Today I am writing Python codes in classroom. I first check if the following libraries are available on Anaconda.

In [1]:
```
import numpy as np
import pandas as pd
import sklearn as sl
import scipy as sp
import seaborn as sb
import matplotlib as mpl
from matplotlib import pyplot as plt
import oct2py as oclib
```

The oct2py is not available on Python. No worries! I will now install it :)

In [49]:
```
pip install mlxtend
```

```
Collecting mlxtend
  Downloading mlxtend-0.20.0-py2.py3-none-any.whl (1.3 MB)
     |████████████████████████████████| 1.3 MB 3.2 MB/s eta 0:00:01
Requirement already satisfied: setuptools in /Users/bicer/opt/anaconda3/lib/p
ython3.9/site-packages (from mlxtend) (58.0.4)
Requirement already satisfied: scipy>=1.2.1 in /Users/bicer/opt/anaconda3/li
b/python3.9/site-packages (from mlxtend) (1.7.1)
Requirement already satisfied: joblib>=0.13.2 in /Users/bicer/opt/anaconda3/l
ib/python3.9/site-packages (from mlxtend) (1.1.0)
Requirement already satisfied: pandas>=0.24.2 in /Users/bicer/opt/anaconda3/l
ib/python3.9/site-packages (from mlxtend) (1.3.4)
Requirement already satisfied: matplotlib>=3.0.0 in /Users/bicer/opt/anaconda
3/lib/python3.9/site-packages (from mlxtend) (3.4.3)
Requirement already satisfied: numpy>=1.16.2 in /Users/bicer/opt/anaconda3/li
b/python3.9/site-packages (from mlxtend) (1.20.3)
Collecting scikit-learn>=1.0.2
  Downloading scikit_learn-1.1.2-cp39-cp39-macosx_10_9_x86_64.whl (8.7 MB)
     |████████████████████████████████| 8.7 MB 6.5 MB/s eta 0:00:01
Requirement already satisfied: pillow>=6.2.0 in /Users/bicer/opt/anaconda3/li
b/python3.9/site-packages (from matplotlib>=3.0.0->mlxtend) (8.4.0)
Requirement already satisfied: python-dateutil>=2.7 in /Users/bicer/opt/anaco
nda3/lib/python3.9/site-packages (from matplotlib>=3.0.0->mlxtend) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/bicer/opt/anaconda
3/lib/python3.9/site-packages (from matplotlib>=3.0.0->mlxtend) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /Users/bicer/opt/anaconda3/li
b/python3.9/site-packages (from matplotlib>=3.0.0->mlxtend) (0.10.0)
Requirement already satisfied: pyparsing>=2.2.1 in /Users/bicer/opt/anaconda
3/lib/python3.9/site-packages (from matplotlib>=3.0.0->mlxtend) (3.0.4)
Requirement already satisfied: six in /Users/bicer/opt/anaconda3/lib/python
3.9/site-packages (from cycler>=0.10->matplotlib>=3.0.0->mlxtend) (1.16.0)
Requirement already satisfied: pytz>=2017.3 in /Users/bicer/opt/anaconda3/li
b/python3.9/site-packages (from pandas>=0.24.2->mlxtend) (2021.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/bicer/opt/anaco
nda3/lib/python3.9/site-packages (from scikit-learn>=1.0.2->mlxtend) (2.2.0)
Installing collected packages: scikit-learn, mlxtend
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 0.24.2
    Uninstalling scikit-learn-0.24.2:
      Successfully uninstalled scikit-learn-0.24.2
```

```
Successfully installed mlxtend-0.20.0 scikit-learn-1.1.2
Note: you may need to restart the kernel to use updated packages.
```

Python has some built-in functions: You can call them directly. They are not part of any library. See the list: https://docs.python.org/3/library/functions.html

In [2]:
```
year_list = [1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004
       2007, 2008, 2009, 2010, 2011, 2012, 2013]
```

In [3]:
```
type(year_list)
```

Out[3]:
```
list
```

In [4]:
```
min(year_list)
```

Out[4]:
```
1994
```

In [5]:
```
max(year_list)
```

Out[5]:
```
2013
```

In [6]:
```
sum(year_list)
```

Out[6]:
```
40070
```

In [7]:
```
len(year_list)
```

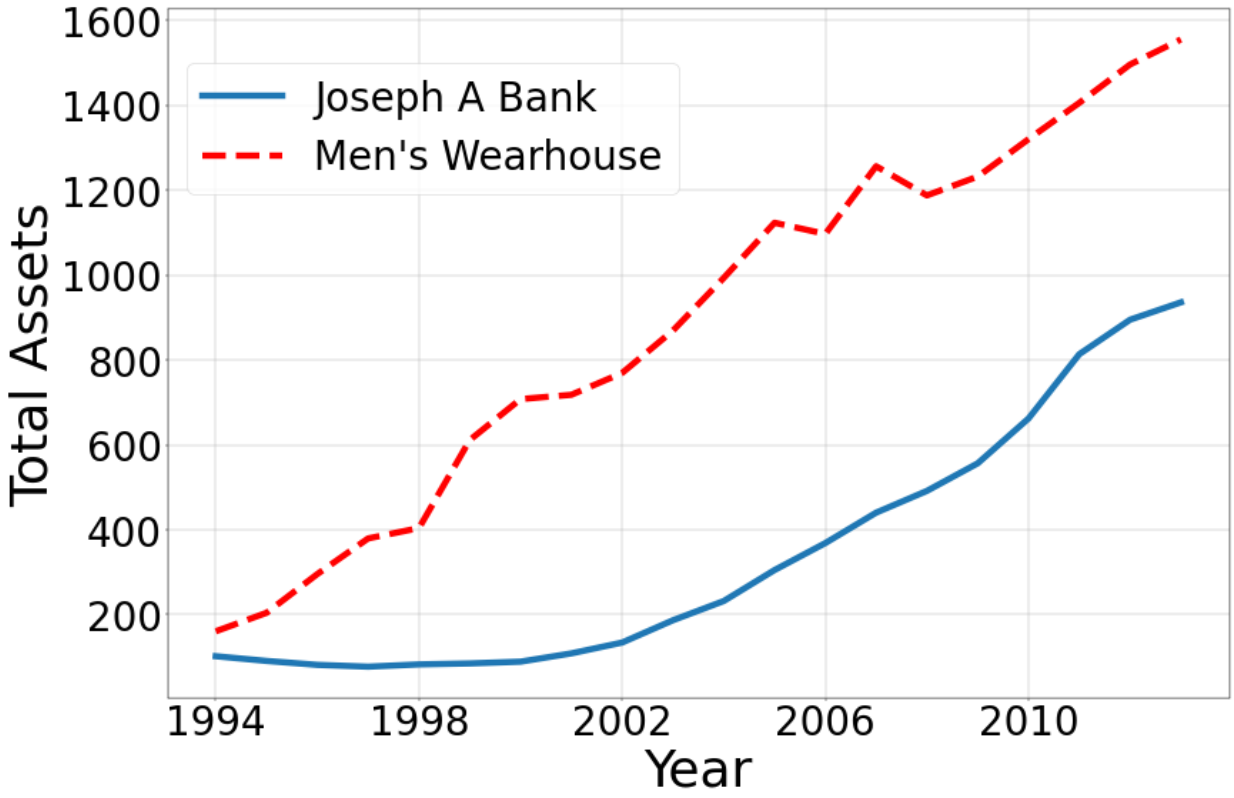
Out[7]:
```
20
```

Total asset values of Jos. A Bank and Men's Wearhouse (two US retailers) from 1994 to 2013.

In [8]:
```
josb_assets_list = [101.783,90.671,81.41,77.144,82.515,84.751,88.954,108.457,
                    304.832,368.392,440.098,491.366,556.364,662.037,813.612,894.
mw_assets_list = [160.494,204.105,295.478,379.415,403.732,611.195,707.734,717
                  1123.274,1096.952,1256.467,1187.73,1232.106,1320.318,1405.952,
```

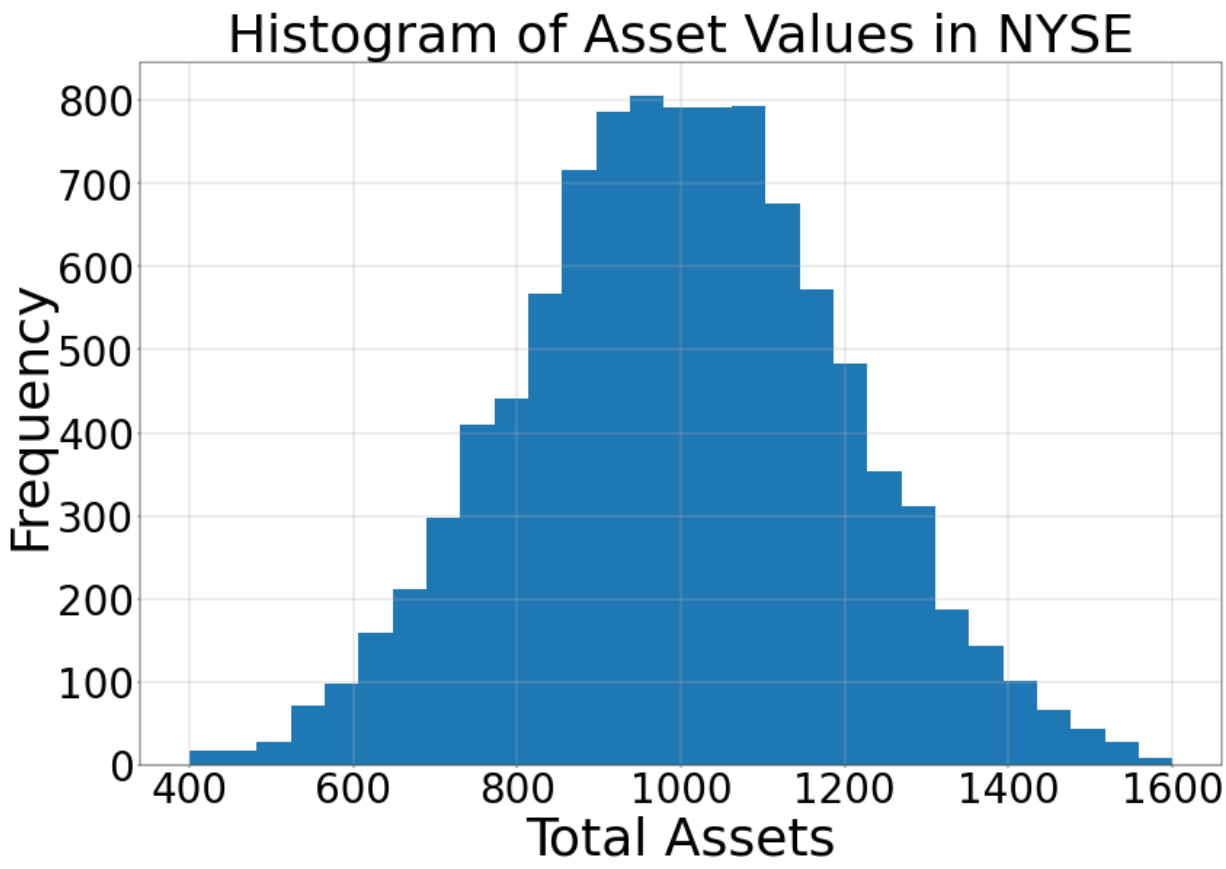
In [9]:
```
plt.figure(figsize=(24,16), dpi= 40)
plt.plot(year_list,josb_assets_list,label="Joseph A Bank",linewidth=8)
plt.plot(year_list,mw_assets_list,'r--',label="Men's Wearhouse",linewidth=8)
xtick_location = np.arange(1994,2013,4)
plt.xticks(ticks=xtick_location, rotation=0, fontsize=50, horizontalalignment
plt.legend(bbox_to_anchor=(0.50, 0.95),fontsize=50)
plt.yticks(fontsize=50)
plt.grid(axis='both')
plt.xlabel('Year', fontsize = 65)
plt.ylabel('Total Assets', fontsize = 65)
```

Out[9]:
```
Text(0, 0.5, 'Total Assets')
```



In [10]:
```
random_values = np.random.normal(1000,200,10000)
bins = np.linspace(400,1600,30)
plt.figure(figsize=(24,16), dpi= 40)
plt.hist(random_values,bins)
plt.xlabel('Total Assets', fontsize = 65)
plt.ylabel('Frequency', fontsize = 65)
plt.xticks(fontsize=50)
plt.yticks(fontsize=50)
plt.grid(axis='both')
plt.title('Histogram of Asset Values in NYSE', fontsize = 65)
```

Out[10]:
```
Text(0.5, 1.0, 'Histogram of Asset Values in NYSE')
```



In [11]:
```
list_vec = [1,2,3,4]
list_vec + [1]
```

Out[11]:
```
[1, 2, 3, 4, 1]
```

In [12]:
```
list_vec = [1,2,3,4]
list_vec.append(1)
list_vec
```

Out[12]:
```
[1, 2, 3, 4, 1]
```

In [13]:
```
import numpy as np
array_vec = np.array([1,2,3,4])
array_vec + np.array([1])
```

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

In [14]:
```
matrix_example = np.array([[1,2,3],[0,-1,2],[1,0,4]])
array_example = np.array([1,-2,1])
```

In [15]:
```
np.dot(matrix_example,array_example)
```

Out[15]:
```
array([0, 4, 5])
```

In [16]:
```
np.linalg.inv(matrix_example)
```

Out[16]:
```
array([[-1.33333333, -2.66666667,  2.33333333],
       [ 0.66666667,  0.33333333, -0.66666667],
       [ 0.33333333,  0.66666667, -0.33333333]])
```

In [17]:
```
np.transpose(matrix_example)
```

Out[17]:
```
array([[ 1,  0,  1],
       [ 2, -1,  0],
       [ 3,  2,  4]])
```

In [18]:
```
np.linalg.eigvals(matrix_example)
```

Out[18]:
```
array([ 4.93394491+0.j        , -0.46697245+0.62447535j,
       -0.46697245-0.62447535j])
```

In [19]:
```
np.linalg.matrix_rank(matrix_example)
```

Out[19]:
```
3
```

In [20]:
```
matrix_example
```

Out[20]:
```
array([[ 1,  2,  3],
       [ 0, -1,  2],
       [ 1,  0,  4]])
```

In [21]:
```
matrix_example[0]
```

Out[21]:
```
array([1, 2, 3])
```

In [22]:
```
matrix_example[1]
```

Out[22]:
```
array([ 0, -1,  2])
```

In [23]:
```
matrix_example[2]
```

Out[23]:
```
array([1, 0, 4])
```

In [24]:
```
matrix_example[-1]
```

Out[24]:
```
array([1, 0, 4])
```

In [25]:
```
matrix_example[2][0]
```

Out[25]:
```
1
```

In [26]:
```
np.random.normal(10,2,[10,3])
```

Out[26]:
```
array([[12.91441419,  8.30828157, 10.973616  ],
       [ 7.84261491,  6.81950351,  5.77954707],
       [12.69116544, 13.1880173 , 12.04609933],
       [ 5.11171636, 11.94145726,  8.7628363 ],
       [11.47035536,  7.58890851, 12.35378731],
       [11.45951297, 11.32183905, 10.26560704],
       [11.23606016, 10.02442418,  9.45153838],
       [14.20407383,  9.50845634, 10.60725446],
       [ 7.96111676, 12.44720155, 11.73183729],
       [ 9.38210558, 11.1864316 , 11.74386397]])
```

In [27]:
```
import scipy as sp
```

In [28]:
```
sp.stats.norm.pdf(14,10,2)
```

Out[28]:
```
0.02699548325659403
```

In [29]:
```
sp.stats.norm.cdf(14,10,2)
```

Out[29]:
```
0.9772498680518208
```

In [30]:
```
sp.stats.norm.isf(1-0.9772498680518208,10,2)
```

Out[30]:
```
14.0
```

In [31]:
```
sp.stats.norm.cdf(2,0,1)
```

Out[31]:
```
0.9772498680518208
```

In [32]:
```
sp.stats.norm.isf(1-0.9772498680518208,0,1)*2+10
```

Out[32]:
```
14.0
```

In [33]:
```
import pandas as pd
```

In [34]:
```
pd.Series(np.random.normal(10,2,8))
```

Out[34]:
```
0    12.146091
1     7.770245
2    11.059969
3    10.838351
4    14.348875
5     7.250432
6    12.346772
7     9.617587
```

```
dtype: float64
```

In [36]:
```
DataTable = pd.DataFrame(np.random.normal(10,2,[8,3]),columns=list(["column 1
DataTable
```

Out[36]:

	column 1	column 2	column 3
0	6.189621	6.076293	8.424513
1	11.799024	9.703526	10.042558
2	8.196007	5.054987	12.961477
3	7.269611	4.702802	10.525184
4	11.410864	7.255142	8.316879
5	9.915314	9.773392	9.926654
6	10.052845	8.422706	8.763066
7	6.435722	4.479016	5.969367

In [37]:
```
DataTable[DataTable["column 1"]>13]
```

Out[37]:

	column 1	column 2	column 3

In [40]:
```
import pandas as pd
input_table = pd.read_csv('sales_data.csv')
input_table.head()
```

Out[40]:

	Order ID	Product ID	Amount (kg)	Order Confirm Date	Required Delivery Date
0	1005781.0	8908023.0	20000.0	NaN	2020-07-30
1	1005998.0	8909373.0	6297.0	NaN	2020-07-15
2	1005969.0	8908023.0	20000.0	2020-05-29	NaN
3	1005969.0	8908023.0	20000.0	2020-05-29	NaN
4	NaN	NaN	NaN	NaN	NaN

In [41]:
```
pd.pivot_table(input_table,values="Amount (kg)",index="Product ID")
```

Out[41]:

	Amount (kg)
Product ID	
8601971.0	740.000000
8601986.0	956.000000
8902398.0	3937.372553
8902615.0	2960.000000
8902635.0	17292.335506

Product ID	Amount (kg)
8905735.0	1754.080000
8906381.0	3579.833333
8907133.0	5435.416667
8908023.0	20063.473486
8908024.0	16400.000000
8908111.0	11000.000000
8908112.0	16647.885521
8908114.0	18929.411765
8908672.0	17875.806452
8908748.0	19866.220736
8908749.0	20299.019704
8908880.0	4766.666667
8909286.0	5975.000000
8909373.0	5698.556667

In [47]:
```
input_table_imputed_partially = input_table.dropna(subset=["Order Confirm Dat
input_table_imputed_fully = input_table.dropna()
input_table_imputed_partially.head()
```

Out[47]:

	Order ID	Product ID	Amount (kg)	Order Confirm Date	Required Delivery Date
2	1005969.0	8908023.0	20000.0	2020-05-29	NaN
3	1005969.0	8908023.0	20000.0	2020-05-29	NaN
5	1006069.0	8902635.0	10000.0	2020-06-16	2020-08-05
6	1005998.0	8909373.0	6500.0	2020-06-30	2020-09-10
7	1006067.0	8908023.0	21000.0	2020-06-22	2020-07-07

In [51]:
```
import numpy as np
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder

shopping_basket = pd.read_csv("small_dataset.csv")
shopping_basket.head()
```

Out[51]:

	Transaction ID	ItemsPurchased
0	1	biscuit,bread,milk
1	2	biscuit,bread,cereal,milk
2	3	bread,tea

	Transaction ID	ItemsPurchased
3	4	bread,jam,milk
4	5	biscuit,tea

In [57]:
```
transaction_list = list(shopping_basket["ItemsPurchased"].apply(lambda t:t.sp
encoder = TransactionEncoder().fit(transaction_list)
encoded_transaction_list = encoder.transform(transaction_list)
ideal_format_table = pd.DataFrame(encoded_transaction_list, columns = encoder
ideal_format_table.head()
```

Out[57]:

	biscuit	bread	cereal	jam	milk	tea
0	True	True	False	False	True	False
1	True	True	True	False	True	False
2	False	True	False	False	False	True
3	False	True	False	True	True	False
4	True	False	False	False	False	True

In [58]:
```
shopping_basket_v2 = pd.read_csv("small_dataset_v2.csv")
shopping_basket_v2.head()
```

Out[58]:

	Transaction ID	ItemsPurchased
0	1	biscuit
1	1	bread
2	1	milk
3	2	biscuit
4	2	bread

In [72]:
```
format_b_to_a = shopping_basket.groupby(["Transaction ID"])["ItemsPurchased"]
apply(lambda t: ','.join(t)).reset_index()

transaction_list = list(format_b_to_a["ItemsPurchased"].apply(lambda t:t.spli
encoder = TransactionEncoder().fit(transaction_list)
encoded_transaction_list = encoder.transform(transaction_list)
ideal_format_table = pd.DataFrame(encoded_transaction_list, columns = encoder
ideal_format_table.head()
```

Out[72]:

	biscuit	bread	cereal	jam	milk	tea
0	True	True	False	False	True	False
1	True	True	True	False	True	False
2	False	True	False	False	False	True
3	False	True	False	True	True	False

	biscuit	bread	cereal	jam	milk	tea
4	True	False	False	False	False	True

In [37]:
```
x = 12
if x<11:
    print("x is less than 11")
```

In [38]:
```
if x>=11:
    print("x is greater than 11")
```

```
x is greater than 11
```

In [39]:
```
IterVariable = ["Iter1","Iter2","Iter3","Iter4","Iter5","Iter6"]

for i in IterVariable:
    print(i)
```

```
Iter1
Iter2
Iter3
Iter4
Iter5
Iter6
```

In [40]:
```
IterVariable = np.array([1,2,3,4,5])
res_sum = 0
for i in IterVariable:
    res_sum = res_sum + i
print(res_sum)
```

```
15
```

In [41]:
```
i = 1
while i<5:
    print("i is less than 5")
    i = i+1
print("Now i is 5")
```

```
i is less than 5
i is less than 5
i is less than 5
i is less than 5
Now i is 5
```

In [42]:
```
for i in range(2,7):
    ifprim = True
    for j in range(2,i):
        if i%j==0:
            print(i, ":not a prime number")
            ifprim = False
            break
        if ifprim ==True:
            print(i, ": a prime number")
```

```
3 : a prime number
4 :not a prime number
5 : a prime number
5 : a prime number
5 : a prime number
6 :not a prime number
```

In [43]:
```
for i in range(2,7):
    if i%2==0:
        print(i,": an even number")
        continue
    print(i,": an odd number")
```

```
2 : an even number
3 : an odd number
4 : an even number
5 : an odd number
6 : an even number
```

In [44]:
```
def standard_normal(x,mean,std_dev):
    return (x-mean)/std_dev
```

In [45]:
```
sp.stats.norm.cdf(14,10,2)
```

Out[45]:
```
0.9772498680518208
```

In [46]:
```
sp.stats.norm.cdf(standard_normal(14,10,2))
```

Out[46]:
```
0.9772498680518208
```

This is the class example from GS textbook

In [60]:
```
matrix_A = np.array([[1,2,3],[2,5,7],[2,7,8]])
```

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

In the class we have found the following values for L and U

In [61]:
```
L_comp = np.array([[1,0,0],[2,1,0],[2,3,1]])
U_comp = np.array([[1,2,3],[0,1,1],[0,0,-1]])
np.dot(L_comp,U_comp)
```

Out[61]:
```
array([[1, 2, 3],
       [2, 5, 7],
       [2, 7, 8]])
```

In scipy we have a standard function that returns

In [53]:
```
p, l, u = sp.linalg.lu(matrix_A,permute_l=False)
np.dot(np.dot(p,l),u)
```

Out[53]:
```
array([[1., 2., 3.],
       [2., 5., 7.],
       [2., 7., 8.]])
```

In [59]:
```
p
```

Out[59]:
```
array([[0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.]])
```

In [56]:
```
np.linalg.inv(p)
```

Out[56]:
```
array([[0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.]])
```

In [58]:
```
np.transpose(p)
```

Out[58]:
```
array([[0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.]])
```

In [63]:
```
l
```

Out[63]:
```
array([[ 1.  ,  0.  ,  0.  ],
       [ 1.  ,  1.  ,  0.  ],
       [ 0.5 , -0.25,  1.  ]])
```

In [64]:
```
u
```

Out[64]:
```
array([[ 2.  ,  5.  ,  7.  ],
       [ 0.  ,  2.  ,  1.  ],
       [ 0.  ,  0.  , -0.25]])
```

In [70]:
```
np.dot(np.dot(p,l),u)
```

Out[70]:
```
array([[1., 2., 3.],
       [2., 5., 7.],
       [2., 7., 8.]])
```

In [ ]: