numpy3

September 17, 2025

1 Numpy 3

1.0.1 1. Element wise operations

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

```
[21]: a = np.arange(1, 6)
```

```
[3]: a * 5
```

```
[3]: array([ 5, 10, 15, 20, 25])
```

```
[4]: a
```

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

```
[6]: a ** 2
```

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

```
[7]: b = np.arange(6, 11)
```

```
[8]: b
```

```
[8]: array([ 6,  7,  8,  9, 10])
```

```
[9]: b + 2
```

```
[9]: array([ 8,  9, 10, 11, 12])
```

```
[11]: a
```

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

```
[12]: b
```

```
[12]: array([ 6,  7,  8,  9, 10])
```

```
[10]: a * b
```

```
[10]: array([ 6, 14, 24, 36, 50])
```

```
[13]: # What if size of these arrays aren't same?
```

```
[19]: c = np.arange(3)
```

```
[20]: c
```

```
[20]: array([0, 1, 2])
```

```
[17]: a * c
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/t5/yhjgrjs907zfp250jyxtw54m0000gn/T/ipykernel_64912/2570290981.py␣
 ↪in <module>
----> 1 a * c

ValueError: operands could not be broadcast together with shapes (5,) (3,)
```

```
[23]: # a + c
```

```
[24]: # 2D arrays element wise operations
```

```
[25]: d = np.arange(12).reshape(3, 4)
      e = np.arange(13, 25).reshape(3, 4)
```

```
[26]: d
```

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

```
[27]: e
```

```
[27]: array([[13, 14, 15, 16],
             [17, 18, 19, 20],
             [21, 22, 23, 24]])
```

```
[28]: d + e
```

```
[28]: array([[13, 15, 17, 19],
             [21, 23, 25, 27],
             [29, 31, 33, 35]])
```

```
[29]: d * e
```

```
[29]: array([[  0,  14,  30,  48],
             [ 68,  90, 114, 140],
             [168, 198, 230, 264]])
```

```
[30]: d.ndim
```

```
[30]: 2
```

```
[31]: a
```

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

```
[33]: d
```

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

```
[34]: d + a
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/t5/yhjgrjs907zfp250jyxtw54m0000gn/T/ipykernel_64912/1526614864.py␣
 ↪in <module>
----> 1 d + a

ValueError: operands could not be broadcast together with shapes (3,4) (5,)
```

```
[35]: # Array * number -> Works
      # Array * Array(Same shape) -> Works
      # Array * Array(Different shape) -> Doesn't work
```

```
[36]: d
```

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

```
[37]: e = np.arange(4)
```

```
[38]: e
```

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

[42]:
```
# Following is an example of broadcasting of an array along with all the rows:
```

[40]:
```
d + e
```

[40]:
```
array([[ 0,  2,  4,  6],
       [ 4,  6,  8, 10],
       [ 8, 10, 12, 14]])
```

[41]:
```
# HW: Add int array with float array. Find out the output?
```

[48]:
```
output = d + e
```

[49]:
```
output
```

[49]:
```
array([[ 0,  2,  4,  6],
       [ 4,  6,  8, 10],
       [ 8, 10, 12, 14]])
```

[51]:
```
d
```

[51]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

[52]:
```
e
```

[52]:
```
array([0, 1, 2, 3])
```

[53]:
```
d * e
```

[53]:
```
array([[ 0,  1,  4,  9],
       [ 0,  5, 12, 21],
       [ 0,  9, 20, 33]])
```

[50]:
```
np.sum(output)
```

[50]:
```
84
```

[46]:
```
np.sum(d)
```

[46]:
```
66
```

[47]:
```
np.sum(e)
```

[47]:
```
6
```

[45]:
```
np.sum(d + e)
```

```
[45]: 84
```

```
[54]: d
```

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

```
[55]: g = np.arange(3).reshape(3, 1)
```

```
[56]: g
```

```
[56]: array([[0],
             [1],
             [2]])
```

```
[57]: d + g
```

```
[57]: array([[ 0,  1,  2,  3],
             [ 5,  6,  7,  8],
             [10, 11, 12, 13]])
```

```
[58]: ar = np.arange(8).reshape(2, 4)
```

```
[59]: ar
```

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

```
[60]: d + ar
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/t5/yhjgrjs907zfp250jyxtw54m0000gn/T/ipykernel_64912/647454574.py i
  ↪<module>
----> 1 d + ar

ValueError: operands could not be broadcast together with shapes (3,4) (2,4)
```

```
[ ]:
```

```
[ ]:
```

1.0.2 Matrix multiplication

- np.matmul()
- np.dot()

- a @ b (python way of matrix multiplication -> python 3.5 onwards)

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

```
[66]: a
```

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

```
[67]: b
```

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

```
[68]: np.matmul(a, b)
```

```
[68]: array([[ 80,  90, 100],
             [184, 210, 236],
             [288, 330, 372]])
```

```
[69]: np.dot(a, b)
```

```
[69]: array([[ 80,  90, 100],
             [184, 210, 236],
             [288, 330, 372]])
```

```
[70]: a @ b
```

```
[70]: array([[ 80,  90, 100],
             [184, 210, 236],
             [288, 330, 372]])
```

```
[71]: np.matmul(a, 5)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/t5/yhjgrjs907zfp250jyxtw54m0000gn/T/ipykernel_64912/2308400740.py␣
 ↪in <module>
----> 1 np.matmul(a, 5)

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

```
[73]: # Following is not doable
      # a @ 5
```

```
[75]: a
```

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

```
[74]: np.dot(a, 5)
```

```
[74]: array([[ 5, 10, 15, 20],
             [25, 30, 35, 40],
             [45, 50, 55, 60]])
```

```
[76]: np.dot(2, 5)
```

```
[76]: 10
```

```
[77]: e
```

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

```
[78]: np.dot(e, 3)
```

```
[78]: array([0, 3, 6, 9])
```

```
[ ]:
```

1.0.3 Shallow copy vs Deep copy

```
[82]: # help(np)
```

```
[86]: # Shallow copy
```

```
[90]: l1 = [1, 2, 3, 4]
      l2 = l1
```

```
[91]: l2[0] = 10
```

```
[92]: print(l1)
```

```
[10, 2, 3, 4]
```

```
[94]: print(id(l1), id(l2))
```

```
140318741021312 140318741021312
```

```
[95]: # Arrays
```

```
[103]: # Shallow copy is being created in below cell
```

```
[96]: a1 = np.arange(4)
      a2 = a1
```

```
[97]: a1
```

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

```
[98]: a2
```

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

```
[99]: a2[0] = 10
```

```
[100]: a1
```

```
[100]: array([10,  1,  2,  3])
```

```
[101]: np.shares_memory(a1, a2)
```

```
[101]: True
```

```
[104]: c = a1 + 2 # element wise operations -> Deep copy
```

```
[106]: c[0] = 15
```

```
[107]: a1
```

```
[107]: array([10,  1,  2,  3])
```

```
[108]: id(a1)
```

```
[108]: 140319830716112
```

```
[109]: id(c)
```

```
[109]: 140318742875664
```

```
[110]: id(a2)
```

```
[110]: 140319830716112
```

```
[111]: np.shares_memory(a1, c)
```

```
[111]: False
```

```
[113]: a = np.arange(4)
```

```
[114]: b = a.reshape(2, 2)
```

```
[115]: b[0] = 20
```

```
[116]: a
```

```
[116]: array([20, 20,  2,  3])
```

```
[117]: b
```

```
[117]: array([[20, 20],
              [ 2,  3]])
```

```
[118]: np.shares_memory(a, b)
```

```
[118]: True
```

```
[119]: id(a)
```

```
[119]: 140318743537200
```

```
[120]: id(b)
```

```
[120]: 140318742874320
```

```
[121]: a[0]
```

```
[121]: 20
```

```
[122]: b[0][0]
```

```
[122]: 20
```

```
[123]: print(id(a[0]), id(b[0][0]))
```

```
140318742938448 140318742938448
```

```
[124]: a
```

```
[124]: array([20, 20,  2,  3])
```

```
[125]: b = a
```

```
[126]: id(a)

[126]: 140318743537200

[127]: id(b)

[127]: 140318743537200

[128]: # view -> shallow copy
       b = a.view()

[129]: np.shares_memory(a, b)

[129]: True

[130]: # copy -> deep copy

[131]: c = a.copy()

[132]: np.shares_memory(a, c)

[132]: False

[133]: c

[133]: array([20, 20,  2,  3])

[134]: c[0] = 90

[135]: a

[135]: array([20, 20,  2,  3])

[ ]:

[136]: #homework
       a = np.array([1,2,3,4,5,6])
       b = a[a%2==0] #masking

       #deep copy / shallow copy

[137]: #slicing
       c = a[:2] # deep copy/ shallow copy
       #np.share_memory()

[ ]:
```

1.0.4 Splitting

```
[139]: a = np.arange(9)
```

```
[140]: np.split(a, 3)
```

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

```
[142]: # np.split(a, 4)
```

```
[144]: # Unequal sizes
```

```
[143]: np.split(a, (3, 6, 7))
```

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

```
[ ]:
```

```
[145]: # 2D array
```

```
[146]: a = np.arange(16).reshape(4, 4)
```

```
[147]: a
```

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

```
[148]: np.split(a, 2)
```

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

```
[156]: np.split(a, 2, axis = 0) # vertical
```

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

```
[155]: np.split(a, 2, axis = 1) # Horizontal
```

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

```
[ ]:
```

```
[151]: # hsplit, vsplit
```

```
[152]: a
```

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

```
[154]: np.hsplit(a, 2)
```

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

```
[157]: np.vsplit(a, 2)
```

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

```
[ ]:
```

```
[158]: # Slicing : Will give you subarray
       # Splitting : Split your original array into multiple smaller ones
```

```
[159]: # HW: How to access these arrays after splitting?
```

```
[ ]:
```

1.0.5 Stacking

```
[160]: a = np.arange(5)
```

```
[161]: b = np.arange(5, 10)
```

```
[162]: a
```

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

```
[163]: b
```

```
[163]: array([5, 6, 7, 8, 9])
```

```
[165]: np.vstack((a, b))
```

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

```
[166]: np.hstack((a, b))
```

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

```
[167]: np.vstack((a, a, a))
```

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

```
[ ]:
```

```
[168]: # Quiz
```

```
[170]: a = np.array([[1], [2], [3]])
       b = np.array([[4], [5], [6]])
```

```
[174]: np.vstack((a, b))
```

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

```
[171]: a
```

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

```
[172]: b
```

```
[172]: array([[4],
              [5],
              [6]])
```

```
[173]: a = np.array([[1], [2], [3]])
       b = np.array([[4], [5], [6]])
       np.hstack((a, b))
```

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

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```