In [1]:
```
import numpy as np
```

In [2]:
```
import matplotlib.pyplot as plt
```

In [8]:
```


x=np.array([[1,2],[3,4]], dtype=np.int32)
y=np.array([[5,6],[7,8]], dtype=np.int32)
v=np.array([9,10])
w=np.array([11,12])

#Inner product
print("Inner Product:")
print(v.dot(w))
print(np.dot(v,w))
print(np.inner(v,w))

#outer products of vectors
print("Outer product:")
print(np.outer(v,w))

#Norm of a vector
print("Norm:")
print(np.linalg.norm(v))#Euclidean
print(np.linalg.norm(v,1))#Manhattan
print(np.linalg.norm(v,np.inf))#Infinite
print(np.linalg.norm(x))
print(np.linalg.norm(x,axis=1))
```

```
Inner Product:
219
219
219
Outer product:
[[ 99 108]
 [110 120]]
Norm:
13.45362404707371
19.0
10.0
5.477225575051661
[2.23606798 5.        ]
```

Vector matrix and matrix-matrix multiplication

In [16]:
```
#matrix/vector product
print(x.dot(v))
print(np.dot(x,v))
print(np.inner(x,v))

#Matrix/matrix product(multiplication)
print(x.dot(y))
print(np.dot(x,y))
print(np.matmul(x,y))
print(x@y)
```

```
[29 67]
[29 67]
[29 67]
[[19 22]
 [43 50]]
[[19 22]
 [43 50]]
[[19 22]
 [43 50]]
[[19 22]
 [43 50]]
```

Determiniant ,trace and rank of a matrix

In [27]:
```
m=np.array([[2,4,6],[1,5,9],[3,7,8]])

#determiniant of a matrix
print('Determiniant: %.2f'%np.linalg.det(m))

#trace of a matrix
print('Trace: %.2f'%np.trace(m))

#rank of a matrix
print('Rank: ',np.linalg.matrix_rank(m))
```

```
Determiniant: -18.00
Trace: 15.00
Rank:  3
```

Transpose and Inverse of a matrix

In [30]:
```
#transpose of a matrix
mt1=m.T
print(mt1)

mt2=np.transpose(m)
print(mt2)

#Inverse of a matrix
inv_m=np.linalg.inv(m)
print(inv_m)
```

```
[[2 1 3]
 [4 5 7]
 [6 9 8]]
[[2 1 3]
 [4 5 7]
 [6 9 8]]
[[ 1.27777778 -0.55555556 -0.33333333]
 [-1.05555556  0.11111111  0.66666667]
 [ 0.44444444  0.11111111 -0.33333333]]
```

Solving a Linear system

In [33]:
```
#solving a kinear system
# Ax=Z=>x=inv(A).Z
A=np.array([[1,2],[3,4]])
Z=np.array([[5],[6]])

#using inverse of a matrix
print('Using inverse :\n',np.linalg.inv(A).dot(Z)) #slow

#using solve function
print('Using solve:\n',np.linalg.solve(A,Z))
```

```
Using inverse :
 [[-4. ]
 [ 4.5]]
Using solve:
 [[-4. ]
 [ 4.5]]
```

Least square solution to a linear matrix equation

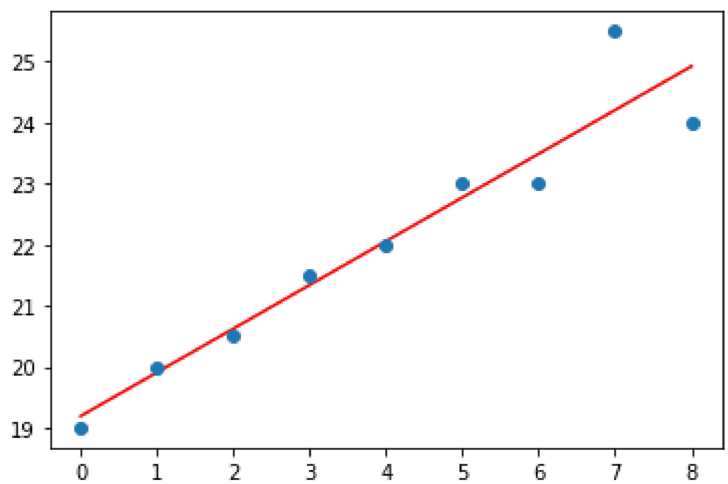
In [45]:
```
x=np.arange(0,9)
#print(x)
A=np.array([x,np.ones(9)])
print(A)

#linearly generated sequence
y=[19,20,20.5,21.5,22,23,23,25.5,24]

#obtaining the parameters of regression line
w = np.linalg.lstsq(A.T, y, rcond=None)[0]
print(w)

#plotting the line
line=w[0]*x+w[1] #regression line
plt.plot(x,line,'r-')
plt.plot(x,y,'o')
plt.show()
```

```
[[0. 1. 2. 3. 4. 5. 6. 7. 8.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1.]]
[ 0.71666667 19.18888889]
```



Eigenvalues and Eigenvectors

In [48]:
```
A=np.array([[0,1],[-2,-3]])
#function to calculate eigenvalues and vector
val,vect=np.linalg.eig(A)
print(val)
print(vect)
```

```
[-1. -2.]
[[ 0.70710678 -0.4472136 ]
 [-0.70710678  0.89442719]]
```

Importing the Dataset and Splitting into Train and Test Dataset

In [50]:
```
import pandas as pd
#Importing the data set
dataset=pd.read_csv('SimpleLinearRegression.csv')
dataset.describe()
```

Out[50]:

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

In [54]:
```
X=dataset[['YearsExperience']]
y=dataset[['Salary']]
#splitting the dataset into training and test dataset
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=1/3,random_state=0)
```

Fitting the dataset into Simple Linear Regression Model

In [56]:
```
#Fitting Simple Linear Regression to the Tarining set
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train,y_train)
```

Out[56]:
```
LinearRegression()
```

In [58]:
```
#predicting the values of the test set
y_pred=regressor.predict(X_test)
```

visualizing the correlation

In [59]:
```
plt.scatter(X_train,y_train,color='red')
plt.plot(X_train, regressor.predict(X_train),color='blue')
plt.title('Salary vs Experiance(Training set)')
plt.xlabel('Years of Experiancs')
plt.ylabel('Salary')
plt.show()
```

In [60]:
```
plt.scatter(X_test,y_test,color='green')
plt.plot(X_train, regressor.predict(X_train),color='blue')
plt.title('Salary vs Experiance(Training set)')
plt.xlabel('Years of Experiancs')
plt.ylabel('Salary')
plt.show()
```

Model Evalution for test dataset

In [61]:
```
from sklearn import metrics
# model evaluation for testing set
mae = metrics.mean_absolute_error(y_test, y_pred)
mse = metrics.mean_squared_error(y_test, y_pred)
r2 = metrics.r2_score(y_test, y_pred)
print("The model performance for testing set")
print("--------------------------------------")
print('MAE is %.2f'% mae)
print('MSE is %.2f'% mse)
print('R2 score is %.2f'% r2)
```

```
The model performance for testing set
--------------------------------------
MAE is 3426.43
MSE is 21026037.33
R2 score is 0.97
```