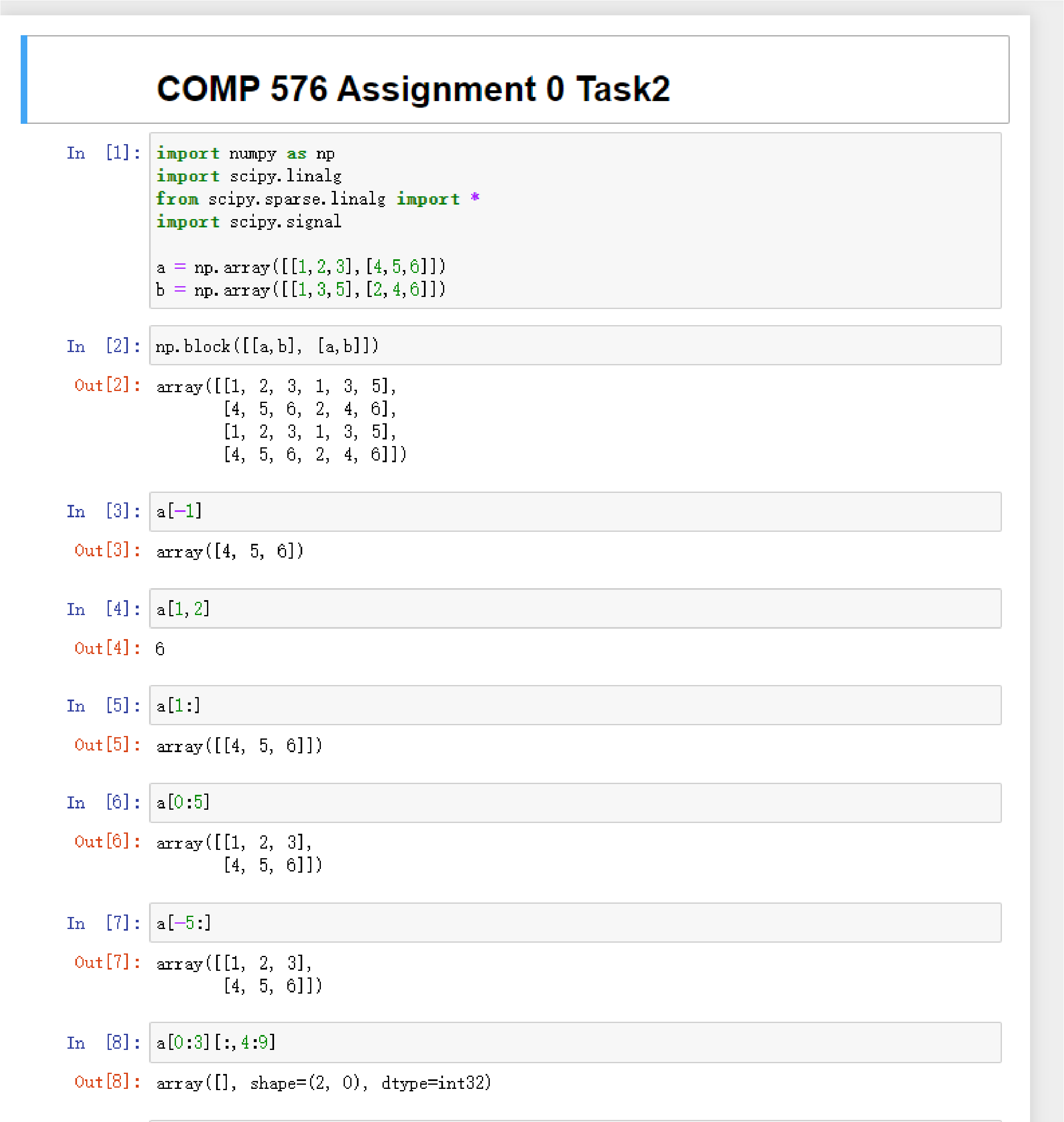
Task 1

```
C:\WINDOWS\system32>conda info

     active environment : None
       user config file : C:\Users\MingzLiu\.condarc
 populated config files :
          conda version : 4.10.3
    conda-build version : 3.21.4
         python version : 3.8.8.final.0
       virtual packages : __cuda=11.2=0
                          __win=0=0
                          __archspec=1=x86_64
       base environment : C:\Users\MingzLiu\anaconda3  (writable)
      conda av data dir : C:\Users\MingzLiu\anaconda3\etc\conda
  conda av metadata url : None
           channel URLs : https://repo.anaconda.com/pkgs/main/win-64
                          https://repo.anaconda.com/pkgs/main/noarch
                          https://repo.anaconda.com/pkgs/r/win-64
                          https://repo.anaconda.com/pkgs/r/noarch
                          https://repo.anaconda.com/pkgs/msys2/win-64
                          https://repo.anaconda.com/pkgs/msys2/noarch
          package cache : C:\Users\MingzLiu\anaconda3\pkgs
                          C:\Users\MingzLiu\.conda\pkgs
                          C:\Users\MingzLiu\AppData\Local\conda\conda\pkgs
       envs directories : C:\Users\MingzLiu\anaconda3\envs
                          C:\Users\MingzLiu\.conda\envs
                          C:\Users\MingzLiu\AppData\Local\conda\conda\envs
               platform : win-64
             user-agent : conda/4.10.3 requests/2.25.1 CPython/3.8.8 Windows/10
Windows/10.0.19041
          administrator : True
             netrc file : None
           offline mode : False
```

Task 2



COMP 576 Assignment 0 Task2

```
In  [1]: import numpy as np
         import scipy.linalg
         from scipy.sparse.linalg import *
         import scipy.signal

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

In  [2]: np.block([[a,b], [a,b]])

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

In  [3]: a[-1]

Out[3]: array([4, 5, 6])

In  [4]: a[1,2]

Out[4]: 6

In  [5]: a[1:]

Out[5]: array([[4, 5, 6]])

In  [6]: a[0:5]

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

In  [7]: a[-5:]

Out[7]: array([[1, 2, 3],
               [4, 5, 6]])

In  [8]: a[0:3][:,4:9]

Out[8]: array([], shape=(2, 0), dtype=int32)
```

```
In  [9]: a[np.ix_([1,1],[0,1])]
```

```
Out[9]: array([[4, 5],
               [4, 5]])
```

```
In  [10]: a[ 2:21:2,:]
```

```
Out[10]: array([], shape=(0, 3), dtype=int32)
```

```
In  [11]: a[ ::2,:]
```

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

```
In  [12]: a[ ::-1,:]
```

```
Out[12]: array([[4, 5, 6],
                [1, 2, 3]])
```

```
In  [13]: a[np.r_[:len(a),0]]
```

```
Out[13]: array([[1, 2, 3],
                [4, 5, 6],
                [1, 2, 3]])
```

```
In  [14]: a.T
```

```
Out[14]: array([[1, 4],
                [2, 5],
                [3, 6]])
```

```
In  [15]: a.conj().T
```

```
Out[15]: array([[1, 4],
                [2, 5],
                [3, 6]])
```

```
In  [16]: a.T @ b
```

```
Out[16]: array([[ 9, 19, 29],
                [12, 26, 40],
                [15, 33, 51]])
```

```
In  [17]: a * b
```

```
Out[17]: array([[ 1,  6, 15],
In [18]: c = a / b
         c

Out[18]: array([[1.        , 0.66666667, 0.6       ],
                [2.        , 1.25      , 1.        ]])

In [19]: a**3

Out[19]: array([[  1,   8,  27],
                [ 64, 125, 216]], dtype=int32)

In [20]: (a>0.5)

Out[20]: array([[ True,  True,  True],
                [ True,  True,  True]])

In [21]: np.nonzero(a>0.5)

Out[21]: (array([0, 0, 0, 1, 1, 1], dtype=int64),
          array([0, 1, 2, 0, 1, 2], dtype=int64))

In [22]: a[:,np.nonzero(a>0.5)[0]]

Out[22]: array([[1, 1, 1, 2, 2, 2],
                [4, 4, 4, 5, 5, 5]])

In [23]: a = np.array([[1,2,3],[4,5,6]])
         c= np.array([4,2,3])
         a[:,c.T>0.5]

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

In [24]: a[a<0.5]=0
         a

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

In [25]: a * (a>0.5)

Out[25]: array([[1, 2, 3],
                [4, 5, 6]])
```

```
In  [26]: a[:] = 3
          a
```

```
Out[26]: array([[3, 3, 3],
                [3, 3, 3]])
```

```
In  [27]: y = a.copy()
          y
```

```
Out[27]: array([[3, 3, 3],
                [3, 3, 3]])
```

```
In  [28]: y = a[1,:].copy()
          y
```

```
Out[28]: array([3, 3, 3])
```

```
In  [29]: y = a.flatten()
          y
```

```
Out[29]: array([3, 3, 3, 3, 3, 3])
```

```
In  [30]: np.arange(1.,11.)
```

```
Out[30]: array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])
```

```
In  [31]: np.arange(10.)
```

```
Out[31]: array([0., 1., 2., 3., 4., 5., 6., 7., 8., 9.])
```

```
In  [32]: np.arange(1.,11.)[:, np.newaxis]
```

```
Out[32]: array([[ 1.],
                [ 2.],
                [ 3.],
                [ 4.],
                [ 5.],
                [ 6.],
                [ 7.],
                [ 8.],
                [ 9.],
                [10.]])
```

```
In  [33]: np.zeros((3,4))

Out[33]: array([[0., 0., 0., 0.],
                [0., 0., 0., 0.],
                [0., 0., 0., 0.]])
```

```
In  [34]: np.zeros((3,4,5))

Out[34]: array([[[0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.]],

                [[0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.]],

                [[0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.]]])
```

```
In  [35]: np.ones((3,4))

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

```
In  [36]: np.eye(3)

Out[36]: array([[1., 0., 0.],
                [0., 1., 0.],
                [0., 0., 1.]])
```

```
In  [37]: np.diag(a)

Out[37]: array([3, 3])
```

```
In  [38]: np.diag(a,0)

Out[38]: array([3, 3])
```

```
In  [39]: np.random.rand(3,4)
```

```
Out[39]: array([[0.00421695, 0.86073371, 0.15351978, 0.98562724],
                [0.73717565, 0.21474524, 0.54830296, 0.26126506],
                [0.43549931, 0.74617176, 0.86455733, 0.02154422]])
```

```
In  [40]: np.linspace(1,3,4)
```

```
Out[40]: array([1.        , 1.66666667, 2.33333333, 3.        ])
```

```
In  [41]: np.mgrid[0:9.,0:6.]
```

```
Out[41]: array([[[0., 0., 0., 0., 0., 0.],
                 [1., 1., 1., 1., 1., 1.],
                 [2., 2., 2., 2., 2., 2.],
                 [3., 3., 3., 3., 3., 3.],
                 [4., 4., 4., 4., 4., 4.],
                 [5., 5., 5., 5., 5., 5.],
                 [6., 6., 6., 6., 6., 6.],
                 [7., 7., 7., 7., 7., 7.],
                 [8., 8., 8., 8., 8., 8.]],

                [[0., 1., 2., 3., 4., 5.],
                 [0., 1., 2., 3., 4., 5.],
                 [0., 1., 2., 3., 4., 5.],
                 [0., 1., 2., 3., 4., 5.],
                 [0., 1., 2., 3., 4., 5.],
                 [0., 1., 2., 3., 4., 5.],
                 [0., 1., 2., 3., 4., 5.],
                 [0., 1., 2., 3., 4., 5.],
                 [0., 1., 2., 3., 4., 5.]]])
```

```
In  [42]: np.ogrid[0:9.,0:6.]
```

```
Out[42]: [array([[0.],
                 [1.],
                 [2.],
                 [3.],
                 [4.],
                 [5.],
                 [6.],
                 [7.],
                 [8.]]),
          array([[0., 1., 2., 3., 4., 5.]])]
```

```
In  [43]: np.meshgrid([1,2,4],[2,4,5])

Out[43]: [array([[1, 2, 4],
                 [1, 2, 4],
                 [1, 2, 4]]),
          array([[2, 2, 2],
                 [4, 4, 4],
                 [5, 5, 5]])]
```

```
In  [44]: np.ix_([1,2,4],[2,4,5])

Out[44]: (array([[1],
                 [2],
                 [4]]),
          array([[2, 4, 5]]))
```

```
In  [45]: np.tile(a, (3, 2))

Out[45]: array([[3, 3, 3, 3, 3, 3],
                [3, 3, 3, 3, 3, 3],
                [3, 3, 3, 3, 3, 3],
                [3, 3, 3, 3, 3, 3],
                [3, 3, 3, 3, 3, 3],
                [3, 3, 3, 3, 3, 3]])
```

```
In  [46]: np.concatenate((a,b),1)

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

```
In  [47]: np.vstack((a,b))

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

```
In  [48]: a.max()

Out[48]: 3
```

```
In  [49]: a.max(0)

Out[49]: array([3, 3, 3])
```

```
In  [50]: a.max(1)

Out[50]: array([3, 3])
```

```
In  [51]: np.maximum(a, b)

Out[51]: array([[3, 3, 5],
                [3, 4, 6]])
```

```
In  [52]: v = np.array([1,2,3,4,5])
```

```
In  [53]: np.linalg.norm(v)

Out[53]: 7.416198487095663
```

```
In  [54]: np.logical_and(a,b)

Out[54]: array([[ True,  True,  True],
                [ True,  True,  True]])
```

```
In  [55]: np.logical_or(a,b)

Out[55]: array([[ True,  True,  True],
                [ True,  True,  True]])
```

```
In  [56]: a & b

Out[56]: array([[1, 3, 1],
                [2, 0, 2]], dtype=int32)
```

```
In  [57]: a | b

Out[57]: array([[3, 3, 7],
                [3, 7, 7]], dtype=int32)
```

```
In  [58]: square = abs(np.random.rand(2,2))
```

```
In  [59]: np.linalg.inv(square)

Out[59]: array([[ 5.85258018, -4.46091315],
                [-7.5709798 ,  9.04771294]])
```

```
In  [60]: np.linalg.pinv(a)

Out[60]: array([[0.05555556, 0.05555556],
                [0.05555556, 0.05555556],
                [0.05555556, 0.05555556]])
```

```
In  [61]: np.linalg.matrix_rank(a)

Out[61]: 1
```

```
In  [62]: np.linalg.solve(square, square.T)

Out[62]: array([[ 1.72338235,  0.94905529],
                [-1.46717851, -0.22771124]])
```

```
In  [63]: np.linalg.svd(square)

Out[63]: (array([[-0.72586763, -0.68783441],
                 [-0.68783441,  0.72586763]]),
          array([0.72137082, 0.07227962]),
          array([[-0.85109418, -0.52501305],
                 [-0.52501305,  0.85109418]]))
```

```
In  [64]: c = np.linalg.cholesky(square@square.T)
          square
          c

Out[64]: array([[0.52597466, 0.        ],
                [0.48900296, 0.09913103]])
```

```
In  [65]: np.linalg.eig(square)

Out[65]: (array([0.70270816, 0.07419924]),
          array([[ 0.70959969, -0.5049857 ],
                 [ 0.70460506,  0.86312771]]))
```

```
In  [66]: scipy.linalg.eig(square,square)

Out[66]: (array([1.+0.j, 1.+0.j]),
          array([[-1.        , -0.4472136 ],
                 [-0.        ,  0.89442719]]))
```

In [67]:

```
scipy.sparse.linalg.eigs(square,k=3)
```

```
C:\Users\MingzLiu\anaconda3\lib\site-packages\scipy\sparse\linalg\eigen\arpack\arpack.p
y:1266: RuntimeWarning: k >= N - 1 for N * N square matrix. Attempting to use scipy.lin
alg.eig instead.
  warnings.warn("k >= N - 1 for N * N square matrix. "
```

Out[67]:

```
(array([0.70270816+0.j, 0.07419924+0.j]),
 array([[ 0.70959969, -0.5049857 ],
        [ 0.70460506,  0.86312771]]))
```

In [68]:

```
np.linalg.qr(a)
```

Out[68]:

```
(array([[-0.70710678, -0.70710678],
        [-0.70710678,  0.70710678]]),
 array([[-4.24264069e+00, -4.24264069e+00, -4.24264069e+00],
        [ 0.00000000e+00, -1.08176281e-16, -1.08176281e-16]]))
```

In [69]:

```
scipy.linalg.lu(a)
```

Out[69]:

```
(array([[1., 0.],
        [0., 1.]]),
 array([[1., 0.],
        [1., 1.]]),
 array([[3., 3., 3.],
        [0., 0., 0.]]))
```

In [70]:

```
np.fft.fft(a)
```

Out[70]:

```
array([[9.+0.j, 0.+0.j, 0.+0.j],
       [9.+0.j, 0.+0.j, 0.+0.j]])
```

In [71]:

```
np.fft.ifft(a)
```

Out[71]:

```
array([[3.+0.j, 0.+0.j, 0.+0.j],
       [3.+0.j, 0.+0.j, 0.+0.j]])
```

In [72]:

```
np.sort(a)
```

Out[72]:

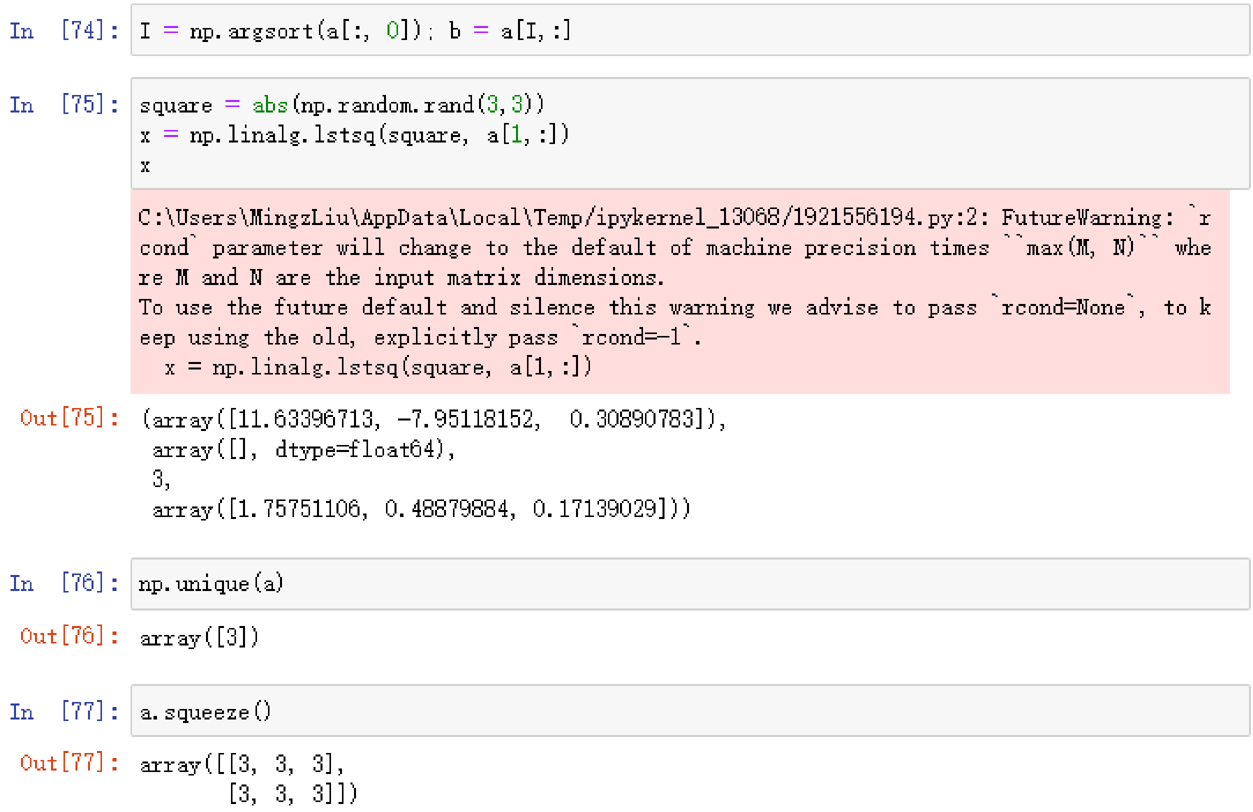
```
array([[3, 3, 3],
       [3, 3, 3]])
```

In [73]:

```
np.sort(a, axis = 1)
```

Out[73]:

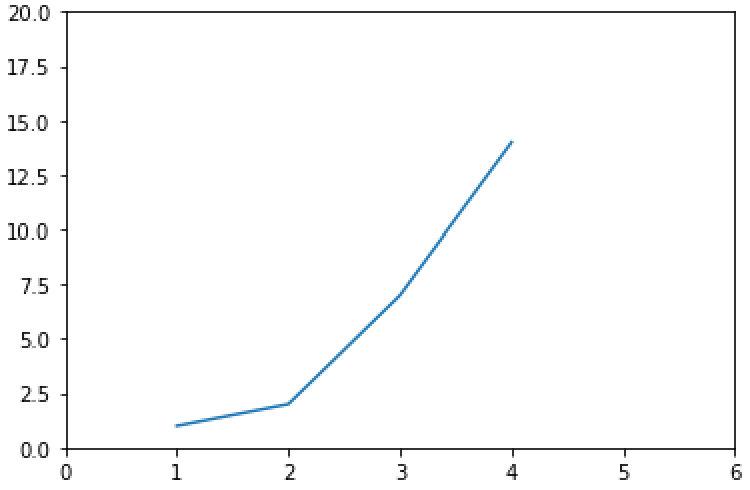
```
array([[3, 3, 3],
       [3, 3, 3]])
```



```
In [74]: I = np.argsort(a[:, 0]); b = a[I,:]

In [75]: square = abs(np.random.rand(3,3))
         x = np.linalg.lstsq(square, a[1,:])
         x

C:\Users\MingzLiu\AppData\Local\Temp/ipykernel_13068/1921556194.py:2: FutureWarning: `r
cond` parameter will change to the default of machine precision times ``max(M, N)`` whe
re M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=None`, to k
eep using the old, explicitly pass `rcond=-1`.
  x = np.linalg.lstsq(square, a[1,:])

Out[75]: (array([11.63396713, -7.95118152,  0.30890783]),
          array([], dtype=float64),
          3,
          array([1.75751106, 0.48879884, 0.17139029]))

In [76]: np.unique(a)

Out[76]: array([3])

In [77]: a.squeeze()

Out[77]: array([[3, 3, 3],
                [3, 3, 3]])
```

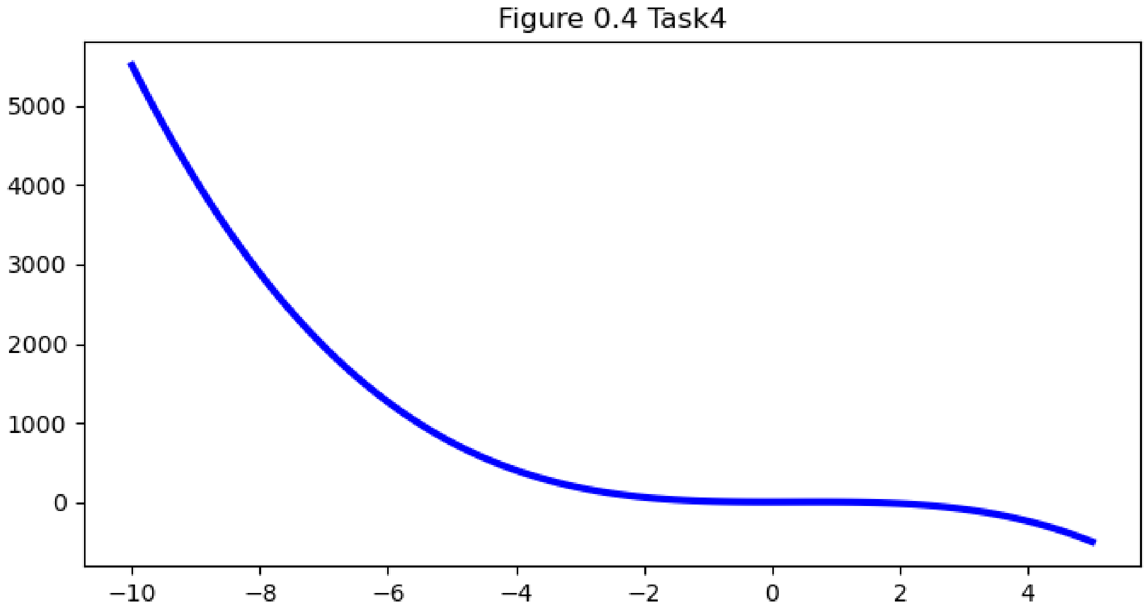
Task 3

import matplotlib.pyplot as plt
plt.plot([1,2,3,4], [1,2,7,14])
plt.axis([0, 6, 0, 20])
plt.show()



Task 4

```
import numpy as np
from matplotlib import pyplot as plt

x = np.linspace(-10, 5, 100)
y = -5 * x ** 3 + 5 * x ** 2 + 5
plt.figure(figsize=(8, 4))
plt.plot(x, y, color="blue", linewidth=3)
plt.title("Figure 0.4 Task4")
plt.show()
```



Task 5

Github account: MingzLiu

Task 6

https://github.com/MingzLiu/ELEC-576-intro2DL