```
In [1]: print("Welcome to Numpy-4")
```

Welcome to Numpy-4

- **Matrix Multiplication**✅
  - `matmul()`, `@`, `dot()` ✅
- **Array splitting and Merging**
  - Splitting arrays - `split()`, `hsplit()`, `vsplit()`
  - Merging Arrays - `hstack()`, `vstack()`
- **Vectorization**✅
  - `np.vectorize()` ✅
- **Broadcasting**✅
- **Dimension Expansion and Reduction**
  - `np.expand_dims()`
  - `np.newaxis`
  - `np.sqeeze()`
- **Shallow vs Deep Copy**
  - `view()`
  - `copy()`
  - `copy.deepcopy()`

```
In [2]: # Matrix Multiplication

In [4]: a=np.arange(1,13).reshape((3,4))
        b=np.arange(2,14).reshape((3,4))
        c=np.arange(1,13).reshape((4,3))

In [5]: a
Out[5]: array([[ 1,  2,  3,  4],
               [ 5,  6,  7,  8],
               [ 9, 10, 11, 12]])

In [6]: b
Out[6]: array([[ 2,  3,  4,  5],
               [ 6,  7,  8,  9],
               [10, 11, 12, 13]])

In [7]: c
Out[7]: array([[ 1,  2,  3],
               [ 4,  5,  6],
               [ 7,  8,  9],
               [10, 11, 12]])

In [8]: a*b
Out[8]: array([[  2,   6,  12,  20],
               [ 30,  42,  56,  72],
               [ 90, 110, 132, 156]])

In [9]: a
Out[9]: array([[ 1,  2,  3,  4],
               [ 5,  6,  7,  8],
               [ 9, 10, 11, 12]])

In [10]: b
Out[10]: array([[ 2,  3,  4,  5],
                [ 6,  7,  8,  9],
                [10, 11, 12, 13]])

In [11]: a.shape
Out[11]: (3, 4)

In [12]: b.shape
Out[12]: (3, 4)

In [13]: c
Out[13]: array([[ 1,  2,  3],
                [ 4,  5,  6],
                [ 7,  8,  9],
                [10, 11, 12]])

In [14]: c.shape
Out[14]: (4, 3)

In [15]: np.matmul(a,c)
Out[15]: array([[ 70,  80,  90],
                [158, 184, 210],
                [246, 288, 330]])

In [16]: np.matmul(c,a)
Out[16]: array([[ 38,  44,  50,  56],
                [ 83,  98, 113, 128],
                [128, 152, 176, 200],
                [173, 206, 239, 272]])

In [18]: # np.matmul(a,b)

In [19]: a@c
Out[19]: array([[ 70,  80,  90],
                [158, 184, 210],
                [246, 288, 330]])

In [20]: np.dot(a,c)
Out[20]: array([[ 70,  80,  90],
                [158, 184, 210],
                [246, 288, 330]])

In [21]: a=np.arange(1,4)

         b=np.arange(2,5)
         print(a)
         print(b)

[1 2 3]
[2 3 4]

In [22]: np.dot(a,b)
Out[22]: 20

In [23]: np.matmul(a,b)
Out[23]: 20

In [24]: c=np.arange(3,7)
         print(c)

[3 4 5 6]

In [26]: # np.dot(a,c)

In [27]: print(a)
         print(b)

[1 2 3]
[2 3 4]

In [28]: a*b
Out[28]: array([ 2,  6, 12])

In [29]: np.matmul(a,b)
Out[29]: 20

In [30]: np.dot(a,b)
Out[30]: 20

In [31]: a
Out[31]: array([1, 2, 3])

In [32]: b=5

In [33]: a*b
Out[33]: array([ 5, 10, 15])

In [34]: np.dot(a,b)
Out[34]: array([ 5, 10, 15])

In [35]: np.matmul(a,b)

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/hd/9z4dczb56dj54lb7q8w7s4zw0000gn/T/ipykernel_15704/684171433.py in <module>
----> 1 np.matmul(a,b)

ValueError: matmul: Input operand 1 does not have enough dimensions (has 0, gufunc core with signature (n?,k),(k,m?)->(n?,m?) requires 1)

In [37]: a=np.arange(1,13).reshape((1,3,4))
         b=np.arange(1,5)
         print(a)
         print(b)

[[[ 1  2  3  4]
  [ 5  6  7  8]
  [ 9 10 11 12]]]
[1 2 3 4]

In [38]: np.dot(a,b)
Out[38]: array([[ 30,  70, 110]])

In [40]: # np.dot(b,a)

In [43]: a=np.arange(1,25).reshape((2,3,4))
         b=np.arange(1,5)
         print(a)
         print(b)

[[[ 1  2  3  4]
  [ 5  6  7  8]
  [ 9 10 11 12]]

 [[13 14 15 16]
  [17 18 19 20]
  [21 22 23 24]]]
[1 2 3 4]

In [44]: np.dot(a,b)
Out[44]: array([[ 30,  70, 110],
                [150, 190, 230]])

In [45]: b=np.arange(1,4)

In [46]: b
Out[46]: array([1, 2, 3])

In [48]: # np.dot(a,b)

In [51]: a=np.arange(1,25).reshape((2,3,4))
         b=np.arange(1,5).reshape((4,1))
         print(a)
         print(b)

[[[ 1  2  3  4]
  [ 5  6  7  8]
  [ 9 10 11 12]]

 [[13 14 15 16]
  [17 18 19 20]
  [21 22 23 24]]]
[[1]
 [2]
 [3]
 [4]]

In [52]: np.dot(a,b)
Out[52]: array([[[ 30],
                 [ 70],
                 [110]],

                [[150],
                 [190],
                 [230]]])

In [53]: a=np.arange(1,25).reshape((2,3,4))
         b=np.arange(1,5).reshape((1,4))
         print(a)
         print(b)
         np.dot(a,b)

[[[ 1  2  3  4]
  [ 5  6  7  8]
  [ 9 10 11 12]]

 [[13 14 15 16]
  [17 18 19 20]
  [21 22 23 24]]]
[[1 2 3 4]]
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/hd/9z4dczb56dj54lb7q8w7s4zw0000gn/T/ipykernel_15704/1102676030.py in <module>
      3 print(a)
      4 print(b)
----> 5 np.dot(a,b)

<__array_function__ internals> in dot(*args, **kwargs)

ValueError: shapes (2,3,4) and (1,4) not aligned: 4 (dim 2) != 1 (dim 0)

In [54]: a=np.arange(1,25).reshape((2,3,4))
         b=np.arange(1,25).reshape((2,3,4))
         print(a)
         print(b)
         np.dot(a,b)

[[[ 1  2  3  4]
  [ 5  6  7  8]
  [ 9 10 11 12]]

 [[13 14 15 16]
  [17 18 19 20]
  [21 22 23 24]]]
[[[ 1  2  3  4]
  [ 5  6  7  8]
  [ 9 10 11 12]]

 [[13 14 15 16]
  [17 18 19 20]
  [21 22 23 24]]]
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/hd/9z4dczb56dj54lb7q8w7s4zw0000gn/T/ipykernel_15704/4099205924.py in <module>
      3 print(a)
      4 print(b)
----> 5 np.dot(a,b)

<__array_function__ internals> in dot(*args, **kwargs)

ValueError: shapes (2,3,4) and (2,3,4) not aligned: 4 (dim 2) != 3 (dim 1)

In [55]: a=np.arange(1,25).reshape((2,3,4))
         b=np.arange(1,25).reshape((2,4,3))
         print(a)
         print(b)
         np.dot(a,b)

[[[ 1  2  3  4]
  [ 5  6  7  8]
  [ 9 10 11 12]]

 [[13 14 15 16]
  [17 18 19 20]
  [21 22 23 24]]]
[[[ 1  2  3]
  [ 4  5  6]
  [ 7  8  9]
  [10 11 12]]

 [[13 14 15]
  [16 17 18]
  [19 20 21]
  [22 23 24]]]
Out[55]: array([[[[  70,   80,   90],
                  [ 190,  200,  210]],

                 [[ 158,  184,  210],
                  [ 470,  496,  522]],

                 [[ 246,  288,  330],
                  [ 750,  792,  834]]],


                [[[ 334,  392,  450],
                  [1030, 1088, 1146]],

                 [[ 422,  496,  570],
                  [1310, 1384, 1458]],

                 [[ 510,  600,  690],
                  [1590, 1680, 1770]]]])

In [ ]:

In [56]: #vectorize

In [57]: a=np.arange(10)
         a
Out[57]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [58]: a*2
Out[58]: array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])

In [ ]:

In [59]: import math
         math.log
Out[59]: <function math.log>

In [60]: a=np.arange(1,10)

In [61]: math.log(a)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/var/folders/hd/9z4dczb56dj54lb7q8w7s4zw0000gn/T/ipykernel_15704/1428990103.py in <module>
----> 1 math.log(a)

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

In [62]: def custom_func(x):
             if x%2==0:
                 return (x**2)**(1/3)
             else:
                 return 3.14*x*x

In [70]: # custom_func(a)

In [ ]:

In [64]: y=np.vectorize(math.log)

In [65]: type(y)
Out[65]: numpy.vectorize

In [66]: y(a)
Out[66]: array([0.        , 0.69314718, 1.09861229, 1.38629436, 1.60943791,
                1.79175947, 1.94591015, 2.07944154, 2.19722458])

In [68]: a
Out[68]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])

In [67]: vectorized_custom_func=np.vectorize(custom_func)

In [69]: z(a)
Out[69]: array([  3.14      ,   1.58740105,  28.26      ,   2.5198421 ,
                 78.5       ,   3.30192725, 153.86      ,   4.        ,
                254.34      ])

In [71]: np.vectorize(custom_func)(a)
Out[71]: array([  3.14      ,   1.58740105,  28.26      ,   2.5198421 ,
                 78.5       ,   3.30192725, 153.86      ,   4.        ,
                254.34      ])

In [ ]:

In [72]: #broadcasting
```

Broadcasting

Rules

For each dimension (going from right side)

1. The size of each dimension should be same OR

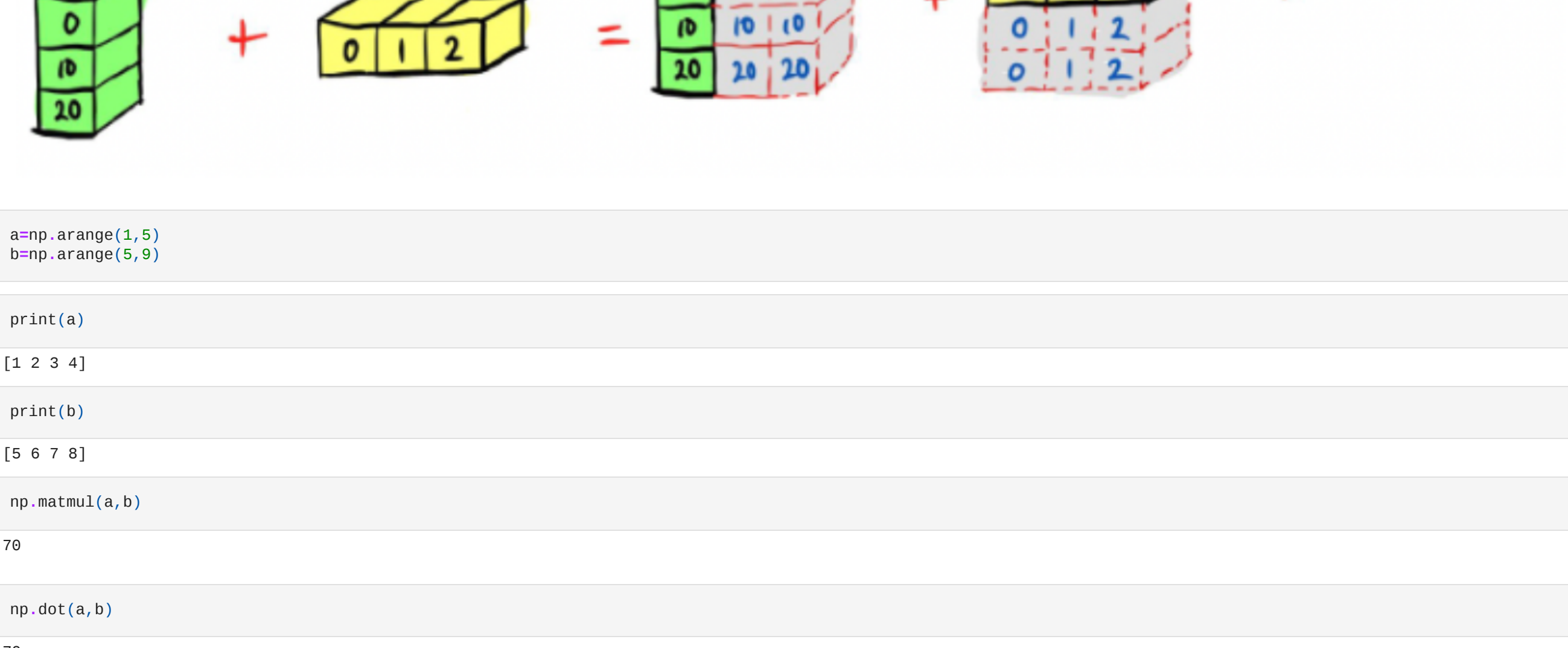
2. The size of one dimension should be 1

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.

Rule 3 : If in any dimesion the sizes disagree and neither equal to 1 , then Error is raised.

```
In [74]: a=np.arange(1,4)
         b=np.arange(2,5)
         print(a)
         print(a.shape)
         print(a.ndim)
         print("-"*50)
         print(b)
         print(b.shape)
         print(b.ndim)

[1 2 3]
(3,)
1
--------------------------------------------------
[2 3 4]
(3,)
1

In [75]: a+b
Out[75]: array([3, 5, 7])

In [76]: a+2
Out[76]: array([3, 4, 5])

In [77]: c=np.arange(1,7).reshape((3,2))
         d=np.arange(1,3)
         print(c)
         print(c.shape)
         print(c.ndim)
         print("-"*50)
         print(d)
         print(d.shape)
         print(d.ndim)

[[1 2]
 [3 4]
 [5 6]]
(3, 2)
2
--------------------------------------------------
[1 2]
(2,)
1

In [78]: c+d
Out[78]: array([[2, 4],
                [4, 6],
                [6, 8]])

In [79]: a
Out[79]: array([1, 2, 3])

In [80]: a+c

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/hd/9z4dczb56dj54lb7q8w7s4zw0000gn/T/ipykernel_15704/3032047929.py in <module>
----> 1 a+c

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

In [81]: a=np.arange(1,7).reshape((2,3))
         b=np.arange(1,19).reshape((2,3,3))

In [82]: a+b

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/hd/9z4dczb56dj54lb7q8w7s4zw0000gn/T/ipykernel_15704/3553919051.py in <module>
----> 1 a+b

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

In [ ]:

In [83]: a=np.arange(1,10).reshape((3,3))
         b=np.arange(1,19).reshape((2,3,3))

In [85]: (a+b).shape
Out[85]: (2, 3, 3)

In [86]: a=np.arange(1,4)
         b=np.arange(1,25).reshape((2,4,3))

In [88]: (a+b).shape
Out[88]: (2, 4, 3)

In [ ]:

In [89]: a=np.arange(1,4).reshape((1,3))
         b=np.array([2])

In [91]: (a+b).shape
Out[91]: (1, 3)

In [92]: a=np.arange(1,4).reshape((1,3))
         b=np.arange(1,4).reshape((3,1))

In [93]: a+b
Out[93]: array([[2, 3, 4],
                [3, 4, 5],
                [4, 5, 6]])

In [94]: a=np.arange(1,7).reshape((3,2))
         b=np.arange(1,4).reshape((1,3))
         a+b

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/hd/9z4dczb56dj54lb7q8w7s4zw0000gn/T/ipykernel_15704/3921988710.py in <module>
      1 a=np.arange(1,7).reshape((3,2))
      2 b=np.arange(1,4).reshape((1,3))
----> 3 a+b

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



```
In [98]: a=np.arange(1,5)
         b=np.arange(5,9)

In [99]: print(a)

[1 2 3 4]

In [100…: print(b)

[5 6 7 8]

In [101…: np.matmul(a,b)
Out[101…: 70

In [102…: np.dot(a,b)
Out[102…: 70

In [103…: a=np.arange(1,4).reshape((3,1))
          b=np.arange(1,4)
          c=np.arange(1,4).reshape((3,1,1))

In [111…: a+b+c
Out[111…: array([[[3, 4, 5],
                  [4, 5, 6],
                  [5, 6, 7]],

                 [[4, 5, 6],
                  [5, 6, 7],
                  [6, 7, 8]],

                 [[5, 6, 7],
                  [6, 7, 8],
                  [7, 8, 9]]])

In [113…: (a+b+c).flatten().reshape((3,3,3))
Out[113…: array([[[3, 4, 5],
                  [4, 5, 6],
                  [5, 6, 7]],

                 [[4, 5, 6],
                  [5, 6, 7],
                  [6, 7, 8]],

                 [[5, 6, 7],
                  [6, 7, 8],
                  [7, 8, 9]]])

In [ ]:
```