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

In [2]:
```
import numpy as np
arr = np.array([1,2,3,4,5,6,7,8,9,10,11,12]) #This is 1d array
newarr1 = arr.reshape(2,3,2) #With reshape function now converted it into 3d arr
newarr2 = arr.reshape(4,3) #2d array
print(newarr1)

print(newarr2)
```

```
[[[ 1  2]
  [ 3  4]
  [ 5  6]]

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

In [3]:
```
newarr1
```

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

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

In [4]:
```
newarr1 = [...,2]
print(newarr1)
```

```
[Ellipsis, 2]
```

In [5]:
```
newarr1 = [1,...]
print(newarr1)
```

```
[1, Ellipsis]
```

2. []

In [6]:
```
import numpy as np
b = np.arange (0,20)
print(b)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
```

In [7]:
```
b1 = np.reshape(b,(5,4))
print(b1)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]]
```

In [8]:
```
b1[0,0]
```

Out[8]:
```
np.int64(0)
```

In [9]:
```
b1[-1]
```

Out[9]:
```
array([16, 17, 18, 19])
```

In [10]:
```
b1[1:3,1:4]
```

Out[10]:
```
array([[ 5,  6,  7],
       [ 9, 10, 11]])
```

In [11]:
```
b1
```

Out[11]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19]])
```

In [12]:
```
b1[1:5]
```

Out[12]:
```
array([[ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19]])
```

In [13]:
```
b1>15
```

Out[13]:
```
array([[False, False, False, False],
       [False, False, False, False],
       [False, False, False, False],
       [False, False, False, False],
       [ True,  True,  True,  True]])
```

In [14]:
```
a = np.array([ [ 0,1,2,3,4],[10,11,12,13,14],[20,21,22,23,24],[30,31,32,33,34] ]
c = np.array( [True,False,True,False] ) #boolean row selector
print(a)
```

```
[[ 0  1  2  3  4]
 [10 11 12 13 14]
 [20 21 22 23 24]
 [30 31 32 33 34]]
```

In [15]:
```
a[c,:]
```

Out[15]:
```
array([[ 0,  1,  2,  3,  4],
       [20, 21, 22, 23, 24]])
```

In [16]:
```
d = np.array([False,True,True,False,True]) #boolean column selector
a[:,d]
```

Out[16]:
```
array([[ 1,  2,  4],
       [11, 12, 14],
       [21, 22, 24],
       [31, 32, 34]])
```

```
a
```

In [17]:
```
i = np.array([0,1,2,1])
j = np.array([1,2,3,4])
```

In [18]:
```
a[i,j]
```

Out[18]:
```
array([ 1, 12, 23, 14])
```

3.abs() & 4.ABSOLUTE- both are same -Synonym for abs()

In [19]:
```
import numpy as np
ARR = np.arange(-4,5).reshape(3,3)
ARR
```

Out[19]:
```
array([[-4, -3, -2],
       [-1,  0,  1],
       [ 2,  3,  4]])
```

In [20]:
```
ARR1 = 1.2+ 5J
ARR1
```

Out[20]:
```
(1.2+5j)
```

In [21]:
```
abs(ARR1)
```

Out[21]:
```
5.141984052872976
```

In [22]:
```
np.absolute(ARR1)
```

Out[22]:
```
np.float64(5.141984052872976)
```

In [23]:
```
np.absolute([-10.-15])
```

Out[23]:
```
array([25.])
```

5.accumulate

In [24]:
```
import numpy as np
```

In [25]:
```
A1 = np.arange(0,10)
print(A1)
```

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

In [26]:
```
np.add.accumulate(A1)
```

Out[26]:
```
array([ 0,  1,  3,  6, 10, 15, 21, 28, 36, 45])
```

In [27]:
```
A2 = np.arange(1,6)
```

In [28]:
```
np.multiply.accumulate(A2)
```

Out[28]:
```
array([  1,   2,   6,  24, 120])
```

In [29]:
```
AC = np.array([[1,2,3],[4,5,6]])
AC
```

Out[29]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [30]:
```
np.add.accumulate(AC)
```

Out[30]:
```
array([[1, 2, 3],
       [5, 7, 9]])
```

In [31]:
```
np.multiply.accumulate(AC)
```

Out[31]:
```
array([[ 1,  2,  3],
       [ 4, 10, 18]])
```

6.add

In [32]:
```
import numpy as np
```

In [33]:
```
ar1 =[2,-6]
ar2 = [1,3]
ar3 = [2.2,8.7]
ar4 = [7,-3]
print(ar1,ar2,ar3,ar4)
```

```
[2, -6] [1, 3] [2.2, 8.7] [7, -3]
```

np.add(ar1 , ar2)

In [34]:
```
np.add(ar3,ar4)
```

Out[34]:
```
array([9.2, 5.7])
```

In [35]:
```
C1 = 5+2j
```

In [36]:
```
np.add(ar3,C1)
```

Out[36]:
```
array([ 7.2+2.j, 13.7+2.j])
```

7.all

In [37]:
```
a = np.arange(0,10)
a
```

Out[37]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [38]:
```
np.all(a>5)
```

Out[38]:
```
np.False_
```

In [39]:
```
np.all(a>=0)
```

Out[39]:
```
np.True_
```

In [40]:
```
b7 = np.array([True,False,True,True])
np.all(b7)
```

Out[40]:
```
np.False_
```

8.allclose

In [41]:
```
np.allclose([0.11,0.0033],[0.11000005,0.00330001])
```

Out[41]: True

In [42]:
```
np.allclose([1e10,1e-8],[1.0001e10,1e-9])
```

Out[42]: False

In [43]:
```
np.allclose([1e10,1e-8],[1.00001e10,1e-9])
```

Out[43]: True

9. alltrue-same as all function

In [44]:
```
b = np.array([True, False, True, True])
np.alltrue(b)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[44], line 2
      1 b = np.array([True, False, True, True])
----> 2 np.alltrue(b)

File ~\AppData\Roaming\Python\Python313\site-packages\numpy\__init__.py:794, in __getattr__(attr)
    791     raise AttributeError(__former_attrs__[attr], name=None)
    793 if attr in __expired_attributes__:
--> 794     raise AttributeError(
    795         f"`np.{attr}` was removed in the NumPy 2.0 release. "
    796         f"{__expired_attributes__[attr]}",
    797         name=None
    798     )
    800 if attr == "chararray":
    801     warnings.warn(
    802         "`np.chararray` is deprecated and will be removed from "
    803         "the main namespace in the future. Use an array with a string "
    804         "or bytes dtype instead.", DeprecationWarning, stacklevel=2)

AttributeError: `np.alltrue` was removed in the NumPy 2.0 release. Use `np.all` instead.
```

In [ ]:
```
a = np.array([1,5,2,7])
np.alltrue(a >= 5)
```

In [ ]:
```
np.alltrue(a>0)
```

10. angle

In [ ]:
```
np.angle([5+3j, 1j, 1.0])
```

In [ ]:
```
np.angle([5+3j, 1j, 1.0], deg=True)
```

In [ ]:
```
np.angle(1+1j,deg=True)
```

In [ ]:
```
np.angle(1+1j,deg=False)
```

11.any

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

In [ ]:
```
c2 = np.array([True,False,True])
```

In [ ]:
```
np.any(c2)
```

In [ ]:
```
c3 = np.arange(0,10)
c3
```

In [ ]:
```
any(c3>=5)
```

12.append

In [45]:
```
c3= np.append([1,2,3],[[4,5,6],[7,8,9]])
```

In [46]:
```
c3
```

Out[46]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [47]:
```
c5 = np.append(c3,10)
c5
```

Out[47]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

In [48]:
```
np.append(c5,('ABC',1 +2j))
```

Out[48]:
```
array(['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'ABC', '(1+2j)'],
      dtype='<U64')
```

In [49]:
```
c9 = np.arange(10,61,10)
c9
```

Out[49]:
```
array([10, 20, 30, 40, 50, 60])
```

In [50]:
```
np.append(c9,(70,80,90))
```

Out[50]:
```
array([10, 20, 30, 40, 50, 60, 70, 80, 90])
```

In [51]:
```
c10 = np.reshape(c9, (2,3))
c10
```

Out[51]:
```
array([[10, 20, 30],
       [40, 50, 60]])
```

In [52]:
```
c11 = np.append(c10, [[70,80,90]],axis = 0)  #AT ROW
c11
```

Out[52]:
```
array([[10, 20, 30],
       [40, 50, 60],
       [70, 80, 90]])
```

In [53]:
```
c13 = np.append(c11,[[100], [120], [130]],axis = 1)
c13
```

Out[53]:
```
array([[ 10,  20,  30, 100],
       [ 40,  50,  60, 120],
       [ 70,  80,  90, 130]])
```

13.apply_along_axis

it will apply function to axis

In [54]:
```
def func1(Q): #EX-1
    return (Q[0]+Q[-1])
abc = np.array([[1,2,3],[4,5,6],[7,8,9]])
abc
```

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

In [55]:
```
np.apply_along_axis(func1,0,abc)
```

Out[55]:
```
array([ 8, 10, 12])
```

In [56]:
```
np.apply_along_axis(func1,1,abc)
```

Out[56]:
```
array([ 4, 10, 16])
```

In [57]:
```
def func2(p): #2
    return (p[0] + p[-1])
abc1 = np.array([[1,2,3],[4,5,6],[7,8,9]])
abc1
```

Out[57]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [58]:
```
np.apply_along_axis(func2,0,abc1)
```

Out[58]:
```
array([ 8, 10, 12])
```

In [59]:
```
np.apply_along_axis(func1,1,abc)
```

Out[59]:
```
array([ 4, 10, 16])
```

In [60]:
```
def func2(p):
    return (p[0] + p[-1])*2
abc1 = np.array([[1,2,3],[4,5,6],[7,8,9]])
abc1
```

Out[60]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [61]:
```
np.apply_along_axis(func2,0,abc1)
```

Out[61]:
```
array([16, 20, 24])
```

In [62]:
```
np.apply_along_axis(func2,1,abc1)
```

Out[62]:
```
array([ 8, 20, 32])
```

14.apply_over_axes()

In [63]:
```
D = np.arange(24)
D
```

Out[63]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23])
```

In [64]:
```
D=np.reshape(D, (2,3,4))
D
```

Out[64]:
```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],

       [[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]]])
```

In [65]:
```
DQ = np.apply_over_axes(np.sum,D, [0,2])
DQ
```

Out[65]:
```
array([[[ 60],
        [ 92],
        [124]]])
```

15.arrange

In [66]:
```
np.arange(9)
```

Out[66]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8])
```

In [67]:
```
np.arange(11.0)
```

Out[67]:
```
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])
```

In [68]:
```
np.arange(8, dtype=float)
```

Out[68]:
```
array([0., 1., 2., 3., 4., 5., 6., 7.])
```

In [69]:
```
np.arange(5,10)
```

Out[69]:
```
array([5, 6, 7, 8, 9])
```

In [70]:
```
np.arange(3,30,9)
```

Out[70]:
```
array([ 3, 12, 21])
```

16.arccos

In [71]:
```
np.arccos([1,-1])
```

Out[71]:
```
array([0.        , 3.14159265])
```

17.arccosh

In [72]:
```
np.arccosh([np.e,10.0])
```

Out[72]:
```
array([1.65745445, 2.99322285])
```

In [73]:
```
np.arccosh(1)
```

Out[73]:
```
np.float64(0.0)
```

18.arcsin

In [74]:
```
np.arcsin([1,-1,0])
```

Out[74]:
```
array([ 1.57079633, -1.57079633,  0.        ])
```

19.arcsinh

20.arctan

In [75]:
```
np.arctan([0,1,-1])
```

Out[75]:
```
array([ 0.        ,  0.78539816, -0.78539816])
```

21.arctan2

In [76]:
```
x = np.array([-1, +1, +1 ,-1])
y = np.array([-1, -1 ,+1 , +1])
np.arctan2(y, x) * 180 / np.pi  #in Degree unit
```

Out[76]:
```
array([-135.,  -45.,   45.,  135.])
```

In [77]:
```
np.arctan2(y,x)
```

Out[77]:
```
array([-2.35619449, -0.78539816,  0.78539816,  2.35619449])
```

In [78]:
```
np.arctan2([0, 1],[1, 0])
```

Out[78]:
```
array([0.        , 1.57079633])
```

22.arctanh

In [79]:
```
np.arctanh([0,-0.5])
```

Out[79]:
```
array([ 0.        , -0.54930614])
```

23.argmax

In [80]:
```
j = np.array([0,11,95,2,-5,55])
```

In [81]:
```
np.argmax(j)
```

Out[81]:
```
np.int64(2)
```

In [82]:
```
j1 = np.arange(6).reshape(2,3) +10
j1
```

Out[82]:
```
array([[10, 11, 12],
       [13, 14, 15]])
```

In [83]:
```
np.argmax(j1)
```

Out[83]:
```
np.int64(5)
```

In [84]:
```
j2 = np.array([[10,50,32],[60,20,60]])
```

In [85]:
```
j2 = np.array([[1,9,0,4],[2,0,8,-1]])
j2
```

Out[85]:
```
array([[ 1,  9,  0,  4],
       [ 2,  0,  8, -1]])
```

In [86]:
```
np.argmax(j2, axis =0)
```

Out[86]:
```
array([1, 0, 1, 0])
```

In [87]:
```
np.argmax(j2, axis=1)
```

Out[87]:
```
array([1, 2])
```

24.argmin

In [88]:
```
d4 = np.array([0,11,95,2,-5,55])
```

In [89]:
```
np.argmin(d4)
```

Out[89]:
```
np.int64(4)
```

In [90]:
```
d9 = np.arange(6).reshape(2,3) +10
d9
```

Out[90]:
```
array([[10, 11, 12],
       [13, 14, 15]])
```

In [91]:
```
np.argmin(d9)
```

Out[91]:
```
np.int64(0)
```

In [92]:
```
da2 = np.array([[50,50,10],[60,10,40]])
da2
```

Out[92]:
```
array([[50, 50, 10],
       [60, 10, 40]])
```

In [93]:
```
np.argmin(da2)
```

Out[93]:
```
np.int64(2)
```

In [94]:
```
np.argmin(da2, axis=0)
```

Out[94]:
```
array([0, 1, 0])
```

In [95]:
```
np.argmin(da2, axis=1)
```

Out[95]:
```
array([2, 1])
```

25.argsoft

In [96]:
```
AB1 = np.array([2,0,1,5,4,1,9])
```

In [97]:
```
BQ = np.argsort(AB1)
BQ
```

Out[97]:
```
array([1, 2, 5, 0, 4, 3, 6])
```

In [98]:
```
AB1[BQ]
```

Out[98]:
```
array([0, 1, 1, 2, 4, 5, 9])
```

In [99]:
```
aq1 = np.array([[8,4,1],[2,0,9]])
aq1
```

Out[99]:
```
array([[8, 4, 1],
       [2, 0, 9]])
```

In [100…
```
JQ1 = aq1.argsort(axis =0)
JQ1
```

Out[100…
```
array([[1, 1, 0],
       [0, 0, 1]])
```

In [101…
```
aq1[JQ1,[[0,1,2], [0,1,2]]]
```

Out[101…
```
array([[2, 0, 1],
       [8, 4, 9]])
```

In [102…
```
JQ1 = aq1.argsort(axis=1)
JQ1
```

Out[102…
```
array([[2, 1, 0],
       [1, 0, 2]])
```

26.array

In [103…
```
np.array([1,2,3.0])
```

Out[103…
```
array([1., 2., 3.])
```

In [104…
```
np.array([[1,2],[3,4]])
```

Out[104…
```
array([[1, 2],
       [3, 4]])
```

In [105…
```
np.array([1,2,3],dtype=complex)
```

Out[105…
```
array([1.+0.j, 2.+0.j, 3.+0.j])
```

In [106…
```
np.array([1,2,3], ndmin =2)
```

Out[106…
```
array([[1, 2, 3]])
```

In [110…
```
np.array(1, copy=0, subok=1, ndmin=1) #basically equivalent to atleast_1d
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[110], line 1
----> 1 np.array(1, copy=0, subok=1, ndmin=1)

ValueError: Unable to avoid copy while creating an array as requested.
If using `np.array(obj, copy=False)` replace it with `np.asarray(obj)` to allow a
copy when needed (no behavior change in NumPy 1.x).
For more details, see https://numpy.org/devdocs/numpy_2_0_migration_guide.html#ad
apting-to-changes-in-the-copy-keyword.
```

In [111…
```
np.array(1, copy=0, subok=1, ndmin=2)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[111], line 1
----> 1 np.array(1, copy=0, subok=1, ndmin=2)

ValueError: Unable to avoid copy while creating an array as requested.
If using `np.array(obj, copy=False)` replace it with `np.asarray(obj)` to allow a
copy when needed (no behavior change in NumPy 1.x).
For more details, see https://numpy.org/devdocs/numpy_2_0_migration_guide.html#ad
apting-to-changes-in-the-copy-keyword.
```

In [112…
```
np.array(1, subok=1, ndmin=2)
```

Out[112…
```
array([[1]])
```

27.arrayrange

synonym for arange

28.array split

In [114…
```
c4 = np.array([[1,2,3,4],[5,6,7,8]])
c4
```

Out[114…
```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

In [115…
```
np.array_split(c4,2,axis=0)
```

Out[115…
```
[array([[1, 2, 3, 4]]), array([[5, 6, 7, 8]])]
```

In [116…
```
np.array_split(c4,4,axis=1)
```

Out[116…
```
[array([[1],
        [5]]),
 array([[2],
        [6]]),
 array([[3],
        [7]]),
 array([[4],
        [8]])]
```

In [117…
```
np.array_split(c4,[2,3],axis=1)
```

Out[117…
```
[array([[1, 2],
        [5, 6]]),
 array([[3],
        [7]]),
 array([[4],
        [8]])]
```

In [118…
```
np.array_split(c4,3,axis=1)
```

Out[118…
```
[array([[1, 2],
        [5, 6]]),
 array([[3],
        [7]]),
 array([[4],
        [8]])]
```

29.asarray

In [119…
```
my_list = [1, 3, 5, 7, 9]
np.asarray(my_list)
```

Out[119…
```
array([1, 3, 5, 7, 9])
```

In [121…
```
my_tuple = [1,3,9],[8,2,6]
np.asarray(my_tuple)
```

Out[121…
```
array([[1, 3, 9],
       [8, 2, 6]])
```

In [122…
```
m = np.matrix('1 2; 5 8')
m
```

Out[122…
```
matrix([[1, 2],
        [5, 8]])
```

In [123…
```
B9 = np.asarray(m)
B9
```

Out[123…
```
array([[1, 2],
       [5, 8]])
```

In [124…
```
m[0,1] = -190
m
```

Out[124…
```
matrix([[   1, -190],
        [   5,    8]])
```

In [125…
```
B9
```

Out[125…
```
array([[   1, -190],
       [   5,    8]])
```

30.asanyarray

In [126…
```
c7 = np.array([[1,2],[5,8]])
c7
```

Out[126…
```
array([[1, 2],
       [5, 8]])
```

In [127…
```
mq = np.matrix('1 2; 5 8')
mq
```

Out[127…
```
matrix([[1, 2],
        [5, 8]])
```

In [128…
```
np.asanyarray(c7)
```

Out[128…
```
array([[1, 2],
       [5, 8]])
```

In [129…
```
np.asanyarray(mq)
```

Out[129…
```
matrix([[1, 2],
        [5, 8]])
```

In [130…
```
my_tuple1 = ([1, 3, 9], [8, 2, 6])
```

In [131…
```
np.asanyarray(my_tuple1)
```

Out[131…
```
array([[1, 3, 9],
       [8, 2, 6]])
```

31.asmatrix

In [132…
```
x = np.array([[1,2], [3, 4]])
m = np.asmatrix(x)
m
```

Out[132…
```
matrix([[1, 2],
        [3, 4]])
```

In [133…
```
x[0,0] = 5
print(m)

print(x)
```

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

32.astype

In [134…
```
x6 = np.array([1, 2, 2.5])
x6
```

Out[134… array([1. , 2. , 2.5])

In [135…
```
x6.astype(int)
```

Out[135… array([1, 2, 2])

In [136…
```
x6.astype(None)
```

Out[136… array([1. , 2. , 2.5])

33.atleast 1d

In [137…
```
a01 = 1
b01 = np.array([2,3])
c01 = np.array([[4,5],[6,7]])
d01 = np.arange(8).reshape(2,2,2)
d01
```

Out[137…
```
array([[[0, 1],
        [2, 3]],

       [[4, 5],
        [6, 7]]])
```

In [138…
```
np.atleast_1d(a01,b01,c01,d01)
```

Out[138…
```
(array([1]),
 array([2, 3]),
 array([[4, 5],
        [6, 7]]),
 array([[[0, 1],
         [2, 3]],

        [[4, 5],
         [6, 7]]]))
```

34.atleast_2d()

In [140…
```
a11 = 1
b11 = np.array([2,3])
c11 = np.array([[4,5],[6,7]])
d11 = np.arange(8).reshape(2,2,2)
d11
```

Out[140…
```
array([[[0, 1],
        [2, 3]],

       [[4, 5],
        [6, 7]]])
```

In [141…
```
np.atleast_2d(a11,b11,c11,d11)
```

Out[141…
```
(array([[1]]),
 array([[2, 3]]),
 array([[4, 5],
        [6, 7]]),
 array([[[0, 1],
         [2, 3]],

        [[4, 5],
         [6, 7]]]))
```

35.atleast_3d()

In [142…
```
f = 1
g = np.array([2,3])
h = np.array([[4,5],[6,7]])
i = np.arange(8).reshape(2,2,2)
i
```

Out[142…
```
array([[[0, 1],
        [2, 3]],

       [[4, 5],
        [6, 7]]])
```

In [143…
```
np.atleast_3d(f,g,h,i)
```

Out[143…
```
(array([[[1]]]),
 array([[[2],
         [3]]]),
 array([[[4],
         [5]],

        [[6],
         [7]]]),
 array([[[0, 1],
         [2, 3]],

        [[4, 5],
         [6, 7]]]))
```

36. Average

In [144…
```
bz = np.array([1,4,5,7,5])
bz
```

Out[144…
```
array([1, 4, 5, 7, 5])
```

In [145…
```
np.average(bz)
```

Out[145…
```
np.float64(4.4)
```

In [146…
```
w = np.array([0.1,0.2,0.5,0.2,0.2])
```

In [147…
```
np.average(bz,weights=w)
```

Out[147…
```
np.float64(4.833333333333333)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: