

Colab Link: <https://colab.research.google.com/drive/1s3bTSyenZ32rJw2CfapfTSFi8VzmYkKg?usp=sharing>

+ Code

+ Text

---

How likely is it that you would recommend [company X] to a friend or colleague?

0	1	2	3	4	5	6	7	8	9	10
Not at all likely										Extremely likely

---

NPS - Net Promoter Score = %Promoters - %Dectractors

```
a = [1, 2, 3, 4, 5]
```

```
a
```

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

```
type(a)
```

```
list
```

```
# take square of each element
[i**2 for i in a]
```

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

```
import numpy as np
```

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

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

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

```
numpy.ndarray
```

```
a_np ** 2
```

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

```
# cleaner syntax (for doing elementwise operations)
# Numpy makes Numerical Computing (Element wise operations) much much faster than r
```

```
l = range(1000000)
```

```
%timeit [i**2 for i in l]
```

```
321 ms ± 9.41 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

```
l_np = np.array(range(1000000))
```

```
%timeit l_np ** 2
```

```
2.02 ms ± 138 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

```
l_np * l_np # student doubt
```

```
array([          0,          1,          4, ..., 999994000009,
        999996000004, 999998000001])
```

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

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

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

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

```
[1, 2, 3] + 2
```

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-19-0736e8f5bfd5> in <module>
----> 1 [1, 2, 3] + 2
```

```
TypeError: can only concatenate list (not "int") to list
```

SEARCH STACK OVERFLOW

```
# [1, 2, 3] - 1D array
```

```
# [[1, 2, 3], [4, 5, 6], [7, 8, 9]] - 2D array
```

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

```
type(arr)
```

```
numpy.ndarray
```

```
arr.ndim
```

```
1
```

```
arr2 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

```
arr2.ndim
```

2

`arr.shape``(3,)``arr2.shape``(3, 3)`# Shaik's question - `np.array([[1,2,3],[4,3,5,6],[7,8,9]]).shape`# `np.array([[1,2,3],[4,3,5,6],[7,8,9]]).shape`

u meant to do this, you must specify 'dtype=object' when creating the ndarray.

`list(range(1, 10, 2))``[1, 3, 5, 7, 9]``np.arange(1, 10, 2)``array([1, 3, 5, 7, 9])``np.arange(1, 5, 0.5)``array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])``range(1, 10, 0.5)`

```

-----
TypeError                                Traceback (most recent call last)
<ipython-input-31-9b6226ca7e45> in <module>
----> 1 range(1, 10, 0.5)

```

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

# linspace - post-read

`np.array([1, 2, 3, 4.5])``array([1. , 2. , 3. , 4.5])``np.array([1, 2.0, "A"])`

```
array(['1', '2.0', 'A'], dtype='<U32')
```

```
np.array([1, 2.0, "A"]).dtype
```

```
dtype('<U32')
```

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

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

```
m1 = np.arange(12)
```

```
m1
```

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

```
m1[1]
```

```
1
```

```
m1[12]
```

```
-----
IndexError                                Traceback (most recent call last)
<ipython-input-39-0abd94d7097d> in <module>
----> 1 m1[12]
```

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

SEARCH STACK OVERFLOW

```
m1[-1]
```

```
11
```

```
m1 = np.array([100,200,300,400,500,600]) # new stuff
```

```
m1
```

```
array([100, 200, 300, 400, 500, 600])
```

```
m1[[1, 4]] # list of indexes
```

```
array([200, 500])
```

```
m1[[1, 4, 5]]
```

```
array([200, 500, 600])
```

```
m1[[1, 4, 1]] # indexes can be repeated
```

```
array([200, 500, 200])
```

```
# Slicing
```

```
m1 = np.arange(12)
m1
```

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

```
m1[:5]
```

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

```
m1[-5:-1]
```

```
array([ 7,  8,  9, 10])
```

```
m1[-5:-1:-1]
```

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

```
# indexing, slicing, assignment (=), elementwise operation (hint)
```

```
a = np.arange(6)
a
```

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

```
a[4:] = 10
```

```
a
```

```
array([ 0,  1,  2,  3, 10, 10])
```

```
# Fancy Indexing
```

```
m1 = np.arange(12)
m1
```

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

```
m1 < 6 # boolean mask
```

```
array([ True,  True,  True,  True,  True,  True, False, False, False,
        False, False, False])
```

```
m1[m1 < 6]
```

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

```
m1[m1 % 2 == 0]
```

```
array([ 0,  2,  4,  6,  8, 10])
```

```
# multiple conditions for indexing
```

```
# filter the numbers which are either divisible or divisible by - or
```

```
m1[(m1 % 2 == 0) | (m1 % 5 == 0)] # its imp to sep the conditions with brackets
```

```
array([ 0,  2,  4,  5,  6,  8, 10])
```

```
# | for or, & for and
```

```
a = np.arange(6)
```

```
mask = (a%2 == 0) # even numbers
```

```
a[mask] = -1 # fancy indexing and assignment
```

```
print(a) # will show -1 at the place of even numbers
```

```
[-1  1 -1  3 -1  5]
```

```
np.arange(3) + 3 # vectorisation
```

```
# just for explanation: [0, 1, 2] + [3, 3, 3] --> 3 is vectorised
```

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

```
a = np.arange(4)
```

```
a
```

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

```
a * a
```

```
array([0, 1, 4, 9])
```

```
b = np.arange(5, 9)
```

```
b
```

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

```
a * b
```

```
array([ 0,  6, 14, 24])
```

```
b = np.arange(5, 10)
```

```
a * b
```

```
-----
ValueError                                Traceback (most recent call last)
<ipython-input-75-c9d4da4572fe> in <module>
      1 b = np.arange(5, 10)
----> 2 a * b
```

**ValueError:** operands could not be broadcast together with shapes (4,) (5,)

# aggregate/ universal functions (ufuncs)

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

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

```
np.sum([1, 2, 3, 4])
```

```
10
```

```
a
```

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

```
np.mean(a)
```

```
1.5
```

```
np.min(a)
```

```
0
```

```
np.max(a)
```

```
3
```

```
# https://drive.google.com/uc?id=1c0ClC8SrPwJq5rrkyMKyPn80nyHcFikK
```

```
!gdown 1c0ClC8SrPwJq5rrkyMKyPn80nyHcFikK
```

```
Downloading...
```

```
From: https://drive.google.com/uc?id=1c0ClC8SrPwJq5rrkyMKyPn80nyHcFikK
```

```
To: /content/survey.txt
```

```
100% 2.55k/2.55k [00:00<00:00, 2.83MB/s]
```

```
!ls
```

```
sample_data  survey.txt
```

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

```
score[:5]

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

score.ndim

1

score.shape

(1167,)

len(score)

1167

np.min(score)

1

np.max(score)

10

# %promoters
# %detractors

# lets bin our data: numerical data --> categories

detractors = len(score[score <= 6])

detractors

332

total = len(score)

percent_detractors = (detractors/total)*100

promoters = score[score >= 9].shape[0]

score[score >= 9].shape[0] # student doubt

609

percent_promoters = promoters/total*100
```



```
percent_promoters
```

```
52.185089974293064
```

```
nps = np.round(percent_promoters - percent_detractors, 2)
nps
```

```
23.74
```

```
score[score <= 6] = "D" # numpy stores homo data and hence only int
```

```
-----
ValueError                                Traceback (most recent call last)
<ipython-input-113-c303c8505bc7> in <module>
----> 1 score[score <= 6] = "D"

ValueError: invalid literal for int() with base 10: 'D'
```

SEARCH STACK OVERFLOW

```
cat_arr = np.empty(dtype="str", shape=score.shape)
```

```
cat_arr[score <= 6] = "D"
```

```
cat_arr
```

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

```

```
cat_arr[score >= 9] = "P"
```

```
cat_arr
```

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

```

```
# homework - write code for replacing '' with "N"
```

[Colab paid products](#) - [Cancel contracts here](#)

✓ 0s completed at 23:17

