Task 1

```
(base) C:\Users\He Yue>conda info

     active environment : base
    active env location : C:\Users\He Yue\Anaconda3
            shell level : 1
       user config file : C:\Users\He Yue\.condarc
 populated config files : C:\Users\He Yue\.condarc
          conda version : 4.6.14
    conda-build version : 3.15.1
         python version : 3.7.2.final.0
       base environment : C:\Users\He Yue\Anaconda3  (writable)
           channel URLs : https://repo.anaconda.com/pkgs/main/win-64
                          https://repo.anaconda.com/pkgs/main/noarch
                          https://repo.anaconda.com/pkgs/free/win-64
                          https://repo.anaconda.com/pkgs/free/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\He Yue\Anaconda3\pkgs
                          C:\Users\He Yue\.conda\pkgs
                          C:\Users\He Yue\AppData\Local\conda\conda\pkgs
       envs directories : C:\Users\He Yue\Anaconda3\envs
                          C:\Users\He Yue\.conda\envs
                          C:\Users\He Yue\AppData\Local\conda\conda\envs
                platform : win-64
            user-agent : conda/4.6.14 requests/2.21.0 CPython/3.7.2 Windows/10 Windows/10.0.18362
          administrator : False
             netrc file : None
           offline mode : False
```

Task 2

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

In [2]:
```
a = np.array([[i*j for j in range(5)] for i in range(5)])
a
```

Out[2]:
```
array([[ 0,  0,  0,  0,  0],
       [ 0,  1,  2,  3,  4],
       [ 0,  2,  4,  6,  8],
       [ 0,  3,  6,  9, 12],
       [ 0,  4,  8, 12, 16]])
```

In [3]:
```
ndim(a)
# a.ndim
```

Out[3]: 2

In [4]:
```
size(a)
# a.size
```

Out[4]: 25

In [5]:
```
shape(a)
# a.shape
```

Out[5]: (5, 5)

In [6]:
```
a.shape[1]
```

Out[6]: 5

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

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

In [8]:
```
block([[1,3], [4,6]])
```

Out[8]:
```
array([[1, 3],
       [4, 6]])
```

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

Out[9]:
```
array([ 0,  4,  8, 12, 16])
```

In [10]:
```
a[1,2]
```

Out[10]:
```
2
```

In [11]:
```
a[1]
# a[1,:]
```

Out[11]:
```
array([0, 1, 2, 3, 4])
```

In [12]:
```
a[0:1]
# a[:5]
# a[0:5,:]
```

Out[12]:
```
array([[0, 0, 0, 0, 0]])
```

In [13]:
```
a[-1:]
```

Out[13]:
```
array([[ 0,  4,  8, 12, 16]])
```

In [14]:
```
a[0:1][:,1:2]
```

Out[14]:
```
array([[0]])
```

In [15]:
```
a[ix_([1,3,4],[0,2])]
```

Out[15]:
```
array([[0, 2],
       [0, 6],
       [0, 8]])
```

In [16]:
```
a[ 2:21:2,:]
```

Out[16]:
```
array([[ 0,  2,  4,  6,  8],
       [ 0,  4,  8, 12, 16]])
```

In [17]:
```
a[ ::2,:]
```

Out[17]:
```
array([[ 0,  0,  0,  0,  0],
       [ 0,  2,  4,  6,  8],
       [ 0,  4,  8, 12, 16]])
```

In [18]:
```
a[ ::-1,:]
```

Out[18]:
```
array([[ 0,  4,  8, 12, 16],
       [ 0,  3,  6,  9, 12],
       [ 0,  2,  4,  6,  8],
       [ 0,  1,  2,  3,  4],
       [ 0,  0,  0,  0,  0]])
```

In [19]:
```
a[r_[:len(a),0]]
```

Out[19]:
```
array([[ 0,  0,  0,  0,  0],
       [ 0,  1,  2,  3,  4],
       [ 0,  2,  4,  6,  8],
       [ 0,  3,  6,  9, 12],
       [ 0,  4,  8, 12, 16],
       [ 0,  0,  0,  0,  0]])
```

In [20]:
```
a.transpose()
# a.T
```

Out[20]:
```
array([[ 0,  0,  0,  0,  0],
       [ 0,  1,  2,  3,  4],
       [ 0,  2,  4,  6,  8],
       [ 0,  3,  6,  9, 12],
       [ 0,  4,  8, 12, 16]])
```

In [21]:
```
a.conj().transpose()
# a.conj().T
```

Out[21]:
```
array([[ 0,  0,  0,  0,  0],
       [ 0,  1,  2,  3,  4],
       [ 0,  2,  4,  6,  8],
       [ 0,  3,  6,  9, 12],
       [ 0,  4,  8, 12, 16]])
```

In [22]:
```
b = a.T
a @ b
```

Out[22]:
```
array([[  0,   0,   0,   0,   0],
       [  0,  30,  60,  90, 120],
       [  0,  60, 120, 180, 240],
       [  0,  90, 180, 270, 360],
       [  0, 120, 240, 360, 480]])
```

In [23]:
```
a * b
```

Out[23]:
```
array([[  0,   0,   0,   0,   0],
       [  0,   1,   4,   9,  16],
       [  0,   4,  16,  36,  64],
       [  0,   9,  36,  81, 144],
       [  0,  16,  64, 144, 256]])
```

In [24]:
```
a / b
```

```
C:\Users\He Yue\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: RuntimeWarning: in
valid value encountered in true_divide
  """Entry point for launching an IPython kernel.
```

Out[24]:
```
array([[nan, nan, nan, nan, nan],
       [nan,  1.,  1.,  1.,  1.],
       [nan,  1.,  1.,  1.,  1.],
       [nan,  1.,  1.,  1.,  1.],
       [nan,  1.,  1.,  1.,  1.]])
```

In [25]:
```
a ** 3
```

Out[25]:
```
array([[   0,    0,    0,    0,    0],
       [   0,    1,    8,   27,   64],
       [   0,    8,   64,  216,  512],
       [   0,   27,  216,  729, 1728],
       [   0,   64,  512, 1728, 4096]], dtype=int32)
```

In [26]:
```
(a>0.5)
```

Out[26]:
```
array([[False, False, False, False, False],
       [False,  True,  True,  True,  True],
       [False,  True,  True,  True,  True],
       [False,  True,  True,  True,  True],
       [False,  True,  True,  True,  True]])
```

In [27]:
```
nonzero(a>0.5)
```

Out[27]:
```
(array([1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4], dtype=int64),
 array([1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4], dtype=int64))
```

In [28]:
```
v= np.array([1,0.3,0,-1,5])
a[:,nonzero(v>0.5)[0]]
```

Out[28]:
```
array([[ 0,  0],
       [ 0,  4],
       [ 0,  8],
       [ 0, 12],
       [ 0, 16]])
```

In [29]:
```
a[:,v.T>0.5]
```

Out[29]:
```
array([[ 0,  0],
       [ 0,  4],
       [ 0,  8],
       [ 0, 12],
       [ 0, 16]])
```

In [30]:
```
a[a<0.5]=0
a
```

Out[30]:
```
array([[ 0,  0,  0,  0,  0],
       [ 0,  1,  2,  3,  4],
       [ 0,  2,  4,  6,  8],
       [ 0,  3,  6,  9, 12],
       [ 0,  4,  8, 12, 16]])
```

In [31]:
```
a * (a>0.5)
```

Out[31]:
```
array([[ 0,  0,  0,  0,  0],
       [ 0,  1,  2,  3,  4],
       [ 0,  2,  4,  6,  8],
       [ 0,  3,  6,  9, 12],
       [ 0,  4,  8, 12, 16]])
```

In [32]:
```
a[:] = 3
a
```

Out[32]:
```
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]])
```

In [33]:
```
x = a.copy()
x
```

Out[33]:
```
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]])
```

In [34]:
```
y = x.flatten()
y
```

Out[34]:
```
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])
```

In [35]:
```
arange(1.,11.)
# r_[1.:11.]
# r_[1:10:10j]
```

Out[35]:
```
array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])
```

In [36]:
```
arange(10.)
# r_[:10.]
# r_[:9:10j]
```

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

In [37]:
```
arange(1,3)[:, newaxis]
```

Out[37]:
```
array([[1],
       [2]])
```

In [38]:
```
zeros((3,4))
```

Out[38]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

In [39]:
```
zeros((3,4,5))
```

Out[39]:
```
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 [40]:
```
ones((3,4))
```

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

In [41]:
```
eye(3)
```

Out[41]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [42]:
```
a = np.array([[i + j for i in range(5)] for j in range(5)])
diag(a)
```

Out[42]:
```
array([0, 2, 4, 6, 8])
```

In [43]:
```
diag(a,0)
```

Out[43]:
```
array([0, 2, 4, 6, 8])
```

In [44]:
```
random.rand(3,4)
# random.random_sample((3, 4))
```

Out[44]:
```
array([[0.6349406 , 0.66862164, 0.90967505, 0.53673325],
       [0.04493657, 0.02470661, 0.96364489, 0.29862059],
       [0.89265823, 0.83582281, 0.62901544, 0.42067588]])
```

In [45]:
```
linspace(1,3,4)
```

Out[45]:
```
array([1.        , 1.66666667, 2.33333333, 3.        ])
```

In [46]:
```
mgrid[0:9.,0:6.]
# meshgrid(r_[0:9.],r_[0:6.]
```

Out[46]:
```
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 [47]:
```
ogrid[0:9.,0:6.]
# ix_(r_[0:9.],r_[0:6.]
```

Out[47]:
```
[array([[0.],
        [1.],
        [2.],
        [3.],
        [4.],
        [5.],
        [6.],
        [7.],
        [8.]]), array([[0., 1., 2., 3., 4., 5.]])]
```

In [48]:
```
meshgrid([1,2,4],[2,4,5])
```

Out[48]:
```
[array([[1, 2, 4],
        [1, 2, 4],
        [1, 2, 4]]), array([[2, 2, 2],
        [4, 4, 4],
        [5, 5, 5]])]
```

In [49]:
```
ix_([1,2,4],[2,4,5])
```

Out[49]:
```
(array([[1],
        [2],
        [4]]), array([[2, 4, 5]]))
```

In [50]:
```
tile(a, (2, 3))
```

Out[50]:
```
array([[0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4],
       [1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5],
       [2, 3, 4, 5, 6, 2, 3, 4, 5, 6, 2, 3, 4, 5, 6],
       [3, 4, 5, 6, 7, 3, 4, 5, 6, 7, 3, 4, 5, 6, 7],
       [4, 5, 6, 7, 8, 4, 5, 6, 7, 8, 4, 5, 6, 7, 8],
       [0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4],
       [1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5],
       [2, 3, 4, 5, 6, 2, 3, 4, 5, 6, 2, 3, 4, 5, 6],
       [3, 4, 5, 6, 7, 3, 4, 5, 6, 7, 3, 4, 5, 6, 7],
       [4, 5, 6, 7, 8, 4, 5, 6, 7, 8, 4, 5, 6, 7, 8]])
```

In [51]:
```
concatenate((a,b),1)
# hstack((a,b))
# column_stack((a,b)) or c_[a,b]
```

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

In [52]:
```
concatenate((a,b))
# vstack((a,b))
# r_[a,b]
```

Out[52]:
```
array([[0, 1, 2, 3, 4],
       [1, 2, 3, 4, 5],
       [2, 3, 4, 5, 6],
       [3, 4, 5, 6, 7],
       [4, 5, 6, 7, 8],
       [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 [53]:
```
a = np.array([[-1,0,3,4,5],[-2,-3,4,5,6]])
a.max()
```

Out[53]: 6

In [54]:
```
a.max(0)
```

Out[54]:
```
array([-1,  0,  4,  5,  6])
```

In [55]:
```
a.max(1)
```

Out[55]:
```
array([5, 6])
```

In [56]:
```
b = [-1,5,0,7,8]
maximum(a, b)
```

Out[56]:
```
array([[-1,  5,  3,  7,  8],
       [-1,  5,  4,  7,  8]])
```

In [57]:
```
sqrt(v @ v)
# np.linalg.norm(v)
```

Out[57]: 5.204805471869242

In [58]:
```
logical_and(a,b)
```

Out[58]:
```
array([[ True, False, False,  True,  True],
       [ True,  True, False,  True,  True]])
```

In [59]:
```
logical_or(a,b)
```

Out[59]:
```
array([[ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True]])
```

In [60]:
```
a = 10
b = 20
a & b
```

Out[60]: 0

In [61]:
```
a | b
```

Out[61]: 30

In [62]:
```
a = np.array([[2,3], [2,4]])
linalg.inv(a)
```

Out[62]:
```
array([[ 2. , -1.5],
       [-1. ,  1. ]])
```

In [63]:
```
linalg.pinv(a**2)
```

Out[63]:
```
array([[ 0.57142857, -0.32142857],
       [-0.14285714,  0.14285714]])
```

In [64]:
```
linalg.matrix_rank(a)
```

Out[64]:
```
2
```

In [65]:
```
b = np.array([4,8])
linalg.solve(a,b)
# linalg.lstsq(a,b)
```

Out[65]:
```
array([-4.,  4.])
```

In [66]:
```
U, S, Vh = linalg.svd(a)
V = Vh.T
print(U,S,Vh,Vh.transpose())
```

```
[[-0.62701799 -0.77900477]
 [-0.77900477  0.62701799]] [5.73396367 0.34879886] [[-0.49041914 -0.8714867 ]
 [-0.8714867   0.49041914]] [[-0.49041914 -0.8714867 ]
 [-0.8714867   0.49041914]]
```

In [67]:
```
a = np.array([[2, -1, 0], [-1, 2, -1], [0, -1, 2]])
linalg.cholesky(a).transpose()
```

Out[67]:
```
array([[ 1.41421356, -0.70710678,  0.        ],
       [ 0.        ,  1.22474487, -0.81649658],
       [ 0.        ,  0.        ,  1.15470054]])
```

In [68]:
```
D, V = linalg.eig(a)
print(D,V)
```

```
[3.41421356 2.         0.58578644] [[-5.00000000e-01 -7.07106781e-01  5.00000000e-01]
 [ 7.07106781e-01  4.05405432e-16  7.07106781e-01]
 [-5.00000000e-01  7.07106781e-01  5.00000000e-01]]
```

In [69]:
```
Q, R = scipy.linalg.qr(a)
print(Q,R)
```

```
[[-0.89442719 -0.35856858  0.26726124]
 [ 0.4472136  -0.71713717  0.53452248]
 [-0.          0.5976143   0.80178373]] [[-2.23606798  1.78885438 -0.4472136 ]
 [ 0.         -1.67332005  1.91236577]
 [ 0.          0.          1.06904497]]
```

In [70]:
```
a = np.array([[2, -1, 0], [-1, 2, -1], [0, -1, 2]])
scipy.linalg.lu(a)
```

Out[70]:
```
(array([[1., 0., 0.],
        [0., 1., 0.],
        [0., 0., 1.]]), array([[ 1.        ,  0.        ,  0.        ],
        [-0.5       ,  1.        ,  0.        ],
        [ 0.        , -0.66666667,  1.        ]]), array([[ 2.        , -1.        ,
0.        ],
        [ 0.        ,  1.5       , -1.        ],
        [ 0.        ,  0.        ,  1.33333333]]))
```

In [71]:
```
import scipy.sparse.linalg as spla
from scipy.fftpack import fft, ifft
spla.cg
```

Out[71]:
```
<function scipy.sparse.linalg.isolve.iterative.cg(A, b, x0=None, tol=1e-05, maxiter=Non
e, M=None, callback=None, atol=None)>
```

In [72]:
```
fft(a)
```

Out[72]:
```
array([[ 1. +0.j        ,  2.5+0.8660254j ,  2.5-0.8660254j ],
       [ 0. +0.j        , -1.5-2.59807621j, -1.5+2.59807621j],
       [ 1. +0.j        , -0.5+2.59807621j, -0.5-2.59807621j]])
```

In [73]:
```
ifft(a)
```

Out[73]:
```
array([[ 0.33333333+0.j        ,  0.83333333-0.28867513j,
         0.83333333+0.28867513j],
       [ 0.        +0.j        , -0.5       +0.8660254j ,
        -0.5       -0.8660254j ],
       [ 0.33333333+0.j        , -0.16666667-0.8660254j ,
        -0.16666667+0.8660254j ]])
```

In [74]:
```
sort(a)
```

Out[74]:
```
array([[-1,  0,  2],
       [-1, -1,  2],
       [-1,  0,  2]])
```

In [75]:
```
I = argsort(a[:,:])
b = a[I,:]
print(b)
```

```
[[[-1  2 -1]
  [ 0 -1  2]
  [ 2 -1  0]]

 [[ 2 -1  0]
  [ 0 -1  2]
  [-1  2 -1]]

 [[-1  2 -1]
  [ 2 -1  0]
  [ 0 -1  2]]]
```

In [76]:
```
x = np.array([1,2,3,4])
y = np.array([3,4,5,6])
A = np.vstack([x, np.ones(len(x))]).T
np.linalg.lstsq(A, y)
```

```
C:\Users\He Yue\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: FutureWarning: `rc
ond` parameter will change to the default of machine precision times ``max(M, N)`` wher
e 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`.
  after removing the cwd from sys.path.
```

Out[76]: (array([1., 2.]), array([6.92612406e-31]), 2, array([5.77937881, 0.77380911]))

In [77]:
```
import scipy.signal
scipy.signal.resample(y, 3)
```

```
C:\Users\He Yue\Anaconda3\lib\site-packages\scipy\signal\signaltools.py:2223: FutureWar
ning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr
[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array
index, `arr[np.array(seq)]`, which will result either in an error or a different resul
t.
  Y[sl] = X[sl]
C:\Users\He Yue\Anaconda3\lib\site-packages\scipy\signal\signaltools.py:2225: FutureWar
ning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr
[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array
index, `arr[np.array(seq)]`, which will result either in an error or a different resul
t.
  Y[sl] = X[sl]
```

Out[77]: array([3.5   , 4.1339746, 5.8660254])

In [78]:
```
unique(a)
```

Out[78]: array([-1, 0, 2])

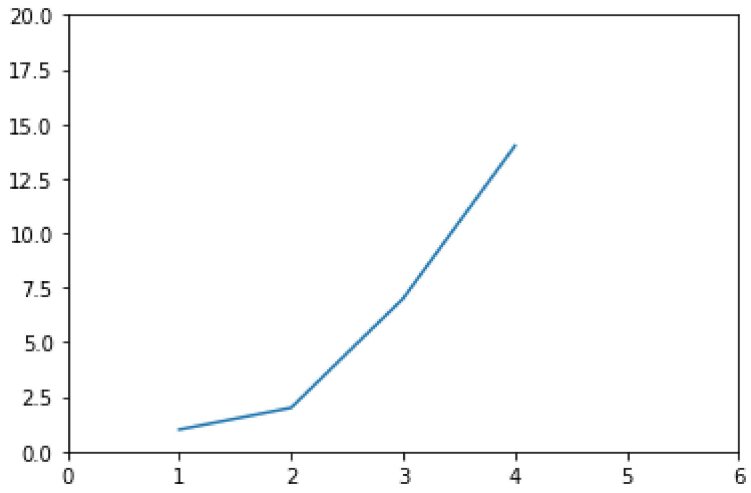
In [79]:
```
a.squeeze()
```

Out[79]:
```
array([[ 2, -1,  0],
       [-1,  2, -1],
       [ 0, -1,  2]])
```

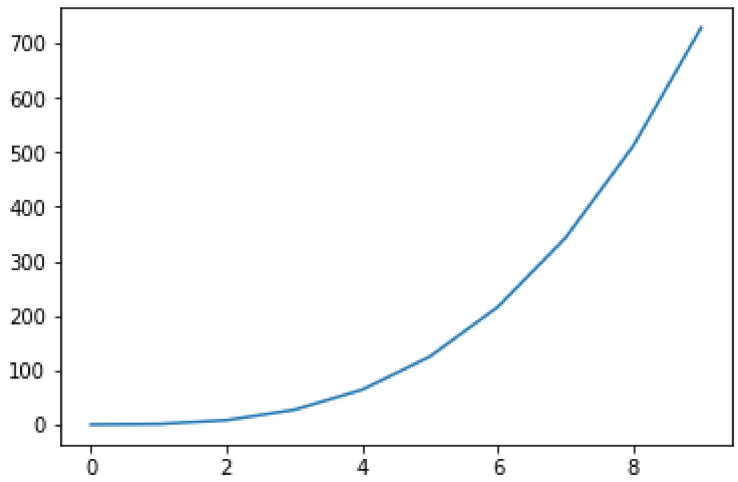
Task 3

In [80]:
```
import matplotlib.pyplot as plt
```

In [81]:
```
plt.plot([1, 2, 3, 4], [1, 2, 7, 14])
plt.axis([0, 6, 0, 20])
plt.show()
```



In [82]:
```
x = np.array([i for i in range(10)])
y = x ** 3
plt.plot(x,y)
plt.show()
```



Task 4

https://github.com/YueHeeeee

Task 5

https://github.com/YueHeeeee/COMP576-A0