In [43]:
```
import numpy as np

A = np.random.rand(3,4)
```

In [44]:
```
A
```

Out[44]:
```
array([[0.99209272, 0.1338347 , 0.94360135, 0.92016352],
       [0.23245756, 0.21893059, 0.64071962, 0.99633501],
       [0.08745165, 0.68346294, 0.56466501, 0.8576714 ]])
```

In [45]:
```
a = A[0:3,0:3]
print(a)
```

```
[[0.99209272 0.1338347  0.94360135]
 [0.23245756 0.21893059 0.64071962]
 [0.08745165 0.68346294 0.56466501]]
```

In [46]:
```
b = A[:,3]
print(b)
```

```
[0.92016352 0.99633501 0.8576714 ]
```

In [9]:
```
n = a.shape[0]
```

Out[9]:
```
3
```

In [11]:
```
n = len(a)
```

In [13]:
```
aa = a.copy()
```

In [14]:
```
aa
```

Out[14]:
```
array([[0.25479862, 0.6736852 , 0.30628764],
       [0.18486258, 0.28355278, 0.89021311],
       [0.61537523, 0.19774345, 0.46385214]])
```

In [15]:
```
a1 = [1,2,3]
```

In [16]:
```
b1 = a1
```

In [18]:
```
a1
```

Out[18]:
```
[1, 2, 3]
```

In [17]:
```
b1
```

Out[17]:
```
[1, 2, 3]
```

In [19]:
```
a1.append(2)
```

In [20]:
```
a1
```

Out[20]:
```
[1, 2, 3, 2]
```

In [21]:
```
b1
```

Out[21]:
```
[1, 2, 3, 2]
```

In [78]:
```
a = np.array([[1.0, 1.0, 1.0],
              [2.0, 1.0, 2.0],
              [3.0, 1.0, 4.0 ]])

n = 3

print(a)
```

```
[[1. 1. 1.]
 [2. 1. 2.]
 [3. 1. 4.]]
```

In [72]:
```
import numpy as np

A = np.random.rand(3,4)
a = A[0:3,0:3]
aa = a.copy()

b = A[:,3]
bb = b.copy()
x = np.zeros(len(b))

n = len(a)
```

In [84]:
```
# Elimination phase
for k in range(0,n-1):      # pivot rows  k = 0, 1, ...
    for i in range(k+1,n):   # range i = 1
        if a[i,k] != 0.0 :
#            print('We are in the loop',i,k)
            lam = a[i,k]/a[k,k]  # Definion of lambda
            a[i,k:n] = a[i,k:n]  - lam*a[k,k:n]    # a[i,k+1:n] 까지 변경
            b[i] = b[i] - lam*b[k]

# Back substitution
for k in range(n-1,-1,-1):  # From end to beginning
    x[k] = ( b[k] - np.dot(a[k,k+1:n],x[k+1:n]) )/a[k,k]  # j = k+1,
 ..., n

print(a)
```

```
[[ 0.84359357  0.56844361  0.37247176]
 [ 0.         -0.53364888  0.05873415]
 [ 0.          0.          0.55307934]]
```

In [ ]:
```
```

In [86]:
```
import numpy as np

A = np.random.rand(3,4)
a = A[0:3,0:3]
aa = a.copy()

b = A[:,3]
bb = b.copy()
x = np.zeros(len(b))

n = len(a)


# Elimination phase
#for k in range(0,n-1):     # pivot rows  k = 0, 1, ... n-1(미포함)
for k in range(n-1,0,-1):     # pivot rows  k = n-1, n-2, ..., 0(미포함)
    for i in range(k-1,-1,-1):   # range i = 1
        if a[i,k] != 0.0 :
#            print('We are in the loop',i,k)
            lam = a[i,k]/a[k,k]  # Definion of lambda
            a[i,k:n] = a[i,k:n]  - lam*a[k,k:n]    # a[i,k+1:n] 까지 변경
            b[i] = b[i] - lam*b[k]

# Back substitution
#for k in range(n-1,-1,-1):  # From end to beginning
#    x[k] = ( b[k] - np.dot(a[k,k+1:n],x[k+1:n]) )/a[k,k]  # j = k+1,
 ..., n

print(a)
```

```
[[0.21935306 0.         0.        ]
 [0.1683095  0.93212672 0.        ]
 [0.16076473 0.65833782 0.43662564]]
```

In [97]:
```
b1 = [1,2,3]
b2 = [2,3,4]
bmat = np.array([[1, 2],
                 [2, 3],
                 [3, 4]])

bmat1 = np.array([b1,b2])
bmat2 = np.vstack((b1,b2))
print(bmat1,'\n \n',bmat2)
#gauss(a,b)

bt1 = bmat1.T
bt2 = bmat2.T
print('\n\n',bt1,'\n \n',bt2)
```

```
[[1 2 3]
 [2 3 4]]

 [[1 2 3]
 [2 3 4]]


  [[1 2]
 [2 3]
 [3 4]]

 [[1 2]
 [2 3]
 [3 4]]
```

In [ ]:
```
# gauss 루틴의 b를 행렬 형태로 받을 수 있게 코드를 바꾸시오
```

In [101]:
```
bt1.shape[1]
```

Out[101]: 2

In [76]:
```
def gauss(a,b):
# Elimination phase
    for k in range(0,n-1):      # pivot rows  k = 0, 1, ...
        for i in range(k+1,n):   # range i = 1
            if a[i,k] != 0.0 :
#             print('We are in the loop',i,k)
                lam = a[i,k]/a[k,k]  # Definion of lambda
                a[i,k:n] = a[i,k:n]  - lam*a[k,k:n]    # a[i,k+1:n] 까지
 변경
                b[i] = b[i] - lam*b[k]

# Back substitution
    for k in range(n-1,-1,-1):  # From end to beginning
        x[k] = ( b[k] - np.dot(a[k,k+1:n],x[k+1:n]) )/a[k,k]  # j = k+1,
..., n

    return x
```

실습문제:

가우스 소거법을 활용해서 하부삼각 행렬을 만들어 봅시다.

In [82]:
```
def gauss(a,b):
# Elimination phase
    for k in range(0,n-1):      # pivot rows  k = 0, 1, ...
        for i in range(k+1,n):   # range i = 1
            if a[i,k] != 0.0 :
#             print('We are in the loop',i,k)
                lam = a[i,k]/a[k,k]  # Definion of lambda
                a[i,k:n] = a[i,k:n]  - lam*a[k,k:n]    # a[i,k+1:n] 까지
 변경
                b[i] = b[i] - lam*b[k]

# Back substitution
    for k in range(n-1,-1,-1):  # From end to beginning
        x[k] = ( b[k] - np.dot(a[k,k+1:n],x[k+1:n]) )/a[k,k]  # j = k+1,
..., n

    return x

print(a)
```

```
[[0.20455832 0.18084286 0.08113489]
 [0.35986835 0.81683725 0.41742305]
 [0.03822878 0.60704263 0.34217049]]
```

In [80]:
```

```

```
[[ 1.  1.  1.]
 [ 0. -1.  0.]
 [ 0.  0.  1.]]
```

In [ ]:
```

```

In [77]:
```
gauss(aa,bb)
```

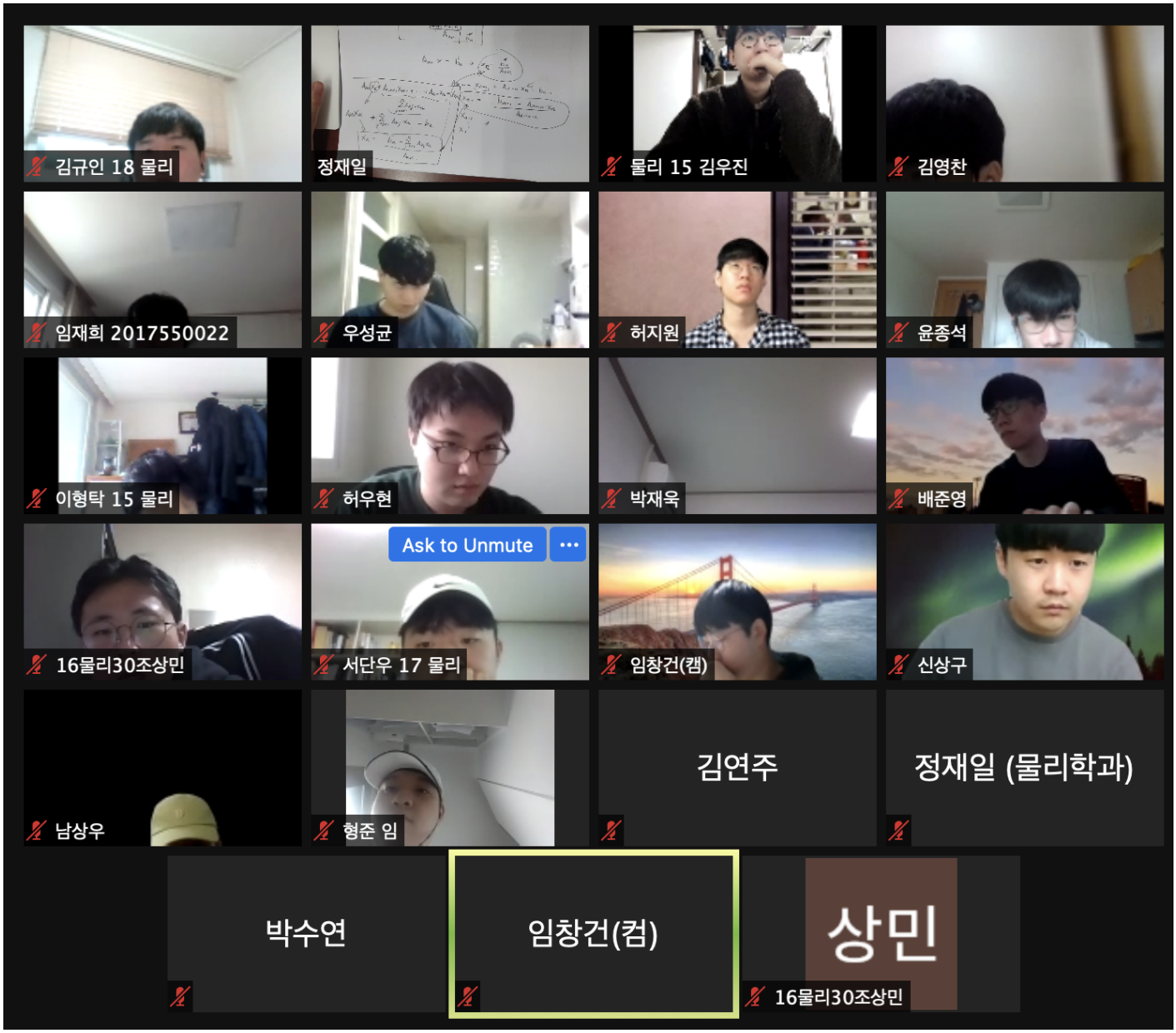
Out[77]:
```
array([ 0.44600605, -0.53072592,  3.71236635])
```

In [74]:
```
print('x ',x)
# BLAS
```

```
x  [ 0.44600605 -0.53072592  3.71236635]
```

In [75]:
```
np.linalg.solve(aa,bb)
```

Out[75]:
```
array([ 0.44600605, -0.53072592,  3.71236635])
```



In [ ]:
```
노캠: 박수연, 김연주
```

In [ ]:
```

```

In [67]:
```
aaa = [1,2,3,4,5]
aaa[0:5]
```

Out[67]:
```
[1, 2, 3, 4, 5]
```

In [60]:
```
print('Before\n ',aa,'\n \n','After \n',a)
print('\nBefore\n ',bb,'\n \n','After \n',b)
```

```
Before
  [[0.16705781 0.72904258 0.01733248]
 [0.05258849 0.79326455 0.19405686]
 [0.22199109 0.98170326 0.58701823]]

 After
 [[0.16705781 0.72904258 0.01733248]
 [0.         0.56376765 0.18860073]
 [0.         0.         0.55966043]]

Before
  [0.90863136 0.35358089 0.72538993]

 After
 [ 0.90863136  0.06755085 -0.48357426]
```

In [ ]:

In [ ]:

In [ ]:

In [27]:
```
import time
time_start = time.time()
A = np.zeros((4000,4000))

for i in range(4000):
    for j in range(4000):
        A[i,j] = A[i,j] + 1

time_finish = time.time()

print('Total elapsed time ',time_finish - time_start)
```

```
Total elapsed time  10.011560916900635
```

In [28]:
```
time_start = time.time()

A = np.zeros((4000,4000))
A = A + np.ones((4000,4000))

time_finish = time.time()

print('Total elapsed time ',time_finish - time_start)
```

```
Total elapsed time  0.11548590660095215
```

In [ ]:

In [ ]:

In [ ]: