype='<U21')
```

```
d = np.arange(1,10)
```

```
d
```

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

```
np.where(d>5,d*5,d*10)
```

```
array([10, 20, 30, 40, 50, 30, 35, 40, 45])
```

```
!gdown 1c0ClC8SrPwJq5rrkyMKyPn80nyHcFikK
```

```
Downloading...
From: https://drive.google.com/uc?id=1c0ClC8SrPwJq5rrkyMKyPn80nyHcFikK
To: /content/survey.txt
100% 2.55k/2.55k [00:00<00:00, 8.21MB/s]
```

```
score = np.loadtxt('survey.txt',dtype='int')
```

```
score
```

```
array([ 7, 10,  5, ...,  5,  9, 10])
```

```
score.ndim
```

```
1
```

```
score.size
```

```
1167
```

```
score.shape
```

```
(1167,)
```

```
score.min()
```

```
1
```

```
score.max()
```

```
10
```

```
promoters = score[score>=9].shape[0]
detractors = score[score<=6].shape[0]
total = score.shape[0]
```

```
pop = promoters/total * 100
pod = detractors/total * 100
nps = pop-pod
nps
```

```
23.73607540702657
```

```
import numpy as np
```

```
arr = np.empty(shape=score.shape,dtype='U12')
```

```
arr
```

```
array(['', '', '', ..., '', '', ''], dtype='<U1')
```

```
arr
```

```
array(['', '', '', ..., '', '', ''], dtype='<U12')
```

```
arr[score>=9] = "promoters"
arr[score<=6] = "Detractors"
arr[(score>=7) &(score<=8)] = "passive"
```

```
arr
```

```
array(['passive', 'promoters', 'Detractors', ..., 'Detractors',
       'promoters', 'promoters'], dtype='<U12')
```

```
arr.shape
```

```
(1167,)
```

```
score.shape
```

```
(1167,)
```

```
Detractors_count = arr[arr == "Detractors"].shape[0]
Promoters_count = arr[arr == "promoters"].shape[0]
```

```
unique,count=np.unique(arr,return_counts='True')
count
```

```
array([332, 226, 609])
```

```
percent_of_detractors = count[0]/count.sum() * 100
percent_of_promoters = count[2]/count.sum() * 100
nps = percent_of_promoters - percent_of_detractors
nps
```

```
23.73607540702657
```

```
## LOGICAL FUNCTIONS
```

```
import numpy as np
```

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

```
np.any(a<b)
```

```
False
```

```
np.all(a<b)
```

```
True
```

```
a = np.arange(12)
```

```
a
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

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

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

```
b.shape
```

```
(2, 3)
```

```
a = np.arange(12).reshape(2,6)
a
```

```
array([[ 0,  1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10, 11]])
```

```
a = a.reshape(6,2)
```

```
a = a.T
```

```
a = a.T
a
```

```
array([[ 0,  2,  4,  6,  8, 10],
       [ 1,  3,  5,  7,  9, 11]])
```

```
i = np.arange(20).reshape(4,5)
i
```

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

```
i
```

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

```
i
```

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

```
i.T
```

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

```
i.reshape(5,4)
```

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

```
b = np.arange(3).reshape(1,3)
b
```

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

```
b.T
```

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

```
b.reshape(3,1)
```

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

```
b
```

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

```
a = np.array([0,1,2,3,4,5])
mask = (a%2 == 0)
a[mask] = -1
a
```

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

```
arr = np.arange(6)
arr = arr.reshape(2,3)
```

```
arr
```

```
array([[0, 1, 2],
       [3, 4, 5]])
```

```
z = arr.flatten()
```

```
z.shape
```

```
(6,)
```

```
z.reshape(2,3)
```

```
array([[0, 1, 2],
       [3, 4, 5]])
```

```
x = np.arange(1,10).reshape(3,3)
x
```

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

```
x[2,1]
```

```
8
```

```
x[0,2]
```

```
3
```

```
x[[0,1,2],[2,1,0]]
```

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

```
## Slicing on 2D array
```

```
a = np.arange(1,13).reshape(3,4)
a
```

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

```
a[:]
```

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

```
           [ 9, 10, 11, 12]])
```

```
a
```

```
    array([[ 1,  2,  3,  4],
           [ 5,  6,  7,  8],
           [ 9, 10, 11, 12]])
```

```
a[0:3,1:4]
```

```
    array([[ 2,  3,  4],
           [ 6,  7,  8],
           [10, 11, 12]])
```

```
## Fancy indexing (Masking)
```

```
a
```

```
    array([[ 1,  2,  3,  4],
           [ 5,  6,  7,  8],
           [ 9, 10, 11, 12]])
```

```
a[a>4].reshape(2,-1)
```

```
    array([[ 5,  6,  7,  8],
           [ 9, 10, 11, 12]])
```

```
## 2D - Axis
```

```
a = np.arange(1,13).reshape(3,4)
```

```
a
```

```
    array([[ 1,  2,  3,  4],
           [ 5,  6,  7,  8],
           [ 9, 10, 11, 12]])
```

```
np.min(a,axis=0)
```

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

```
np.min(a,axis=1)
```

```
    array([1, 5, 9])
```

```
np.sum(a,axis=1)
```

```
    array([10, 26, 42])
```

```
np.sort(a,axis=1)
```

```
    array([[ 1,  2,  3,  4],
           [ 5,  6,  7,  8],
           [ 9, 10, 11, 12]])
```

```
np.sort(a,axis=1)
```

```
    array([[ 1,  2,  3,  4],
           [ 5,  6,  7,  8],
           [ 9, 10, 11, 12]])
```

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

```
a
```

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

```
a.argmin()
np.argmin(a)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-f3126bd49ecd> in <cell line: 1>()
----> 1 a.argmin()
      2 np.argmin(a)

NameError: name 'a' is not defined
```

```
a.argmax()
```

```
5
```

```
a.size
```

```
6
```

```
b = np.array([8,3,9,0,2,1,5])
b
```

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

```
b.argsort()
```

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

```
!gdown 1vk1Pu0djiYcrdc85yUXZ_Rqq2oZNcohd
```

```
Downloading...
From: https://drive.google.com/uc?id=1vk1Pu0djiYcrdc85yUXZ_Rqq2oZNcohd
To: /content/fit.txt
100% 3.43k/3.43k [00:00<00:00, 15.5MB/s]
```

```
import numpy as np
!gdown 1vk1Pu0djiYcrdc85yUXZ_Rqq2oZNcohd



```

```
Downloading...
From: https://drive.google.com/uc?id=1vk1Pu0djiYcrdc85yUXZ_Rqq2oZNcohd
To: /content/fit.txt
100% 3.43k/3.43k [00:00<00:00, 10.1MB/s]
```

```
data = np.loadtxt("fit.txt",dtype ='str')
```

```
data.T[0]
```

```
array(['06-10-2017', '07-10-2017', '08-10-2017', '09-10-2017',
       '10-10-2017', '11-10-2017', '12-10-2017', '13-10-2017',
       '14-10-2017', '15-10-2017', '16-10-2017', '17-10-2017',
       '18-10-2017', '19-10-2017', '20-10-2017', '21-10-2017',
       '22-10-2017', '23-10-2017', '24-10-2017', '25-10-2017',
       '26-10-2017', '27-10-2017', '28-10-2017', '29-10-2017',
       '30-10-2017', '31-10-2017', '01-11-2017', '02-11-2017',
       '03-11-2017', '04-11-2017', '05-11-2017', '06-11-2017',
       '07-11-2017', '08-11-2017', '09-11-2017', '10-11-2017',
       '11-11-2017', '12-11-2017', '13-11-2017', '14-11-2017',
       '15-11-2017', '16-11-2017', '17-11-2017', '18-11-2017',
       '19-11-2017', '20-11-2017', '21-11-2017', '22-11-2017',
       '23-11-2017', '24-11-2017', '25-11-2017', '26-11-2017',
       '27-11-2017', '28-11-2017', '29-11-2017', '30-11-2017',
       '01-12-2017', '02-12-2017', '03-12-2017', '04-12-2017',
       '05-12-2017', '06-12-2017', '07-12-2017', '08-12-2017',
       '09-12-2017', '10-12-2017', '11-12-2017', '12-12-2017',
       '13-12-2017', '14-12-2017', '15-12-2017', '16-12-2017',
       '17-12-2017', '18-12-2017', '19-12-2017', '20-12-2017',
       '21-12-2017', '22-12-2017', '23-12-2017', '24-12-2017',
       '25-12-2017', '26-12-2017', '27-12-2017', '28-12-2017',
       '29-12-2017', '30-12-2017', '31-12-2017', '01-01-2018',
       '02-01-2018', '03-01-2018', '04-01-2018', '05-01-2018',
       '06-01-2018', '07-01-2018', '08-01-2018', '09-01-2018'],
      dtype='<U10')
```

```
data
```

```
['14-11-2017', '4005', 'Happy', '139', '8', 'Active'],
['15-11-2017', '4880', 'Happy', '164', '4', 'Active'],
['16-11-2017', '4136', 'Happy', '137', '5', 'Active'],
['17-11-2017', '705', 'Happy', '22', '6', 'Active'],
['18-11-2017', '570', 'Neutral', '17', '5', 'Active'],
['19-11-2017', '269', 'Happy', '9', '6', 'Active'],
['20-11-2017', '4275', 'Happy', '145', '5', 'Inactive'],
['21-11-2017', '5999', 'Happy', '192', '6', 'Inactive'],
['22-11-2017', '4421', 'Happy', '146', '5', 'Inactive'],
['23-11-2017', '6930', 'Happy', '234', '6', 'Inactive'],
['24-11-2017', '5195', 'Happy', '167', '5', 'Inactive'],
['25-11-2017', '546', 'Happy', '16', '6', 'Inactive'],
['26-11-2017', '493', 'Happy', '17', '7', 'Active'],
['27-11-2017', '995', 'Happy', '32', '6', 'Active'],
['28-11-2017', '1163', 'Neutral', '35', '7', 'Active'],
['29-11-2017', '6676', 'Sad', '220', '6', 'Active'],
['30-11-2017', '3608', 'Happy', '116', '5', 'Active'],
['01-12-2017', '774', 'Happy', '23', '6', 'Active'],
['02-12-2017', '1421', 'Happy', '44', '7', 'Active'],
['03-12-2017', '4064', 'Happy', '131', '8', 'Active'],
['04-12-2017', '2725', 'Happy', '86', '8', 'Active'],
['05-12-2017', '5934', 'Happy', '194', '7', 'Active'],
['06-12-2017', '1867', 'Happy', '60', '8', 'Active'],
['07-12-2017', '3721', 'Sad', '121', '5', 'Active'],
['08-12-2017', '2374', 'Neutral', '76', '4', 'Inactive'],
['09-12-2017', '2909', 'Neutral', '93', '3', 'Active'],
['10-12-2017', '1648', 'Sad', '53', '3', 'Active'],
['11-12-2017', '799', 'Sad', '25', '4', 'Inactive'],
['12-12-2017', '7102', 'Neutral', '227', '5', 'Active'],
['13-12-2017', '3941', 'Neutral', '125', '5', 'Active'],
['14-12-2017', '7422', 'Happy', '243', '5', 'Active'],
['15-12-2017', '437', 'Neutral', '14', '3', 'Active'],
['16-12-2017', '1231', 'Neutral', '39', '4', 'Active'],
['17-12-2017', '1696', 'Sad', '55', '4', 'Inactive'],
['18-12-2017', '4921', 'Neutral', '158', '5', 'Active'],
['19-12-2017', '221', 'Sad', '7', '5', 'Active'],
['20-12-2017', '6500', 'Neutral', '213', '5', 'Active'],
['21-12-2017', '3575', 'Neutral', '116', '5', 'Active'],
['22-12-2017', '4061', 'Sad', '129', '5', 'Inactive'],
['23-12-2017', '651', 'Sad', '21', '5', 'Inactive'],
['24-12-2017', '753', 'Sad', '28', '4', 'Inactive'],
['25-12-2017', '518', 'Sad', '16', '3', 'Inactive'],
['26-12-2017', '5537', 'Happy', '180', '4', 'Active'],
['27-12-2017', '4108', 'Neutral', '138', '5', 'Active'],
['28-12-2017', '5376', 'Happy', '176', '5', 'Active'],
['29-12-2017', '3066', 'Neutral', '99', '4', 'Active'],
['30-12-2017', '177', 'Sad', '5', '5', 'Inactive'],
['31-12-2017', '36', 'Sad', '1', '3', 'Inactive'],
['01-01-2018', '299', 'Sad', '10', '3', 'Inactive'],
['02-01-2018', '1447', 'Neutral', '47', '3', 'Inactive']
```

```
date,step_count,mood,calories_burned,hours_of_sleep,activity_status = data.T
```

```
array(['Happy', 'Neutral', 'Sad'], dtype='<U10')
```

```
step_count = np.array(step_count,dtype="int")
```

```
calories_burned = np.array(calories_burned, dtype ="int")
```

```
hours_of_sleep = np.array(hours_of_sleep,dtype = 'int')
```

```
count
```

```
array([40, 27, 29])
```

```
step_count.mean()
```

```
2935.9375
```

```
step_count.max()
```

```
7422
```

```
step_count.argmax()
```

```
69
```

```
step_count.argmin()
```

```
2
```

```
step_count.min()
```

```
25
```

```
date[step_count.argmin()]
```

```
'08-10-2017'
```

```
calories_burned[step_count.argmax()]
```

```
243
```

```
np.mean(step_count[mood=="Sad"])
```

```
2103.0689655172414
```

```
np.mean(step_count[mood=="Happy"])
```

```
3392.725
```

```
np.unique(mood[step_count>4000], return_counts=True)
```

```
(array(['Happy', 'Neutral', 'Sad'], dtype='<U10'), array([22,  9,  7]))
```

```
np.unique(mood[step_count<2000], return_counts=True)
```

```
(array(['Happy', 'Neutral', 'Sad'], dtype='<U10'), array([13,  8, 18]))
```

```
a = np.arange(9,0,-1).reshape(3,3)
a
```

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

```

```

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

```
np.sort(a,axis=1)
```

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

```
## Matrix Multiplication
```

```
a = np.arange(5)
b = np.ones(5)*2
a*b
```

```
array([0., 2., 4., 6., 8.])
```

```

```

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

```
a = np.arange(12).reshape(3,4)
a
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

```
b = np.arange(12).reshape(3,4)
b
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

```
a*b
```

```
array([[  0,   1,   4,   9],
       [ 16,  25,  36,  49],
       [ 64,  81, 100, 121]])
```

```
np.matmul(a,b)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-93-f6001c33e8b2> in <cell line: 1>()
----> 1 np.matmul(a,b)

ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->
(n?,m?) (size 3 is different from 4)
```

SEARCH STACK OVERFLOW

```
b = b.T
```

```
np.matmul(a,b)
```

```
array([[ 14,  38,  62],
       [ 38, 126, 214],
       [ 62, 214, 366]])
```

```
a @ b
```

```
array([[ 14,  38,  62],
       [ 38, 126, 214],
       [ 62, 214, 366]])
```

```
np.dot(a,b)
```

```
array([[ 14,  38,  62],
       [ 38, 126, 214],
       [ 62, 214, 366]])
```

```
x = np.arange(30).reshape(5,6)
y = np.arange(12).reshape(1,12)
```

```
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-104-6849a5f7ad6c> in <cell line: 1>()
----> 1 np.dot(x,y)

/usr/local/lib/python3.10/dist-packages/numpy/core/overrides.py in dot(*args, **kwargs)

ValueError: shapes (4,3) and (4,4) not aligned: 3 (dim 1) != 4 (dim 0)
```

SEARCH STACK OVERFLOW

```
## Vectorisation
```

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

```
a*2
```

```
array([ 2,  4,  6,  8, 10])
```

```
a = np.arange(1,11)
a
```

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

```
```

```
import math
```

```
1.6094379124341003
```

```
math.log(a)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-115-c35a2570e118> in <cell line: 1>()
----> 1 math.log(a)

TypeError: only size-1 arrays can be converted to Python scalars

SEARCH STACK OVERFLOW
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-119-2d02cb169a36> in <cell line: 1>()
----> 1 map(int[a,b,c,d])

NameError: name 'c' is not defined

SEARCH STACK OVERFLOW
```

```
a = np.arange(48).reshape(2,6,4)
a
```

```
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]]])
```

```
a.size
```

```
24
```

```
a.shape
```

```
(2, 3, 4)
```

```
a.ndim
```

```
3
```

```
import matplotlib.pyplot as plt
```

```
!gdown 17tYTDPBU5hpby9t0kGd7w_-zBsbY7sEd
```

```
Downloading...
From: https://drive.google.com/uc?id=17tYTDPBU5hpby9t0kGd7w_-zBsbY7sEd
To: /content/fruits.png
100% 4.71M/4.71M [00:00<00:00, 27.3MB/s]
```

```
img = plt.imread("fruits.png")
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x792e98930d30>
```

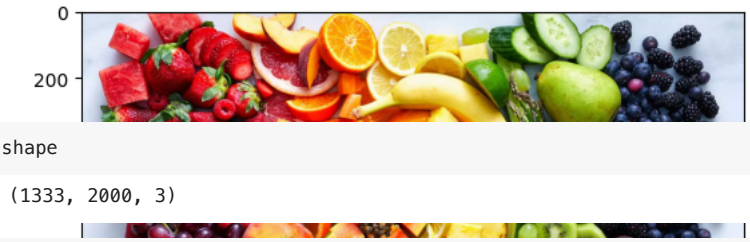

```
img
```

```
array([[[0.8784314 , 0.9137255 , 0.972549  ],
        [0.8784314 , 0.9137255 , 0.972549  ],
        [0.8784314 , 0.9137255 , 0.972549  ],
        ...,
        [0.8       , 0.85490197, 0.9098039 ],
        [0.8       , 0.85490197, 0.9098039 ],
        [0.8       , 0.85490197, 0.9098039 ]],

       [[0.8784314 , 0.9137255 , 0.972549  ],
        [0.8784314 , 0.9137255 , 0.972549  ],
        [0.8784314 , 0.9137255 , 0.972549  ],
        ...,
        [0.8       , 0.85490197, 0.9098039 ],
        [0.8       , 0.85490197, 0.9098039 ],
        [0.8       , 0.85490197, 0.9098039 ]],

       [[0.8784314 , 0.9137255 , 0.972549  ],
        [0.8784314 , 0.9137255 , 0.972549  ],
        [0.8784314 , 0.9137255 , 0.972549  ],
        ...,
        [0.8039216 , 0.85882354, 0.9137255 ],
        [0.8039216 , 0.85882354, 0.9137255 ],
        [0.8039216 , 0.85882354, 0.9137255 ]],

       ...,

       [[0.74509805, 0.79607844, 0.87058824],
        [0.74509805, 0.79607844, 0.87058824],
        [0.74509805, 0.79607844, 0.87058824],
        ...,
        [0.83137256, 0.8627451 , 0.9411765 ],
        [0.83137256, 0.8627451 , 0.9411765 ],
        [0.83137256, 0.8627451 , 0.9411765 ]],

       [[0.74509805, 0.79607844, 0.87058824],
        [0.74509805, 0.79607844, 0.87058824],
        [0.74509805, 0.79607844, 0.87058824],
        ...,
        [0.83137256, 0.8627451 , 0.9411765 ],
        [0.83137256, 0.8627451 , 0.9411765 ],
        [0.83137256, 0.8627451 , 0.9411765 ]],

       [[0.74509805, 0.79607844, 0.87058824],
        [0.74509805, 0.79607844, 0.87058824],
        [0.74509805, 0.79607844, 0.87058824],
        ...,
        [0.83137256, 0.8627451 , 0.9411765 ],
        [0.83137256, 0.8627451 , 0.9411765 ],
        [0.83137256, 0.8627451 , 0.9411765 ]]], dtype=float32)
```

```
img_r = img.copy()
```

```
plt.imshow(img_r)
```

```
<matplotlib.image.AxesImage at 0x792e989da050>
```

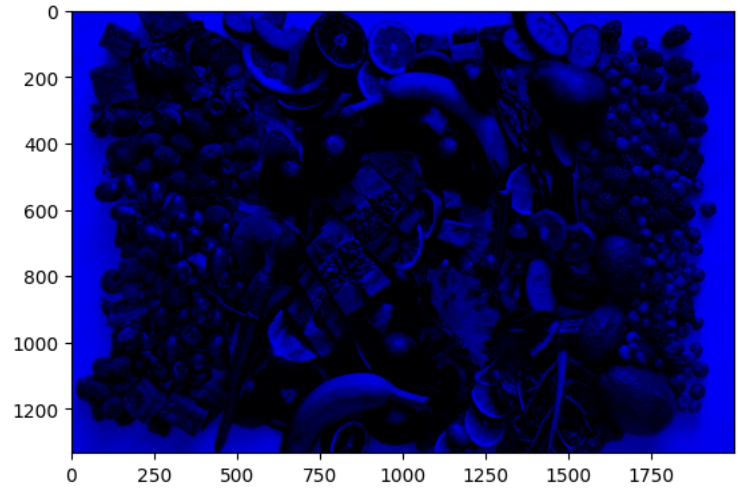



```
img.shape
```

```
(1333, 2000, 3)
```

```
img_r[:,:,(0,1,2)]=1
plt.imshow(img_r)
```

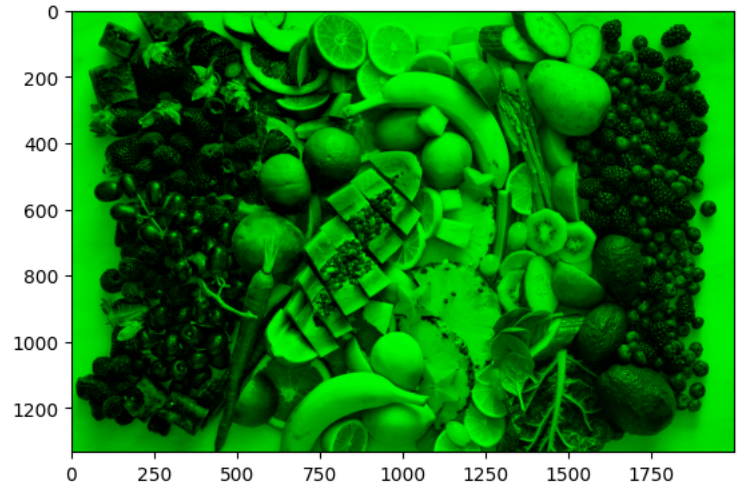
```
<matplotlib.image.AxesImage at 0x792e986c8e50>
```




```
img_g = img.copy()
```

```
img_g[:,:,(0,2)]=0
plt.imshow(img_g)
```

```
<matplotlib.image.AxesImage at 0x792e985933a0>
```




```
img_r = img.copy()
```

```
img_r[:,:,(1,2)]=0
plt.imshow(img_r)
```

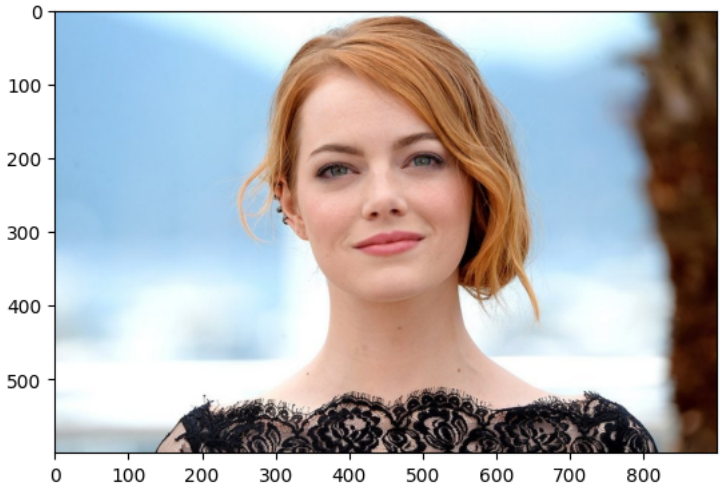
```
<matplotlib.image.AxesImage at 0x792e987a0130>
```


```
!gdown 1o-8yqdTM7cfz_mAaNCi2nH0urFu7pcqI
```

```
Downloading...
From: https://drive.google.com/uc?id=1o-8yqdTM7cfz_mAaNCi2nH0urFu7pcqI
To: /content/emma_stone.jpeg
100% 80.3k/80.3k [00:00<00:00, 3.74MB/s]
```

```
img_emma = plt.imread("emma_stone.jpeg")
plt.imshow(img_emma)
```

```
<matplotlib.image.AxesImage at 0x792e9859d360>
```




```
img_emma.shape
```

```
(600, 900, 3)
```

```
help(np.transpose)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-37-f97903b1fbd8> in <cell line: 1>()
----> 1 help(np.transpose)

NameError: name 'np' is not defined
```

SEARCH STACK OVERFLOW

```
import numpy as np
a = np.array([1,2,3,4,5])
b = a+1
b
```

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

```
np.shares_memory(a,b)
```

```
False
```

```
# Splitting
```

```
a = np.arange(10)
a
```

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

```
b = np.split(a,5)
b
```

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

```
a.shape
```

```
(9,)
```

```
b = np.split(a,(1,2,3,5,8))
b
```

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

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-27-a8244287b6af> in <cell line: 1>()
----> 1 b.shape

AttributeError: 'list' object has no attribute 'shape'

SEARCH STACK OVERFLOW
```

```
x = np.arange(1,17).reshape(4,4)
```

```
x
```

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
```

```
np.split(x,2,axis=1)
```

```
[array([[ 1,  2],
        [ 5,  6],
        [ 9, 10],
        [13, 14]]),
 array([[ 3,  4],
        [ 7,  8],
        [11, 12],
        [15, 16]])]
```

```
np.hsplit(x,2)
```

```
[array([[ 1,  2],
        [ 5,  6],
        [ 9, 10],
        [13, 14]]),
 array([[ 3,  4],
        [ 7,  8],
        [11, 12],
        [15, 16]])]
```

```
np.vsplit(x,2)
```

```
[array([[1, 2, 3, 4],
        [5, 6, 7, 8]]),
 array([[ 9, 10, 11, 12],
        [13, 14, 15, 16]])]
```

```
## Stacking
```

```
a = np.arange(5)
```

```
np.vstack((a,a,a))
```

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

```
np.hstack((a,a,a))
```

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

```
z = np.arange(9).reshape(3,3)
```

```
np.vstack((z,z,z))
```

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

```
np.hstack((z,z,z))
```

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

```
np.concatenate((z,z,z), axis =1)
```

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

```
np.concatenate((z,z,z), axis =0)
```

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

```
# Broadcasting
```

```
a = np.arange(0,40,10)
a
```

```
array([ 0, 10, 20, 30])
```

```
np.vstack((a,a,a))
```

```
array([[ 0, 10, 20, 30],
       [ 0, 10, 20, 30],
       [ 0, 10, 20, 30]])
```

```
np.tile(a,(3,2))
```

```
array([[ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30]])
```

```

```

```
array([[ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30]])
```

```

```

```
array([[ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30]])
```

```

```

```
# Rule 1 : If two array differ in the number of dimensions, the shape of one with fewer dimensions is
# padded with ones on its leading( Left Side).


# Rule 2 : If the shape of two arrays doesnt match in any dimensions, the array with shape equal to
# 1 is stretched to match the other shape.
```

```
import numpy as np

a = np.arange(6)
```

```
a.shape
```

```
(6,)
```

```
a
```

```
array([0, 1, 2, 3, 4, 5])
```

```
a = np.arange(6)
np.expand_dims(a,axis=0).shape
```

```
(1, 6)
```

```
b = np.arange(6)
np.expand_dims(b,axis=1).shape
b
```

```
array([0, 1, 2, 3, 4, 5])
```

```
a = np.arange(6)
a[np.newaxis,:]
```

```
array([[0, 1, 2, 3, 4, 5]])
```

```
a = np.arange(6)
a[:,np.newaxis]
```

```
array([[0],
       [1],
       [2],
       [3],
       [4],
       [5]])
```

```
b = np.arange(6).reshape(2,3)
```

```
np.expand_dims(b,axis=0).shape
```

```
(1, 2, 3)
```

```
c = np.arange(6).reshape(2,3)
np.expand_dims(b,axis=2).shape
```

```
(2, 3, 1)
```

```
a = np.arange(5)
np.expand_dims(a,axis = 0).shape
```

```
(1, 5)
```

```
a = np.arange(12).reshape(12,1,1)
a
```

```
array([[[ 0]],

       [[ 1]],

       [[ 2]],

       [[ 3]],

       [[ 4]],

       [[ 5]],
```

```
        [[ 6]],

        [[ 7]],

        [[ 8]],

        [[ 9]],

        [[10]],

        [[11]]])
```