In [1]:

```
!pip install numpy
```

```
Requirement already satisfied: numpy in /Users/anantm/opt/anaconda3/li
b/python3.8/site-packages (1.22.2)
```

In [2]:

```
import numpy as np
```

In [3]:

```
list1 = [1, 2, 3]
print(list1)
```

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

In [4]:

```
list1 * 2
```

Out[4]:

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

In [5]:

```
# [2, 4, 6] - element-wise operations, LA, ML, Deep Learning
```

In [6]:

```
[element*2 for element in list1]
```

Out[6]:

```
[2, 4, 6]
```

In [8]:

```
arr1 = np.array(list1)
arr1
```

Out[8]:

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

In [9]:

```
arr1 * 2
```

Out[9]:

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

In [10]:

```
# vectorised libraries, vectorised operations
```

In [12]:

```
list(range(1, 5))
```

Out[12]:

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

In [16]:

```
np.arange(1, 5)
```

Out[16]:

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

In [18]:

```
list(range(1, 5, 1.5))
```

```
-----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
 last)
<ipython-input-18-1f3ed239dff3> in <module>
----> 1 list(range(1, 5, 1.5))

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

In [24]:

```
np.arange(1, 5, 1) # start, end, step
```

Out[24]:

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

In [20]:

```
# start, end, count - equal parts
```

In [21]:

```
np.linspace(1, 5, 10) # end is included
```

Out[21]:

```
array([1.        , 1.44444444, 1.88888889, 2.33333333, 2.77777778,
       3.22222222, 3.66666667, 4.11111111, 4.55555556, 5.        ])
```

In [25]:

```
type(arr1)
```

Out[25]:

```
numpy.ndarray
```

In [26]:

```
arr4 = np.array([1, 2, 3, 4])
```

In [27]:

```
arr5 = np.array([1, 2, 3, 4.3])
```

In [28]:

```
arr5
```

Out[28]:

```
array([1. , 2. , 3. , 4.3])
```

In [29]:

```
np.array([1, 2, 3, 4.3, "Anant"])
```

Out[29]:

```
array(['1', '2', '3', '4.3', 'Anant'], dtype='<U32')
```

In [30]:

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

Out[30]:

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

In [31]:

```
np.array([1, 2, 3, 4], dtype="int8")
```

Out[31]:

```
array([1, 2, 3, 4], dtype=int8)
```

In [33]:

```
100**10
```

Out[33]:

```
100000000000000000000
```

In [34]:

```
np.array([0, 10, 100])**10
```

Out[34]:

```
array([                  0,          10000000000, 7766279631452241920])
```

In [35]:

```
# 2-D arrays
```

In [39]:

```
m1  = np.array([[1, 2, 3], [3, 4, 5]])
m1
```

Out[39]:

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

In [40]:

```
m1.shape
```

Out[40]:

```
(2, 3)
```

In [42]:

```
m1.ndim
```

Out[42]:

```
2
```

In [43]:

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

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

In [44]:

```
a.ndim
```

Out[44]:

```
2
```

In [45]:

```
a.shape
```

Out[45]:

```
(1, 5)
```

In [46]:

```
# high dim array
```

In [48]:

```
m2 = np.arange(1, 13)
```

In [49]:

```
m2
```

Out[49]:

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

In [52]:

```
m3 = m2.reshape(3, 4)
```

In [53]:

```
m2
```

Out[53]:

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

In [54]:

```
m3
```

Out[54]:

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

In [55]:

```
m3.shape, m3.ndim
```

Out[55]:

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

In [56]:

```
m2.shape
```

Out[56]:

```
(12,)
```

In [59]:

```
m2.ndim
```

Out[59]:

```
1
```

In [60]:

```
m4 = m2.reshape(1, 12)
m4
```

Out[60]:

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

In [61]:

```
m4.shape
```

Out[61]:

```
(1, 12)
```

In [62]:

```
m4.ndim
```

Out[62]:

```
2
```

In [64]:

```
m5 = m2.reshape(12, 1)
```

In [65]:

```
m5.ndim
```

Out[65]:

```
2
```

In [66]:

```
m5.shape
```

Out[66]:

```
(12, 1)
```

In [ ]:

```
m2.reshape(12, 1)
```

In [67]:

```
a = np.array([[1,2,3],[0,1,4]])
print (a.ndim)
```

```
2
```

In [68]:

```
np.zeros(3)
```

Out[68]:

```
array([0., 0., 0.])
```

In [69]:

```
np.zeros((3, 4))
```

Out[69]:

```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

In [70]:

```
np.ones(3)
```

Out[70]:

```
array([1., 1., 1.])
```

In [71]:

```
np.ones((2, 3))
```

Out[71]:

```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

In [ ]:

```
#(3, 4), 99
```

In [73]:

```
np.ones((3, 4))**99
```

Out[73]:

```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

In [74]:

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

Out[74]:

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

In [77]:

```
np.identity(3) # eye
```

Out[77]:

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

In [78]:

```python
np.eye(3)
```

Out[78]:

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

In [79]:

```python
# indexing and slicing
```

In [80]:

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

In [81]:

```python
m1
```

Out[81]:

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

In [84]:

```python
m1[0][1] # dont use it, people don't it
```

Out[84]:

```
1
```

In [86]:

```python
m1[0, 1]
```

Out[86]:

```
1
```

In [87]:

```python
m1[0, 1:3]
```

Out[87]:

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

In [88]:

```python
# [[5, 6],
# [9, 10]]
```

In [90]:

```
m1[1:, 1:3]
```

Out[90]:

```
array([[ 5,  6],
       [ 9, 10]])
```

In [91]:

```
# [[2, 3],
# [6, 7],
# [10, 11]]
```

In [92]:

```
m1[:, 2:]
```

Out[92]:

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

In [93]:

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

# [[1, 3],
#  [5, 7],
#  [9,11]]
```

In [94]:

```
m1[:, 1::2]
```

Out[94]:

```
array([[ 1,  3],
       [ 5,  7],
       [ 9, 11]])
```

In [95]:

```
m1[:, (1,3)]
```

Out[95]:

```
array([[ 1,  3],
       [ 5,  7],
       [ 9, 11]])
```

In [96]:

```
# splitting
```

In [98]:

```
x = np.arange(9)
x
```

Out[98]:

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

In [101]:

```
np.split(x, 3)
```

Out[101]:

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

In [102]:

```
np.split(x, [3, 5, 6])
```

Out[102]:

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

In [103]:

```
x = np.arange(16.0).reshape(4, 4)
x
```

Out[103]:

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

In [104]:

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

Out[104]:

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

In [106]:

```
np.hsplit(x, [3])
```

Out[106]:

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

In [107]:

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

Out[107]:

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

In [108]:

```
np.vsplit(x, [3])
```

Out[108]:

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

In [111]:

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

Out[111]:

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

In [112]:

```
m1 * 3
```

Out[112]:

```
array([[ 0,  3,  6,  9],
       [12, 15, 18, 21],
       [24, 27, 30, 33]])
```

In [ ]:

```
[], map()
```

In [114]:

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

```
21
```

In [115]:

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

Out[115]:

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

In [116]:

```
m1 < 6
```

Out[116]:

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

In [117]:

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

Out[117]:

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

In [118]:

```
# filter(lambda x: x < 6, [...])
```

In [119]:

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

Out[119]:

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

In [121]:

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

Out[121]:

```
(6,)
```

In [122]:

```
# Vectorisation
```

In [125]:

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

Out[125]:

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

In [126]:

```
A * 2
```

Out[126]:

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

In [129]:

```
import math
math.log(10)
```

Out[129]:

```
2.302585092994046
```

In [131]:

```
math.log(np.array([10, 11]))
```

```
----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
 last)
<ipython-input-131-46a331ec46fb> in <module>
----> 1 math.log(np.array([10, 11]))

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

In [133]:

```
vec_log = np.vectorize(math.log)
```

In [135]:

```
vec_log(np.array([10, 11]))
```

Out[135]:

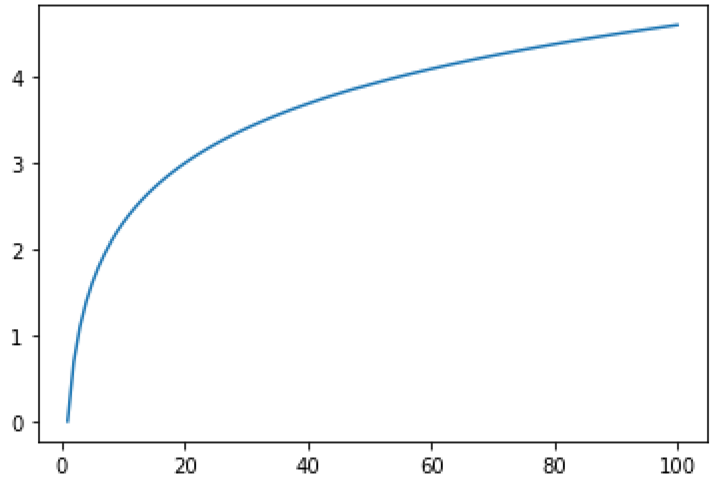
```
array([2.30258509, 2.39789527])
```

In [137]:

```
import math
import matplotlib.pyplot as plt

x = np.arange(1, 101)
y = np.vectorize(math.log)(x)

plt.plot(x, y)
plt.show()
```



In [138]:

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

Out[138]:

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

In [139]:

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

Out[139]:

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

In [140]:

```
A * B # not matrix multiplication
```

Out[140]:

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

In [141]:

```
B = B.reshape(4, 3)
```

In [142]:

```
B
```

Out[142]:

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

In [143]:

```
A
```

Out[143]:

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

In [144]:

```
np.matmul(A, B)
```

Out[144]:

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

In [145]:

```
np.matmul(B, A)
```

Out[145]:

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

In [146]:

```
A @ B
```

Out[146]:

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

In [147]:

```
B @ A
```

Out[147]:

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

In [148]:

```
np.dot(A, B)
```

Out[148]:

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

In [149]:

```
d1 = np.array([1, 2, 3])
d2 = np.array([2, 4, 6])
np.dot(d1, d2)
```

Out[149]:

```
28
```

In [150]:

```
A
```

Out[150]:

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

In [151]:

```
A.T
```

Out[151]:

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

In [152]:

```
d1.T
```

Out[152]:

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

In [153]:

```
d3 = d1.reshape(1, 3)
d3
```

Out[153]:

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

In [154]:

```
d3.T
```

Out[154]:

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

In [156]:

```
A.flatten()
```

Out[156]:

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

In [157]:

```
A.ravel()
```

Out[157]:

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

In [159]:

```
A.reshape(12)
```

Out[159]:

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

In [160]:

```
A.reshape(-1)
```

Out[160]:

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

In [161]:

```
A.size
```

Out[161]:

```
12
```

In [162]:

```
A
```

Out[162]:

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

In [164]:

```
A.reshape(4, -1)
```

Out[164]:

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

In [165]:

```
A.reshape(-1, 3)
```

Out[165]:

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

In [166]:

```
A.reshape(6, -1)
```

Out[166]:

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

In [167]:

```
# 3 - 2
# 4 - 3
# 2 - 1
# 1 - 1
```

In [168]:

```
# stacking
```

In [172]:

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

In [173]:

```
data
```

Out[173]:

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

In [174]:

```
np.vstack((data, data, data))
```

Out[174]:

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

In [175]:

```
np.hstack((data, data, data))
```

Out[175]:

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

In [179]:

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

In [180]:

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

Out[180]:

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

In [181]:

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

Out[181]:

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

In [182]:

```
np.hstack((data.T, data.T, data.T))
```

Out[182]:

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

In [183]:

```
data.T
```

Out[183]:

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

In [184]:

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

In [188]:

```
a.sort()
```

In [189]:

```
a
```

Out[189]:

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

In [191]:

```
A.sort(axis=0)
```

In [192]:

```
A
```

Out[192]:

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

In [193]:

```
A.sort(axis=1)
```

In [194]:

```
A
```

Out[194]:

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

In [195]:

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

In [196]:

```
a
```

Out[196]:

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

In [197]:

```
a.argsort()
```

Out[197]:

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

In [203]:

```
A = np.array([[2, -1, 7],
    [10, 1, 11],
    [9, 8, 11]])
```

In [201]:

```
A.sort(axis=0)
```

In [202]:

```
A
```

Out[202]:

```
array([[ 2, -1,  7],
       [ 9,  1, 11],
       [10,  8, 11]])
```

In [204]:

```
A.sort(axis=1)
```

In [205]:

```
A
```

Out[205]:

```
array([[-1,  2,  7],
       [ 1, 10, 11],
       [ 8,  9, 11]])
```

In [209]:

```
n1 = np.arange(1, 13).reshape(3,4)
log = np.vectorize(math.log)(n1)
```

In [210]:

```
log
```

Out[210]:

```
array([[0.        , 0.69314718, 1.09861229, 1.38629436],
       [1.60943791, 1.79175947, 1.94591015, 2.07944154],
       [2.19722458, 2.30258509, 2.39789527, 2.48490665]])
```