In [70]:

```
import numpy as np
a=np.array([1,2,3])
b=np.array([(1,5,2),(4,5,6)],dtype =float)
```

In [72]:

```
np.add(b,a)
```

Out[72]:

```
array([[2., 7., 5.],
       [5., 7., 9.]])
```

In [73]:

```
np.divide(a,b)
```

Out[73]:

```
array([[1.  , 0.4 , 1.5 ],
       [0.25, 0.4 , 0.5 ]])
```

In [5]:

```
np.multiply(a,b)
```

Out[5]:

```
array([[ 1., 10.,  6.]])
```

In [6]:

```
np.exp(b)
```

Out[6]:

```
array([[  2.71828183, 148.4131591 ,   7.3890561 ]])
```

In [7]:

```
np.sqrt(b)
```

Out[7]:

```
array([[1.        , 2.23606798, 1.41421356]])
```

In [8]:

```
np.sin(a)
```

Out[8]:

```
array([[0.84147098, 0.90929743, 0.14112001]])
```

In [9]:

```
np.cos(a)
```

Out[9]:

```
array([[ 0.54030231, -0.41614684, -0.9899925 ]])
```

In [10]:

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

Out[10]:

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

In [11]:

```
np.ones((2,3,4),dtype=np.int16)
```

Out[11]:

```
array([[[1, 1, 1, 1],
        [1, 1, 1, 1],
        [1, 1, 1, 1]],

       [[1, 1, 1, 1],
        [1, 1, 1, 1],
        [1, 1, 1, 1]]], dtype=int16)
```

In [12]:

```
len(a)
```

Out[12]:

```
1
```

In [13]:

```
a.sum()
```

Out[13]:

```
6.0
```

In [14]:

```
a.min()
```

Out[14]:

```
1.0
```

In [15]:

```
np.log(a)
```

Out[15]:

```
array([[0.        , 0.69314718, 1.09861229]])
```

In [16]:

```
a.sort()
```

In [17]:

```
h=a.view()
```

In [19]:

```
np.copy(a)
```

Out[19]:

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

In [20]:

```
h=a.copy()
```

In [21]:

```
a.sort()
```

In [22]:

```
np.std(b)
```

Out[22]:

```
1.699673171197595
```

In [23]:

```
np.corrcoef(a)
```

Out[23]:

```
1.0
```

In [24]:

```
np.median(b)
```

Out[24]:

```
2.0
```

In [25]:

```
b.cumsum(axis=1)
```

Out[25]:

```
array([[1., 6., 8.]])
```

In [26]:

```
b.max(axis=0)
```

Out[26]:

```
array([1., 5., 2.])
```

In [27]:

```
a.min()
```

Out[27]:

```
1.0
```

In [28]:

```
a<2
```

Out[28]:

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

In [29]:

```
a==b
```

Out[29]:

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

In [30]:

```
np.array_equal(a,b)
```

Out[30]:

```
False
```

In [34]:

```
np.int64
```

Out[34]:

```
numpy.int64
```

In [35]:

```
np.float32
```

Out[35]:

```
numpy.float32
```

In [36]:

```
np.complex
```

```
C:\Users\TEMP\AppData\Local\Temp/ipykernel_5252/82300521.py:1: Deprecation
Warning: `np.complex` is a deprecated alias for the builtin `complex`. To
silence this warning, use `complex` by itself. Doing this will not modify
any behavior and is safe. If you specifically wanted the numpy scalar typ
e, use `np.complex128` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.or
g/devdocs/release/1.20.0-notes.html#deprecations (https://numpy.org/devdoc
s/release/1.20.0-notes.html#deprecations)
  np.complex
```

Out[36]:

```
complex
```

In [37]:

```
np.bool
```

```
C:\Users\TEMP\AppData\Local\Temp/ipykernel_5252/3526611434.py:1: Deprecati
onWarning: `np.bool` is a deprecated alias for the builtin `bool`. To sile
nce this warning, use `bool` by itself. Doing this will not modify any beh
avior and is safe. If you specifically wanted the numpy scalar type, use `
np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.or
g/devdocs/release/1.20.0-notes.html#deprecations (https://numpy.org/devdoc
s/release/1.20.0-notes.html#deprecations)
  np.bool
```

Out[37]:

```
bool
```

In [38]:

```
a.sort()
```

In [39]:

```
a.shape
```

Out[39]:

```
(1, 3)
```

In [40]:

```
len(a)
```

Out[40]:

```
1
```

In [41]:

```
len(a)
```

Out[41]:

```
1
```

In [42]:

```
a.sum()
```

Out[42]:

```
6.0
```

In [43]:

```
g=a-b
```

In [45]:

```
np.add(b,a)
```

Out[45]:

```
array([[2., 7., 5.]])
```

In [46]:

```
a[a<2]
```

Out[46]:

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

In [51]:

```
b[:1]
```

Out[51]:

```
array([[1., 5., 2.]])
```

In [56]:

```
np.delete(a,[1,2])
```

Out[56]:

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

In [57]:

```
np.vstack((a,b))
```

Out[57]:

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

In [58]:

```
np.column_stack((a,b))
```

Out[58]:

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

In [59]:

```
a[a<2]
```

Out[59]:

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

In [61]:

```
a.max(axis=0)
```

Out[61]:

```
3.0
```

In [64]:

```
b.max(axis=0)
```

Out[64]:

```
5.0
```

In [65]:

```
b.max(axis=0)
```

Out[65]:

```
5.0
```

In [77]:

```
np.c_[a,b]
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
~\AppData\Local\Temp/ipykernel_5252/3583131449.py in <module>
----> 1 np.c_[a,b]

C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\index_tricks.py in __
getitem__(self, key)
    405                 objs[k] = objs[k].astype(final_dtype)
    406 
--> 407         res = self.concatenate(tuple(objs), axis=axis)
    408 
    409         if matrix:

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

ValueError: all the input array dimensions for the concatenation axis must
match exactly, but along dimension 0, the array at index 0 has size 3 and
the array at index 1 has size 2
```

In [ ]:

```

```