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

DataTypes & Attributes

in an np.function(), press shift+tab to get function introduction

In [2]:
```python
# Build an array Numpy's main datatype is ndarray
a1 = np.array([1,2,3])
a1
```

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

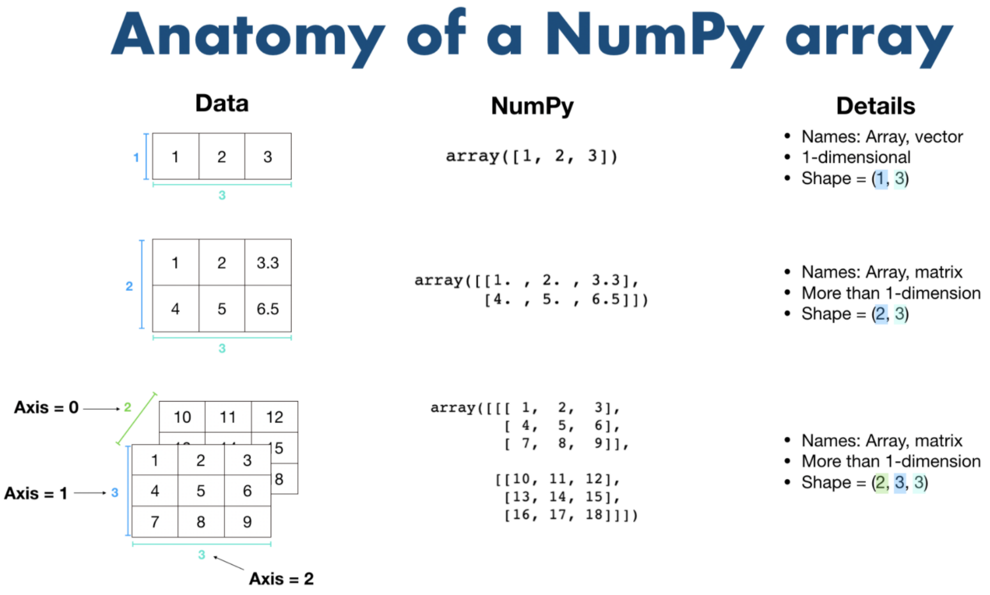
In [3]:
```python
type(a1)
```

Out[3]:
```
numpy.ndarray
```

In [4]:
```python
# need 2 sets of bracket for 2+ dimentional arrays[[],[],[]]
a2 = np.array([[1,2,3],[4,5,6]])
a3 = np.array([[[1, 2, 3],
                [4, 5, 6]],
               [[7, 8, 9],
                [10, 11, 12]],
               [[13, 14, 15],
                [16, 17, 18]]])
```

In [5]:
```python
from IPython.display import Image
Image(filename ='NumpyArray.png')
```

Out[5]:



Anatomy of a NumPy array

Data | NumPy | Details

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

- Names: Array, vector
- 1-dimensional
- Shape = (1, 3)

```
array([[1. , 2. , 3.3],
       [4. , 5. , 6.5]])
```

- Names: Array, matrix
- More than 1-dimension
- Shape = (2, 3)

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

       [[10, 11, 12],
        [13, 14, 15],
        [16, 17, 18]]])
```

- Names: Array, matrix
- More than 1-dimension
- Shape = (2, 3, 3)

In [6]:

```
# dimentions
a3.shape
```

Out[6]:

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

In [7]:

```
# number of dimentions
a1.ndim, a2.ndim, a3.ndim
```

Out[7]:

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

In [8]:

```
# number of elements
a1.size, a2.size, a3.size
```

Out[8]:

```
(3, 6, 18)
```

In [9]:

```
# conver an Numpy array into dataframe
import pandas as pd
df=pd.DataFrame(a2)
df
```

Out[9]:

	0	**1**	**2**
0	1	2	3
1	4	5	6

Create Numpy Arrays

In [10]:

```
# general way - see more above
samplearray=np.array([1,2,3])
samplearray
```

Out[10]:

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

In [11]:

```
# np.ones()function
ones =np.ones((2,3))
ones
```

Out[11]:

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

In [12]:

```
zeros =np.zeros((2,3))
zeros
```

Out[12]:

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

In [13]:

```
#arange (start, stop, step)
range_array=np.arange(0,10,2)
range_array
```

Out[13]:

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

In [14]:

```
#np.random.randint(low, high, size, dtype*)
randomarray=np.random.randint(0,10,size=(3,5))
randomarray
```

Out[14]:

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

In [15]:

```
#np.random.random () Return random floats in the half-open interval [0.0, 1.0)
np.random.random((3,5))
```

Out[15]:

```
array([[0.27386658, 0.25506438, 0.24653402, 0.23846343, 0.55785478],
       [0.6660079 , 0.25957566, 0.62872697, 0.41072581, 0.13172257],
       [0.01698753, 0.07332979, 0.42903516, 0.38273177, 0.03792224]])
```

In [16]:

```
#np.random.rand(3,5) Random values in a given shape.
randomarray= np.random.rand(3,5)
```

In [17]:

```
# Psuedo-random Allow other user who received this can generate the same set of random numbers
np.random.seed(seed=0)
randomarray2=np.random.randint(0,10,size=(3,5))
randomarray2
```

Out[17]:

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

In [18]:

```
# set up the same seed, then run it again, get the same random numbers
np.random.seed(seed=0)
randomarray2=np.random.randint(0,10,size=(3,5))
randomarray2
```

Out[18]:

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

In [19]:

```
np.random.seed(seed=0)
randomarray3= np.random.random((3,5))
randomarray3
```

Out[19]:

```
array([[0.5488135 , 0.71518937, 0.60276338, 0.54488318, 0.4236548 ],
       [0.64589411, 0.43758721, 0.891773  , 0.96366276, 0.38344152],
       [0.79172504, 0.52889492, 0.56804456, 0.92559664, 0.07103606]])
```

View Arrays and matrices

```
In [20]: #find unique values
         np.unique(ones)

Out[20]: array([1.])
```

```
In [21]: #get a row [n]
         a2

Out[21]: array([[1, 2, 3],
                [4, 5, 6]])
```

```
In [22]: a2[0]

Out[22]: array([1, 2, 3])
```

```
In [23]: # slicing
         a3

Out[23]: array([[[ 1,  2,  3],
                 [ 4,  5,  6]],

                [[ 7,  8,  9],
                 [10, 11, 12]],

                [[13, 14, 15],
                 [16, 17, 18]]])
```

```
In [24]: a3[:2]

Out[24]: array([[[ 1,  2,  3],
                 [ 4,  5,  6]],

                [[ 7,  8,  9],
                 [10, 11, 12]]])
```

```
In [25]: a3[:2,:2,:2]

Out[25]: array([[[ 1,  2],
                 [ 4,  5]],

                [[ 7,  8],
                 [10, 11]]])
```

In [26]:
```
# read from right to left: 5 is 5 elements in each row; 4 is 4 columns;
#3 is 3 matrices # 2 is 2 sets of 3 matrices
a4=np.random.randint(10,size=(2,3,4,5))
a4
```

Out[26]:
```
array([[[[9, 4, 3, 0, 3],
         [5, 0, 2, 3, 8],
         [1, 3, 3, 3, 7],
         [0, 1, 9, 9, 0]],

        [[4, 7, 3, 2, 7],
         [2, 0, 0, 4, 5],
         [5, 6, 8, 4, 1],
         [4, 9, 8, 1, 1]],

        [[7, 9, 9, 3, 6],
         [7, 2, 0, 3, 5],
         [9, 4, 4, 6, 4],
         [4, 3, 4, 4, 8]]],


       [[[4, 3, 7, 5, 5],
         [0, 1, 5, 9, 3],
         [0, 5, 0, 1, 2],
         [4, 2, 0, 3, 2]],

        [[0, 7, 5, 9, 0],
         [2, 7, 2, 9, 2],
         [3, 3, 2, 3, 4],
         [1, 2, 9, 1, 4]],

        [[6, 8, 2, 3, 0],
         [0, 6, 0, 6, 3],
         [3, 8, 8, 8, 2],
         [3, 2, 0, 8, 8]]]])
```

In [27]:

```
#get the first 4 number of all the matrices
#slice all the dimention 1 (2), dimention2 (3), dimention 3(4),
# and keey the first 4 elements in demention 4 (first 4 out of 5)
a4[:,:,:,:4]
```

Out[27]:

```
array([[[[9, 4, 3, 0],
         [5, 0, 2, 3],
         [1, 3, 3, 3],
         [0, 1, 9, 9]],

        [[4, 7, 3, 2],
         [2, 0, 0, 4],
         [5, 6, 8, 4],
         [4, 9, 8, 1]],

        [[7, 9, 9, 3],
         [7, 2, 0, 3],
         [9, 4, 4, 6],
         [4, 3, 4, 4]]],


       [[[4, 3, 7, 5],
         [0, 1, 5, 9],
         [0, 5, 0, 1],
         [4, 2, 0, 3]],

        [[0, 7, 5, 9],
         [2, 7, 2, 9],
         [3, 3, 2, 3],
         [1, 2, 9, 1]],

        [[6, 8, 2, 3],
         [0, 6, 0, 6],
         [3, 8, 8, 8],
         [3, 2, 0, 8]]]]])
```

Manipulate and Compare Arrays

Arithmatic

In [28]:

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

Out[28]:

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

In [29]:

```
a1
```

Out[29]:

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

In [30]:
```
#can do math directly on arrays +, -
a1+ones
np.add(a1,ones)
```

Out[30]:
```
array([2., 3., 4.])
```

In [31]:
```
a2
```

Out[31]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [32]:
```
# this is not matrix multiplication; a1 row 1* a2 row 1; a1 row1*a2 row
 2
# the small array broadcasts to larger array
a1*a2
```

Out[32]:
```
array([[ 1,  4,  9],
       [ 4, 10, 18]])
```

In [33]:
```
a3
```

Out[33]:
```
array([[[ 1,  2,  3],
        [ 4,  5,  6]],

       [[ 7,  8,  9],
        [10, 11, 12]],

       [[13, 14, 15],
        [16, 17, 18]]])
```

In [34]:
```
a2*a3
```

Out[34]:
```
array([[[  1,   4,   9],
        [ 16,  25,  36]],

       [[  7,  16,  27],
        [ 40,  55,  72]],

       [[ 13,  28,  45],
        [ 64,  85, 108]]])
```

In [35]:
```
a2/a1
```

Out[35]:
```
array([[1. , 1. , 1. ],
       [4. , 2.5, 2. ]])
```

In [36]:
```
#Floor division removes decimals
a2//a1
```

Out[36]:
```
array([[1, 1, 1],
       [4, 2, 2]])
```

In [37]:
```
# ** means power; either one below works
a2**2
np.square(a2)
```

Out[37]:
```
array([[ 1,  4,  9],
       [16, 25, 36]])
```

In [38]:
```
# find modular
a2%2
```

Out[38]:
```
array([[1, 0, 1],
       [0, 1, 0]])
```

In [39]:
```
np.exp(a1)
```

Out[39]:
```
array([ 2.71828183,  7.3890561 , 20.08553692])
```

In [40]:
```
np.log(a1)
```

Out[40]:
```
array([0.        , 0.69314718, 1.09861229])
```

Aggregation

In [41]:
```
#aggregation = performing the same operation on a number of things
listy_list=[1,2,3]
type(listy_list)
```

Out[41]:
```
list
```

In [42]:
```
sum(listy_list)
```

Out[42]:
```
6
```

In [43]:
```
np.sum(listy_list)
```

Out[43]:
```
6
```

In [44]:
```
massive_array=np.random.random(10000)
massive_array[:10]
```

Out[44]:
```
array([0.57615733, 0.59204193, 0.57225191, 0.22308163, 0.95274901,
       0.44712538, 0.84640867, 0.69947928, 0.29743695, 0.81379782])
```

In [45]:
```
%timeit sum(massive_array) #Python's sum method
%timeit np.sum(massive_array) # Numpy's sum method -- way faster than Py
thon (use Numpy as much as possible)
```

```
1.5 ms ± 35.4 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
6.61 µs ± 333 ns per loop (mean ± std. dev. of 7 runs, 100000 loops eac
h)
```

```
In [46]: a2
```

```
Out[46]: array([[1, 2, 3],
                [4, 5, 6]])
```

```
In [47]: np.mean(a2)
```

```
Out[47]: 3.5
```

```
In [48]: np.max(a2)
```

```
Out[48]: 6
```

```
In [49]: #standard deviation: how spread out
         np.std(a2)
```

```
Out[49]: 1.707825127659933
```

```
In [50]: #variance = standard deviation^2
         np.var(a2)
```

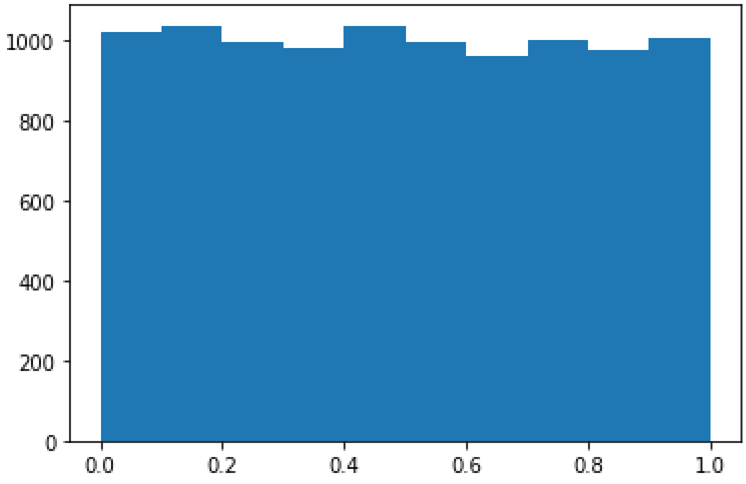
```
Out[50]: 2.9166666666666665
```

```
In [51]: np.sqrt(np.var(a2))
```

```
Out[51]: 1.707825127659933
```

```
In [52]: %matplotlib inline
         import matplotlib.pyplot as plt
         plt.hist(massive_array)    # 能够画出分布
```

```
Out[52]: (array([1022., 1036.,  994.,  979., 1035.,  995.,  959.,  998.,  976.,
                 1006.]),
          array([7.24496385e-05, 1.00063000e-01, 2.00053550e-01, 3.00044100e-01,
                 4.00034650e-01, 5.00025201e-01, 6.00015751e-01, 7.00006301e-01,
                 7.99996851e-01, 8.99987402e-01, 9.99977952e-01]),
          <a list of 10 Patch objects>)
```



Reshape and Transpose

```
In [53]: a2
```

```
Out[53]: array([[1, 2, 3],
                [4, 5, 6]])
```

```
In [54]: a2.shape
```

```
Out[54]: (2, 3)
```

```
In [55]: a3
```

```
Out[55]: array([[[ 1,  2,  3],
                 [ 4,  5,  6]],

                [[ 7,  8,  9],
                 [10, 11, 12]],

                [[13, 14, 15],
                 [16, 17, 18]]])
```

```
In [56]: a3.shape
```

```
Out[56]: (3, 2, 3)
```

```
In [57]: a2*a3
```

```
Out[57]: array([[[  1,   4,   9],
                 [ 16,  25,  36]],

                [[  7,  16,  27],
                 [ 40,  55,  72]],

                [[ 13,  28,  45],
                 [ 64,  85, 108]]])
```

```
In [61]: a2_reshape=a2.reshape (2,3,1)
```

In [62]:
```
#注意矩阵的乘法
a2_reshape*a3
```

```
---------------------------------------------------------------------------
----
ValueError                                Traceback (most recent call l
ast)
<ipython-input-62-94e05e6e354b> in <module>
      1 #注意矩阵的乘法
----> 2 a2_reshape*a3

ValueError: operands could not be broadcast together with shapes (2,3,
1) (3,2,3)
```

In [63]:
```
#Transpose: 行变列，列变行
a2.T
```

Out[63]:
```
array([[1, 4],
       [2, 5],
       [3, 6]])
```

In [64]:
```
a3.T.shape
```

Out[64]:
```
(3, 2, 3)
```

In [65]:
```
#从这开始讲Dot Product: 即两个矩阵的乘积
np.random.seed(0)
mat1=np.random.randint(10,size=(5,3))
mat2=np.random.randint(10,size=(5,3))
mat1
```

Out[65]:
```
array([[5, 0, 3],
       [3, 7, 9],
       [3, 5, 2],
       [4, 7, 6],
       [8, 8, 1]])
```

In [66]:
```
mat2
```

Out[66]:
```
array([[6, 7, 7],
       [8, 1, 5],
       [9, 8, 9],
       [4, 3, 0],
       [3, 5, 0]])
```

In [67]:
```
mat1.shape, mat2.shape
```

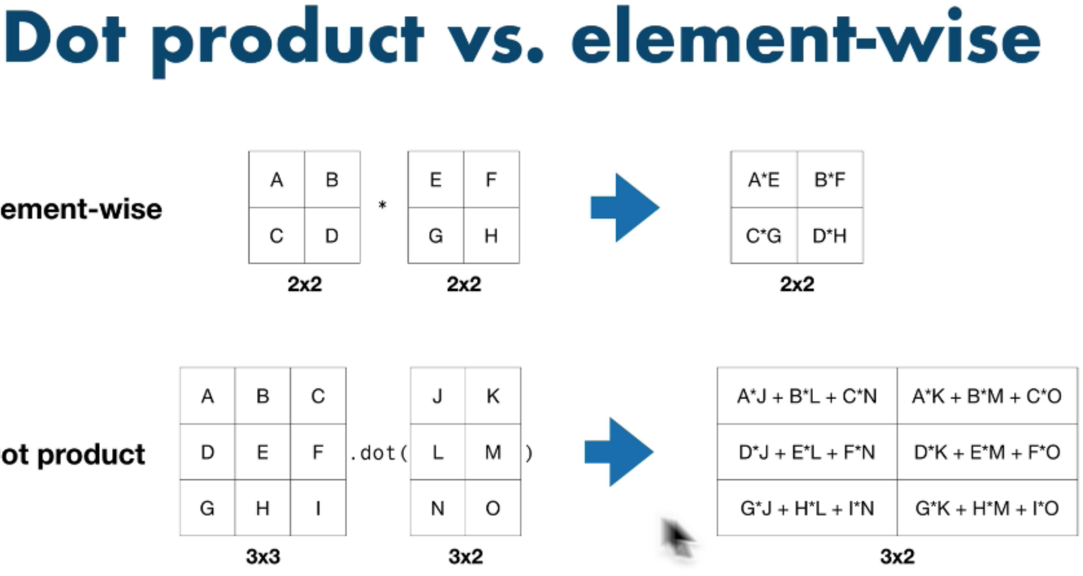
Out[67]:
```
((5, 3), (5, 3))
```

In [68]:
```
#这个给的是对应element乘积，没什么用,element-wise
mat1*mat2
```

Out[68]:
```
array([[30,  0, 21],
       [24,  7, 45],
       [27, 40, 18],
       [16, 21,  0],
       [24, 40,  0]])
```

In [69]:
```
Image(filename ='DotProduct.png')
```

Out[69]:



Dot product vs. element-wise

Element-wise

A B / C D * E F / G H → A*E B*F / C*G D*H

2x2 2x2 2x2

Dot product

A B C / D E F / G H I .dot(J K / L M / N O) → A*J + B*L + C*N A*K + B*M + C*O / D*J + E*L + F*N D*K + E*M + F*O / G*J + H*L + I*N G*K + H*M + I*O

3x3 3x2 3x2

In [70]:
```
# Dot Product: m*n product n*k
np.dot(mat1.T,mat2) #3*3
```

Out[70]:
```
array([[121, 114,  77],
       [153, 108,  80],
       [135,  69,  84]])
```

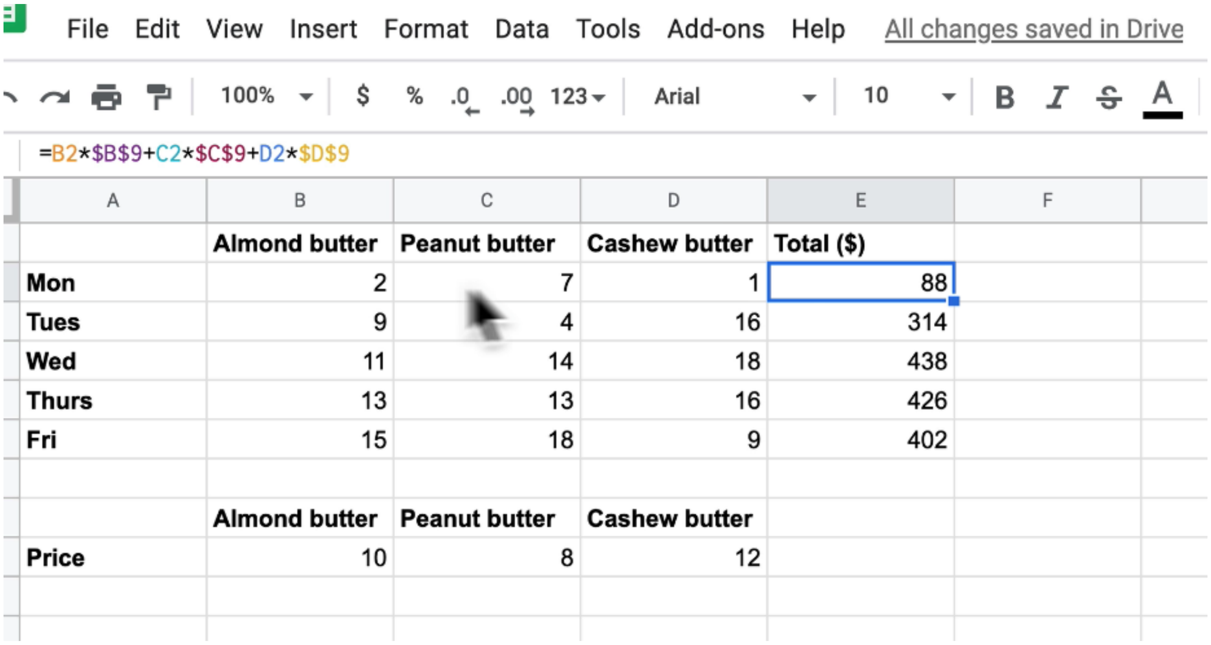
In [71]:
```
mat3=np.dot(mat1,mat2.T) #5*5
mat3
```

Out[71]:
```
array([[ 51,  55,  72,  20,  15],
       [130,  76, 164,  33,  44],
       [ 67,  39,  85,  27,  34],
       [115,  69, 146,  37,  47],
       [111,  77, 145,  56,  64]])
```

In [72]:

```
#练习matrix dot product multiplication
Image(filename ='ArrayMultiplyExcel.png')
```

Out[72]:



=B2*B9+C2*C9+D2*D9

	A	B	C	D	E	F
		Almond butter	**Peanut butter**	**Cashew butter**	**Total ($)**	
	Mon	2	7	1	88	
	Tues	9	4	16	314	
	Wed	11	14	18	438	
	Thurs	13	13	16	426	
	Fri	15	18	9	402	
		Almond butter	**Peanut butter**	**Cashew butter**		
	Price	10	8	12		

In [73]:

```
#create sales matrix , 5*3 matrix
sales_amounts=np.random.randint(20,size=(5,3))
weekly_sales=pd.DataFrame(sales_amounts,index=['Mon','Tues','Wed','Thurs','Fri'],columns=['Almond_butter','Peanut','Cashew_butter'])
weekly_sales
```

Out[73]:

	Almond_butter	Peanut	Cashew_butter
Mon	18	3	17
Tues	19	19	19
Wed	14	7	0
Thurs	1	9	0
Fri	10	3	11

In [74]:

```
#create prices array
prices=np.array([10,8,12])
prices.shape
```

Out[74]:

```
(3,)
```

In [75]:

```
#create prices matrix -- be careful, prices needs reshape to a 1*3 matrix
butter_prices=pd.DataFrame(prices.reshape(1,3),index=["Price"],columns=['Almond_butter','Peanut','Cashew_butter'])
butter_prices
```

Out[75]:

	Almond_butter	Peanut	Cashew_butter
Price	10	8	12

In [76]:

```
total_sales1 = prices.dot(sales_amounts.T)
total_sales1
```

Out[76]:

```
array([408, 570, 196,  82, 256])
```

In [77]:

```
#点积可以在dataframe里进行
total_sales2 = butter_prices.dot(weekly_sales.T)
total_sales2
```

Out[77]:

	Mon	Tues	Wed	Thurs	Fri
Price	408	570	196	82	256

In [78]:

```
#如何partition两个dataframe: shape error
weekly_sales["Total($)"]=total_sales2
weekly_sales
```

```
---------------------------------------------------------------------------
----
KeyError                                  Traceback (most recent call l
ast)
~/opt/miniconda3/lib/python3.7/site-packages/pandas/core/indexes/base.p
y in get_loc(self, key, method, tolerance)
   2645             try:
-> 2646                 return self._engine.get_loc(key)
   2647             except KeyError:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObj
ectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObj
ectHashTable.get_item()

KeyError: 'Total($)'

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call l
ast)
~/opt/miniconda3/lib/python3.7/site-packages/pandas/core/internals/mana
gers.py in set(self, item, value)
   1070         try:
-> 1071             loc = self.items.get_loc(item)
   1072         except KeyError:

~/opt/miniconda3/lib/python3.7/site-packages/pandas/core/indexes/base.p
y in get_loc(self, key, method, tolerance)
   2647             except KeyError:
-> 2648                 return self._engine.get_loc(self._maybe_cast_in
dexer(key))
   2649         indexer = self.get_indexer([key], method=method, tolera
nce=tolerance)

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObj
ectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObj
ectHashTable.get_item()

KeyError: 'Total($)'

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call l
ast)
<ipython-input-78-bd3fefd0393b> in <module>
```

```
      1 #如何partition两个dataframe: shape error
----> 2 weekly_sales["Total($)"]=total_sales2
      3 weekly_sales

~/opt/miniconda3/lib/python3.7/site-packages/pandas/core/frame.py in __
setitem__(self, key, value)
   2936         else:
   2937             # set column
-> 2938             self._set_item(key, value)
   2939 
   2940     def _setitem_slice(self, key, value):

~/opt/miniconda3/lib/python3.7/site-packages/pandas/core/frame.py in _s
et_item(self, key, value)
   2999         self._ensure_valid_index(value)
   3000         value = self._sanitize_column(key, value)
-> 3001         NDFrame._set_item(self, key, value)
   3002 
   3003         # check if we are modifying a copy

~/opt/miniconda3/lib/python3.7/site-packages/pandas/core/generic.py in
_set_item(self, key, value)
   3622 
   3623     def _set_item(self, key, value) -> None:
-> 3624         self._data.set(key, value)
   3625         self._clear_item_cache()
   3626 

~/opt/miniconda3/lib/python3.7/site-packages/pandas/core/internals/mana
gers.py in set(self, item, value)
   1072         except KeyError:
   1073             # This item wasn't present, just insert at end
-> 1074             self.insert(len(self.items), item, value)
   1075             return
   1076 

~/opt/miniconda3/lib/python3.7/site-packages/pandas/core/internals/mana
gers.py in insert(self, loc, item, value, allow_duplicates)
   1179         new_axis = self.items.insert(loc, item)
   1180 
-> 1181         block = make_block(values=value, ndim=self.ndim, placem
ent=slice(loc, loc + 1))
   1182 
   1183         for blkno, count in _fast_count_smallints(self._blknos[
loc:]):

~/opt/miniconda3/lib/python3.7/site-packages/pandas/core/internals/bloc
ks.py in make_block(values, placement, klass, ndim, dtype)
   3045         values = DatetimeArray._simple_new(values, dtype=dtype)
   3046 
-> 3047     return klass(values, ndim=ndim, placement=placement)
   3048 
   3049 

~/opt/miniconda3/lib/python3.7/site-packages/pandas/core/internals/bloc
ks.py in __init__(self, values, placement, ndim)
    123         if self._validate_ndim and self.ndim and len(self.mgr_l
ocs) != len(self.values):
    124                 raise ValueError(
--> 125                     f"Wrong number of items passed {len(self.value
s)}, "
    126                     f"placement implies {len(self.mgr_locs)}"
    127                 )

ValueError: Wrong number of items passed 5, placement implies 1
```

In [79]:

```
# transpose can fix this
weekly_sales["Total($)"]=total_sales2.T
weekly_sales
```

Out[79]:

	Almond_butter	Peanut	Cashew_butter	Total($)
Mon	18	3	17	408
Tues	19	19	19	570
Wed	14	7	0	196
Thurs	1	9	0	82
Fri	10	3	11	256

Comparison Operators

In [80]:

```
a1
```

Out[80]:

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

In [81]:

```
a2
```

Out[81]:

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

In [82]:

```
# whether each element in a1 is greater than a2
a1>a2
bool_array=a1>=a2
bool_array
```

Out[82]:

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

In [83]:

```
a1<5
```

Out[83]:

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

In [84]:

```
a1==a1
```

Out[84]:

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

In [85]:
```
a1==a2
```

Out[85]:
```
array([[ True,  True,  True],
       [False, False, False]])
```

Sort an Array

In [86]:
```
random_array = np.random.randint(10,size=(3,5))
random_array
```

Out[86]:
```
array([[2, 7, 2, 0, 0],
       [4, 5, 5, 6, 8],
       [4, 1, 4, 9, 8]])
```

In [87]:
```
#Sort in each row
np.sort(random_array)
```

Out[87]:
```
array([[0, 0, 2, 2, 7],
       [4, 5, 5, 6, 8],
       [1, 4, 4, 8, 9]])
```

In [88]:
```
#which index has the smallest to largest element
np.argsort(random_array)
```

Out[88]:
```
array([[3, 4, 0, 2, 1],
       [0, 1, 2, 3, 4],
       [1, 0, 2, 4, 3]])
```

In [89]:
```
# index of the smallest value, argmax if looking for the largest
np.argmin(random_array)
```

Out[89]:
```
3
```

In [90]:
```
# index of the smallest value,position starting from 0 for each column
np.argmin(random_array,axis=0)
```

Out[90]:
```
array([0, 2, 0, 0, 0])
```

In [91]:
```
#index of the smalles element ,position starting from 0 for each row
np.argmin(random_array,axis=1)
```

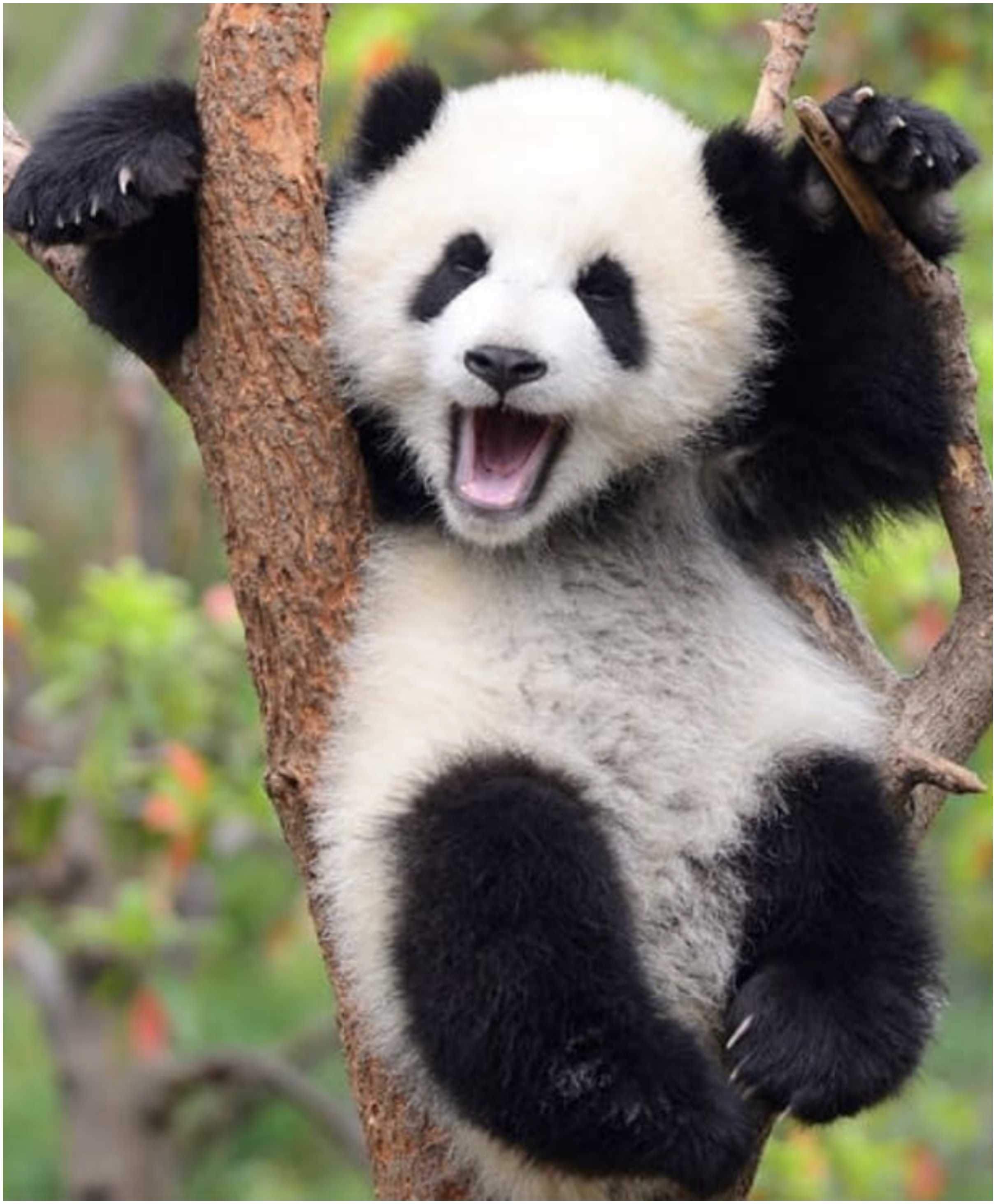
Out[91]:
```
array([3, 0, 1])
```

Turn Image into Numy Arrays (pixel value)

In [92]:

```
Image(filename ='Panda.png')
```

Out[92]:



In [93]:

```
from matplotlib.image import imread
panda=imread("Panda.png")
print(type(panda))
```

```
<class 'numpy.ndarray'>
```

In [94]:
```
panda
```

Out[94]:
```
array([[[0.6156863 , 0.63529414, 0.47843137, 1.        ],
        [0.6156863 , 0.63529414, 0.47843137, 1.        ],
        [0.6156863 , 0.63529414, 0.4745098 , 1.        ],
        ...,
        [0.4392157 , 0.5176471 , 0.3254902 , 1.        ],
        [0.44313726, 0.52156866, 0.32156864, 1.        ],
        [0.44705883, 0.52156866, 0.3254902 , 1.        ]],

       [[0.61960787, 0.63529414, 0.4862745 , 1.        ],
        [0.61960787, 0.63529414, 0.48235294, 1.        ],
        [0.61960787, 0.6392157 , 0.48235294, 1.        ],
        ...,
        [0.44313726, 0.5254902 , 0.33333334, 1.        ],
        [0.44705883, 0.5254902 , 0.32941177, 1.        ],
        [0.44705883, 0.5254902 , 0.32941177, 1.        ]],

       [[0.61960787, 0.6392157 , 0.49411765, 1.        ],
        [0.61960787, 0.6392157 , 0.49019608, 1.        ],
        [0.61960787, 0.6392157 , 0.49019608, 1.        ],
        ...,
        [0.44313726, 0.5254902 , 0.34117648, 1.        ],
        [0.44705883, 0.5254902 , 0.33333334, 1.        ],
        [0.44705883, 0.5254902 , 0.33333334, 1.        ]],

       ...,

       [[0.4       , 0.53333336, 0.3254902 , 1.        ],
        [0.4       , 0.5294118 , 0.3254902 , 1.        ],
        [0.39607844, 0.5294118 , 0.3254902 , 1.        ],
        ...,
        [0.45490196, 0.57254905, 0.3647059 , 1.        ],
        [0.4509804 , 0.57254905, 0.3647059 , 1.        ],
        [0.4509804 , 0.57254905, 0.36862746, 1.        ]],

       [[0.40784314, 0.5372549 , 0.3372549 , 1.        ],
        [0.40392157, 0.53333336, 0.33333334, 1.        ],
        [0.4       , 0.53333336, 0.33333334, 1.        ],
        ...,
        [0.45490196, 0.5686275 , 0.36862746, 1.        ],
        [0.45490196, 0.5686275 , 0.36862746, 1.        ],
        [0.45490196, 0.5686275 , 0.37254903, 1.        ]],

       [[0.4117647 , 0.54509807, 0.34901962, 1.        ],
        [0.4117647 , 0.5411765 , 0.34901962, 1.        ],
        [0.40784314, 0.5372549 , 0.34509805, 1.        ],
        ...,
        [0.45490196, 0.5647059 , 0.36862746, 1.        ],
        [0.45490196, 0.5647059 , 0.36862746, 1.        ],
        [0.45490196, 0.5647059 , 0.37254903, 1.        ]]], dtype=float
32)
```

In [95]:
```
panda.size,panda.shape,panda.ndim # dimentional is three dimentional
```

Out[95]:
```
(4254768, (1134, 938, 4), 3)
```

```
In [ ]:
```

```
In [ ]:
```