Colab Link: https://colab.research.google.com/drive/1xZ0Jr-V6IEf_31dVWL2lC6dex3VP8dt1?usp=sharing

```
import numpy as np
```

```
# Operations on Numpy Arrays using one array
```

```
m1 = np.arange(12).reshape(3, 4)
m1
```

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

```
m1 + 2
```

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

```
m1 * 2
```

```
array([[ 0,  2,  4,  6],
       [ 8, 10, 12, 14],
       [16, 18, 20, 22]])
```

```
sing two numpy arrays
```

Saved successfully!

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

```
a + b
```

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

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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
```

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

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

```
a = np.array([1,2,3,5,8])
b = np.array([0,3,4,2,1])
c = a + b
c = c*a
print (c[2])
```

```
21
```

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

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

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

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

Saved successfully!

```
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]])
```

```
A @ B
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-18-455d622f3b50> in <module>()
```

```
A @ B.reshape(4, 3)
```

```
array([[ 42,  48,  54],
       [114, 136, 158],
       [186, 224, 262]])
```

```
A.reshape(4, 3) @ B
```

```
array([[ 20,  23,  26,  29],
       [ 56,  68,  80,  92],
       [ 92, 113, 134, 155],
       [128, 158, 188, 218]])
```

```
np.matmul(A.reshape(4, 3), B)
```

```
array([[ 20,  23,  26,  29],
       [ 56,  68,  80,  92],
       [ 92, 113, 134, 155],
       [128, 158, 188, 218]])
```

```
# + ==> np.sum()
# @ ==> np.matmul()
```

```
np.dot(A.reshape(4, 3), B)
```

```
array([[ 20,  23,  26,  29],
       [128, 158, 188, 218]])
```

Saved successfully!

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

```
12
```

```
# Universal Functions (ufuncs) - mathematical funcs providing element wise ope
```

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

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

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

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

```
# sin, cos, tan, exp, log
```

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

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

```
np.sum(a)
```

```
66
```

```
a
```

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

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

```
array([12, 15, 18, 21])
```

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

```
array([ 6, 22, 38])
```

Saved successfully!

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

```
5.5
```

```
a
```

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

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

```
array([4., 5., 6., 7.])
```

```
np.min(a), np.max(a)
```

```
(0, 11)
```

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

```
array([ 3,  7, 11])
```

```
a[:, 0].sum()
```

```
12
```

```
np.sum(a, axis=0)[0]
```

```
12
```

```
a
```

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

```
np.sum(a, axis=None) # for agrregate functions, if axis=None, then it will do
```

```
66
```

```
# Logical Functions
```

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



```
True
```

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

```
True
```

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

np.sum(a <= b) == len(a)
```

```
False
```

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

```
False
```

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

```
c = np.array([6, 4, 4, 5])
# a <= b and also b <= c met always or not
```

```
np.all((a <= b) & (b <= c))
```

```
    True
```

```
# Sorting
```

```
a = np.array([2,30,41,7,17,52])
```

```
np.sort(a)
```

```
    array([ 2,  7, 17, 30, 41, 52])
```

```
# argsort
```

```
np.argsort(a)
```

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

```
m = np.array([[23,4,43],
              [12,89,3],
              [69,420,0]])
```



```
    array([[ 23,   4,  43],
           [ 12,  89,   3],
           [ 69, 420,   0]])
```

```
np.sort(m, axis=-1)
```

```
    array([[  4,  23,  43],
           [  3,  12,  89],
           [  0,  69, 420]])
```

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

```
    array([[  4,  23,  43],
           [  3,  12,  89],
           [  0,  69, 420]])
```

```
# if the axis is not provided, it sorts accross the last dimension
```

```
np.sort(m)
```

```
array([[  4,  23,  43],
       [  3,  12,  89],
       [  0,  69, 420]])
```

```
# m[1,2] and m[1][2] - https://numpy.org/devdocs/user/basics.indexing.html
```

```
# Fitness Data Analysis
```

```
# https://drive.google.com/file/d/1kXqcJo4YzmwF1G2BPoA17CI49TZVHANF/view?usp=s
```

```
!gdown 1kXqcJo4YzmwF1G2BPoA17CI49TZVHANF
```

```
Downloading...
From: https://drive.google.com/uc?id=1kXqcJo4YzmwF1G2BPoA17CI49TZVHANF
To: /content/fitness.txt
100% 3.14k/3.14k [00:00<00:00, 5.09MB/s]
```

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

```
type(data)
```

```
numpy.ndarray
```

```
data.ndim
```

```
2
```

Saved successfully!

```
(96, 7)
```

```
# records - 96
# features - 7
```

There are 96 records and each record has 7 features. These features are:

- Date
- Step count
- Mood
- Calories Burned
- Hours of sleep
- activity status
- weight

```
data[:5,:]
```

```
array([['06-10-2017', '5464', '200', '181', '5', '0', '66'],
       ['07-10-2017', '6041', '100', '197', '8', '0', '66'],
       ['08-10-2017', '25', '100', '0', '5', '0', '66'],
       ['09-10-2017', '5461', '100', '174', '4', '0', '66'],
       ['10-10-2017', '6915', '200', '223', '5', '500', '66']],
      dtype='<U10')
```

```
data[0]
```

```
array(['06-10-2017', '5464', '200', '181', '5', '0', '66'], dtype='<U10')
```

```
data[:,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')
```

Saved successfully!

```
date, step_count = data.T[:2]
```

```
date
```

```
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')
```

```
step_count
```

```
array(['5464', '6041', '25', '5461', '6915', '4545', '4340', '1230', '61',
       '1258', '3148', '4687', '4732', '3519', '1580', '2822', '181',
       '3158', '4383', '3881', '4037', '202', '292', '330', '2209',
       '4550', '4435', '4779', '1831', '2255', '539', '5464', '6041',
       '4068', '4683', '4033', '6314', '614', '3149', '4005', '4880',
       '4136', '705', '570', '269', '4275', '5999', '4421', '6930',
       '5195', '546', '493', '995', '1163', '6676', '3608', '774', '1421',
       '4064', '2725', '5934', '1867', '3721', '2374', '2909', '1648',
       '799', '7102', '3941', '7422', '437', '1231', '1696', '4921',
       '221', '6500', '3575', '4061', '651', '753', '518', '5537', '4108',
       '5376', '3066', '177', '36', '299', '1447', '2599', '702', '133',
       '153', '500', '2127', '2203'], dtype='<U10')
```

```
date, step_count, mood, calories_burned, hours_of_sleep, activity_status, weig
```

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

Saved successfully!

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

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

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

```
weight = np.array(weight, dtype = 'float')
weight.dtype
```

```
dtype('float64')
```

```
mood
```

```
array(['200', '100', '100', '100', '200', '100', '100', '100', '100',
       '100', '100', '100', '300', '100', '100', '100', '100', '200',
       '200', '200', '200', '200', '200', '300', '200', '300', '300',
       '300', '300', '300', '300', '300', '200', '300', '300', '300',
       '300', '300', '300', '300', '300', '300', '300', '200', '300',
```

```
       '300', '300', '300', '300', '300', '300', '300', '300', '200',
       '100', '300', '300', '300', '300', '300', '300', '300', '100',
       '200', '200', '100', '100', '200', '200', '300', '200', '200',
       '100', '200', '100', '200', '200', '100', '100', '100', '100',
       '300', '200', '300', '200', '100', '100', '100', '200', '200',
       '100', '100', '300', '200', '200', '300'], dtype='<U10')
```

```
np.unique(mood)
```

```
array(['100', '200', '300'], dtype='<U10')
```

```
# strings = 300-->Happy, 200-->Neutral, 100-->Sad
```

```
mood[mood == "300"] = "Happy"
```

```
mood
```

```
array(['200', '100', '100', '100', '200', '100', '100', '100', '100',
       '100', '100', '100', 'Happy', '100', '100', '100', '100', '200',
       '200', '200', '200', '200', '200', 'Happy', '200', 'Happy',
       'Happy', 'Happy', 'Happy', 'Happy', 'Happy', 'Happy', '200',
       'Happy', 'Happy', 'Happy', 'Happy', 'Happy', 'Happy', 'Happy',
       'Happy', 'Happy', 'Happy', '200', 'Happy', 'Happy', 'Happy',
       'Happy', 'Happy', 'Happy', 'Happy', 'Happy', 'Happy', '200', '100',
       'Happy', 'Happy', 'Happy', 'Happy', 'Happy', 'Happy', 'Happy',
       '100', '200', '200', '100', '100', '200', '200', 'Happy', '200',
       '200', '100', '200', '100', '200', '200', '100', '100', '100',
       '100', 'Happy', '200', 'Happy', '200', '100', '100', '100', '200',
       '200', '100', '100', 'Happy', '200', '200', 'Happy'], dtype='<U10')
```

Saved successfully!

```
mood[mood == '100'] = 'Sad'
```

```
mood
```

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

```
np.unique(activity_status)
```

```
array(['0', '500'], dtype='<U10')
```

```
activity_status[activity_status == '500'] = 'Active'
activity_status[activity_status == '0'] = 'Inactive'
```

```
activity_status
```

```
array(['Inactive', 'Inactive', 'Inactive', 'Inactive', 'Active',
       'Inactive', 'Inactive', 'Inactive', 'Inactive', 'Inactive',
       'Inactive', 'Inactive', 'Active', 'Inactive', 'Inactive',
       'Inactive', 'Inactive', 'Inactive', 'Inactive', 'Inactive',
       'Inactive', 'Inactive', 'Inactive', 'Inactive', 'Inactive',
       'Active', 'Inactive', 'Inactive', 'Inactive', 'Inactive', 'Active',
       'Inactive', 'Inactive', 'Inactive', 'Inactive', 'Inactive',
       'Active', 'Active', 'Active', 'Active', 'Active', 'Active',
       'Active', 'Active', 'Active', 'Inactive', 'Inactive', 'Inactive',
       'Inactive', 'Inactive', 'Inactive', 'Active', 'Active', 'Active',
       'Active', 'Active', 'Active', 'Active', 'Active', 'Active',
       'Active', 'Active', 'Active', 'Inactive', 'Active', 'Active',
       'Inactive', 'Active', 'Active', 'Active', 'Active', 'Active',
       'Inactive', 'Active', 'Active', 'Active', 'Active', 'Inactive',
       'Inactive', 'Inactive', 'Inactive', 'Active', 'Active', 'Active',
       'Active', 'Inactive', 'Inactive', 'Inactive', 'Inactive',
       'Inactive', 'Inactive', 'Inactive', 'Inactive', 'Active',
       'Inactive', 'Active'], dtype='<U10')
```

```
# Analysis
```

```
step_count.mean()
```

Saved successfully!

```
# on which date the step count the highest
```

```
date[np.argsort(step_count)[-1]]
```

```
'14-12-2017'
```

```
np.argmax(step_count)
```

```
69
```

```
np.argmin(step_count)
```

```
2
```

```
date[np.argmax(step_count)]
```

```
'14-12-2017'
```

```
# what is the most frequent mood
```

```
mood
```

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

```
np.sum(mood == "Sad")
```

```
29
```

```
np.sum(mood == "Happy")
```

```
40
```

```
np.sum(mood == "Neutral")
```

Saved successfully!

```
np.unique(mood)
```

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

```
np.unique(mood, return_counts=True)
```

```
(array(['Happy', 'Neutral', 'Sad'], dtype='<U10'), array([40, 27, 29]))
```

```
## activity/step-count <--> mood
```

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

```
2103.0689655172414
```

```
step_count[mood == "Neutral"].mean()
```

```
3153.777777777778
```

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

```
3392.725
```

Vectorisation

```
a = np.array([1, 2, 3])
b = np.array([2, 3, 4])
a + b # sum is vectorised addition, Numpy - vectorised library
```

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

```
from math import log
```

```
log(2) # we are trying to vectorise a function
```

```
0.6931471805599453
```

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

```
array([0.        , 0.69314718, 1.09861229, 1.38629436, 1.60943791])
```

```
# Lets take a function which numpy lib doesnt have
```

```
from math import exp
```



```
0.36787944117144233
```

```
def sigmoid(x):
    return 1/(1 + exp(-x))
```

```
sigmoid(0)
```

```
0.5
```

```
vectorised_sigmoid = np.vectorize(sigmoid)
```

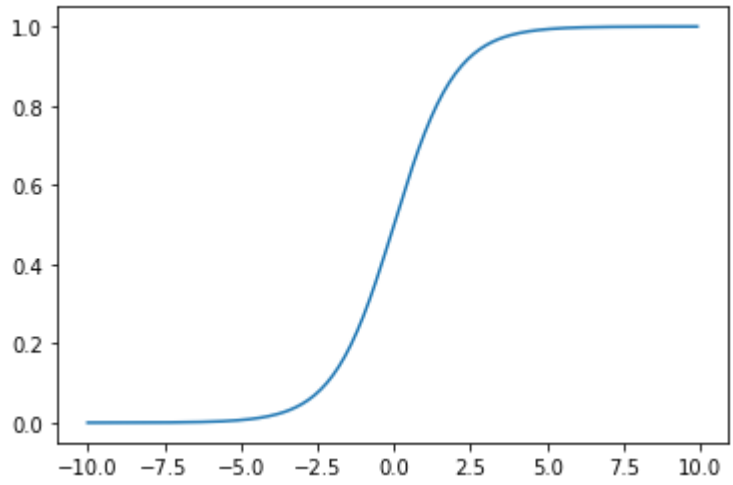
```
vectorised_sigmoid([-2, -1, 0, 1, 1])
```

```
array([0.11920292, 0.26894142, 0.5       , 0.73105858, 0.73105858])
```

```
x = np.arange(-10, 10, 0.1)
```

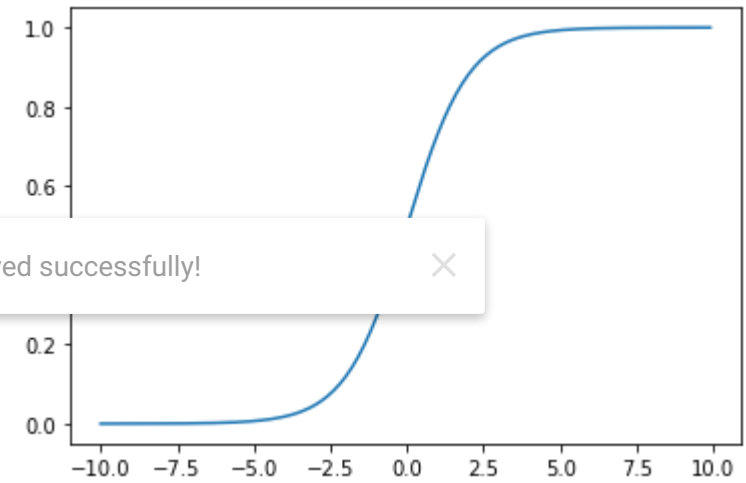
```
y = vectorised_sigmoid(x)

import matplotlib.pyplot as plt
plt.plot(x, y)
```

```
[<matplotlib.lines.Line2D at 0x7f51b276ccd0>]
```




```
x = np.arange(-10, 10, 0.1)
y = 1/(1 + np.exp(-x))
plt.plot(x, y)
```

```
[<matplotlib.lines.Line2D at 0x7f51b229e0d0>]
```



✓ 0s completed at 23:37

Saved successfully!