EXERCISE 1: PART A: IPYTHON

Gökce Sucu 246112

A.1: Number of Unique Non-Stop Words

A.1.1 Opening and Reading The Text

First, we need to open and read the text. !!!!Because the text is latin there is no stop english words.!!!!

In [1]:
```
#openning the text file and cleaning from \n , . characters
mytext = open('latin_text.txt')
mytext = mytext.read() #reading the file
print(mytext) #printingthe original file
```

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec in erat maximus, eleif
end ante eget, feugiat libero. Maecenas vel pharetra orci. Morbi vitae finibus augue.
Nunc pharetra ac lacus vel efficitur. Cras sed urna non ex luctus imperdiet. Sed eu v
olutpat nunc. Maecenas et nunc tellus. Nunc eget purus consectetur, ultricies leo iac
ulis, gravida elit. Nunc molestie tortor dolor, nec laoreet lectus sollicitudin sed.
Duis ultricies at diam nec tincidunt. Donec at metus placerat lacus commodo molestie
nec ut risus. Curabitur fermentum justo lectus, non maximus mauris placerat vel. Fusc
e auctor ex massa, nec condimentum enim interdum nec. Maecenas consectetur varius nun
c, auctor pellentesque mi feugiat eu. Cras lacinia, ipsum vitae malesuada eleifend, m
i magna hendrerit enim, id fringilla urna ante in magna.

Suspendisse dolor nisl, interdum sit amet rutrum eget, congue sit amet odio. Vestibul
um vestibulum orci sed volutpat facilisis. Quisque consectetur convallis mauris. In p
retium velit arcu, sit amet facilisis massa semper at. Praesent nec lectus sit amet e
lit viverra cursus. Donec suscipit diam id metus condimentum tristique. Duis mollis e
nim eu ornare commodo. Nullam molestie vel enim in ornare.

Fusce tempor elit at mauris imperdiet tempor. Donec ut felis eu neque convallis lobor
tis. Donec non tortor ut tellus hendrerit finibus at at nisl. Quisque iaculis finibus
sem non finibus. Aenean lobortis ornare pellentesque. Vivamus euismod urna tortor, in
venenatis nisi venenatis eu. Nullam finibus libero a elementum varius.

Ut id tincidunt metus. In sed ante et sem posuere rutrum at eu leo. Integer consequat
lacus sit amet ante placerat tempor. Fusce commodo enim vitae elit lobortis, non sagi
ttis eros sagittis. Nam ultricies, elit at dapibus convallis, tellus ligula sagittis
risus, non condimentum odio dolor eget ex. Etiam feugiat nulla sed tellus ornare pret
ium. Nam porta vestibulum dui iaculis tincidunt. Aenean a dignissim dui. Nulla facili
si. Aliquam faucibus sodales accumsan. Donec non quam in justo aliquam luctus vel int
erdum risus. Duis sodales rhoncus est, sed semper urna ullamcorper sit amet. Nulla sa
gittis ante vel molestie blandit. Aliquam maximus felis ut porttitor rutrum. Duis eu
tempor mauris. Mauris nisi orci, feugiat sed blandit euismod, tempus sed lorem.

Nullam interdum lorem a lectus efficitur pulvinar. Nullam mattis efficitur elit ac ru
trum. Donec in ante vitae est dignissim mollis. Phasellus at velit id lectus vehicula
condimentum. Vestibulum egestas turpis in nisl tristique luctus. Fusce convallis, aug
ue iaculis dignissim tempus, augue ex convallis ante, non euismod lectus diam eu nis
l. Pellentesque tristique diam in augue facilisis euismod eget et ante. Sed in faucib
us ante. In hac habitasse platea dictumst. Proin elit ligula, suscipit id risus eu, v
olutpat varius libero. Mauris eget metus ut nibh vulputate euismod. Sed justo felis,
accumsan sit amet dui vitae, dictum suscipit ligula. Sed commodo dapibus tincidunt.
```

A.1.2. Cleaning the All Punctuations From text

Second, we should delete the all punctuations.

In [2]:
```
mytext = mytext.replace('\n',' ')  # deleting \n
mytext = mytext.replace(',',' ')   #deleting ,
mytext = mytext.replace('.',' ')   #deleting .
print(mytext)
```

```
Lorem ipsum dolor sit amet  consectetur adipiscing elit  Donec in erat maximus  eleif
end ante eget  feugiat libero  Maecenas vel pharetra orci  Morbi vitae finibus augue
Nunc pharetra ac lacus vel efficitur  Cras sed urna non ex luctus imperdiet  Sed eu v
olutpat nunc  Maecenas et nunc tellus  Nunc eget purus consectetur  ultricies leo iac
ulis  gravida elit  Nunc molestie tortor dolor  nec laoreet lectus sollicitudin sed
Duis ultricies at diam nec tincidunt  Donec at metus placerat lacus commodo molestie
nec ut risus  Curabitur fermentum justo lectus  non maximus mauris placerat vel  Fusc
e auctor ex massa  nec condimentum enim interdum nec  Maecenas consectetur varius nun
c  auctor pellentesque mi feugiat eu  Cras lacinia  ipsum vitae malesuada eleifend  m
i magna hendrerit enim  id fringilla urna ante in magna  Suspendisse dolor nisl  int
erdum sit amet rutrum eget  congue sit amet odio  Vestibulum vestibulum orci sed volu
tpat facilisis  Quisque consectetur convallis mauris  In pretium velit arcu  sit amet
facilisis massa semper at  Praesent nec lectus sit amet elit viverra cursus  Donec su
scipit diam id metus condimentum tristique  Duis mollis enim eu ornare commodo  Nulla
m molestie vel enim in ornare  Fusce tempor elit at mauris imperdiet tempor  Donec u
t felis eu neque convallis lobortis  Donec non tortor ut tellus hendrerit finibus at
at nisl  Quisque iaculis finibus sem non finibus  Aenean lobortis ornare pellentesque
Vivamus euismod urna tortor  in venenatis nisi venenatis eu  Nullam finibus libero a
elementum varius  Ut id tincidunt metus  In sed ante et sem posuere rutrum at eu leo
Integer consequat lacus sit amet ante placerat tempor  Fusce commodo enim vitae elit
lobortis  non sagittis eros sagittis  Nam ultricies  elit at dapibus convallis  tellu
s ligula sagittis risus  non condimentum odio dolor eget ex  Etiam feugiat nulla sed
tellus ornare pretium  Nam porta vestibulum dui iaculis tincidunt  Aenean a dignissim
dui  Nulla facilisi  Aliquam faucibus sodales accumsan  Donec non quam in justo aliqu
am luctus vel interdum risus  Duis sodales rhoncus est  sed semper urna ullamcorper s
it amet  Nulla sagittis ante vel molestie blandit  Aliquam maximus felis ut porttitor
rutrum  Duis eu tempor mauris  Mauris nisi orci  feugiat sed blandit euismod  tempus
sed lorem  Nullam interdum lorem a lectus efficitur pulvinar  Nullam mattis efficitu
r elit ac rutrum  Donec in ante vitae est dignissim mollis  Phasellus at velit id lec
tus vehicula condimentum  Vestibulum egestas turpis in nisl tristique luctus  Fusce c
onvallis  augue iaculis dignissim tempus  augue ex convallis ante  non euismod lectus
diam eu nisl  Pellentesque tristique diam in augue facilisis euismod eget et ante  Se
d in faucibus ante  In hac habitasse platea dictumst  Proin elit ligula  suscipit id
risus eu  volutpat varius libero  Mauris eget metus ut nibh vulputate euismod  Sed ju
sto felis  accumsan sit amet dui vitae  dictum suscipit ligula  Sed commodo dapibus t
incidunt
```

A.1.3. Counting the Unique Words of the Text

Now, we should find how many unique words exist in the text. First, we would define a fuction which counts the unique words of any text. While I am defining a function, first I would make lowercas all lettrs of a word in the text, because 'lorem' and 'Lorem' would be counted as different two words. Then I am gonna use set() function to eliminate repititon of words.

In [3]:
```
#defining a function which counts words
def unique_words(the_text):
    #splitting the test into words
    allwords_of_text = the_text.split()

    #translating the all letters in the text into lowercase
    lower_text = []
    for x in allwords_of_text:
        lower_text.append(x.lower())

    #eleminating the repeated words
    b =set(lower_text)

    #number of unique words
    return len(b)
```

Secondly, we would apply this function to our text to find unique words.

In [4]:
```
#application of unique words function to random_text.txt
unique_words(mytext)
```

Out[4]: 151

So, number of unique words of the text is 151.

A.2: TOP 5 WORDS

A.2.1. Splitting the Text into Words with Lowercase

Now, we would like to find most frequent top 5 words.

In [5]:
```
#splitting the text into the words and making lowercase all of the text characters
lower_text = []
for x in mytext.split():
    lower_text.append(x.lower())

#all unique words in the text
all_words = set(lower_text)
```

Now, we would find the frequency

In [6]:
```
frequency_of_words = []
for x in all_words:
    frequency_of_words.append((lower_text.count(x),x))
```

In [7]:
```
frequency_of_words.sort()
frequency_of_words.reverse()
```

In [8]:
```
print('Most Frequet Words:\n',frequency_of_words[:5])
```

```
Most Frequet Words:
 [(12, 'sed'), (12, 'in'), (9, 'eu'), (9, 'at'), (9, 'ante')]
```

EXERCISE 1: PART B: NUMPY

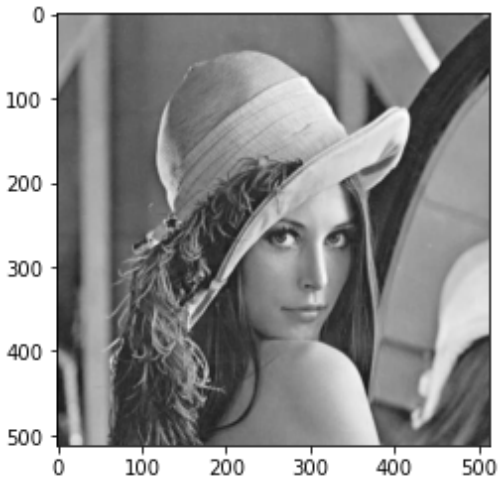
B.1. INSTALLATION AND DISPLAY

In [9]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
```

In [10]:
```
img = mpimg.imread('pic.jpg')
print(img)
plt.imshow(img, cmap='gray', vmin=0,vmax=225)
```

```
[[144 143 143 ... 153 135 110]
 [145 144 144 ... 146 128 104]
 [145 145 144 ... 149 132 109]
 ...
 [ 38  40  43 ...  85  84  84]
 [ 39  41  44 ...  87  87  88]
 [ 40  42  45 ...  86  87  89]]
```

Out[10]: <matplotlib.image.AxesImage at 0x1c46b4866d0>



B.2 AVERAGING FILTER

In [11]:
```
the_filter =(1/9)* np.ones((3,3))
print(the_filter)
```

```
[[0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111]]
```

B.3. PADDING THE IMAGE

To apply padding to a matrix, we should add 0 vector to around matrix. For this reason, we will use np.vstack and np.hstack codes. For this reason we need 4 step. 2 times vertical for under and top, 2 times horizontal for left and right.

In [12]:
```
def padding(the_array):
    v = np.zeros((1, the_array.shape[0])) #creating vertical 0 vector
    h = np.zeros((the_array.shape[1]+2,1)) #creating horizontal 0 vector woth 2 more d
imension because after 2 vertical padding, matrix dimension will increase 2 more
    one = np.vstack((the_array,v)) #adding the vertical 0 vector to under the matrice
    two = np.vstack((v,one))    #adding the vertical 0 matrice t on the matrice
    three = np.hstack((two, h)) #adding the horizontal vector to right side of the mat
rice
    four = np.hstack((h, three)) #adding the horizontal vector to the left side of the
matrice
    return four
```

In [13]:
```
padding(img)
```

Out[13]:
```
array([[  0.,   0.,   0., ...,   0.,   0.,   0.],
       [  0., 144., 143., ..., 135., 110.,   0.],
       [  0., 145., 144., ..., 128., 104.,   0.],
       ...,
       [  0.,  39.,  41., ...,  87.,  88.,   0.],
       [  0.,  40.,  42., ...,  87.,  89.,   0.],
       [  0.,   0.,   0., ...,   0.,   0.,   0.]])
```

B.4. CONVOLUTION

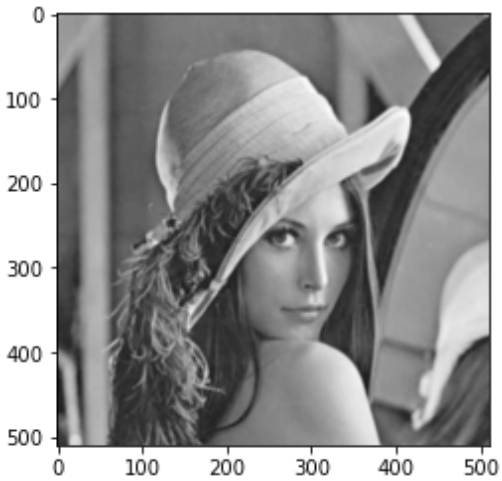
In [14]:
```
#defining the kernel
k = 1/9 * np.ones((3,3))


def convolution_function(c_matrice):
    element_wise =[] #created this set for result of elementwise wultiplication of ker
nell and 3x3 submatrices
    sum_of_ewise =[] #creating this set for summing of every matrix elements in elemen
t_wise set
    for i in range(0,c_matrice.shape[0]-2):
        for j in range(0,c_matrice.shape[0]-2):
            c=k*c_matrice[i:i+3,j:j+3]
            element_wise.append(c)

    for y in element_wise:
        sum_of_ewise.append(y.sum())

    return np.array(sum_of_ewise).reshape(c_matrice.shape[0]-2,c_matrice.shape[0]-2)
```

In [15]:
```
#lets apply the function zo example in the exercise sheet to be sure i
matris = np.array([[224,215,47],[214,90,89],[225,250,247]])
print(convolution_function(matris))
```

```
[[177.88888889]]
```

In [16]:
```
img_convolved_one =convolution_function(img) #first convolution
plt.imshow(img_convolved_one , cmap='gray', vmin=0,vmax=225)
```

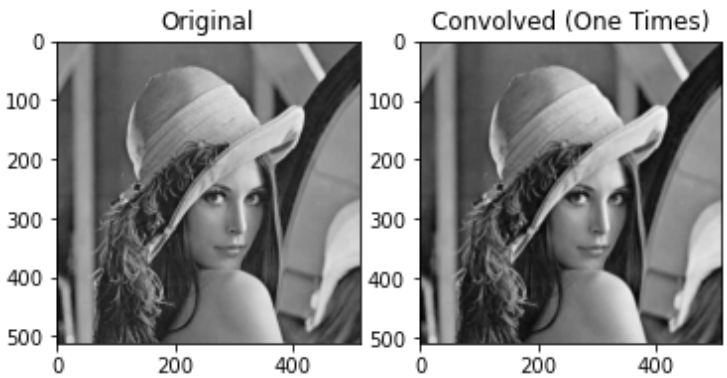
Out[16]:
```
<matplotlib.image.AxesImage at 0x1c46b5219d0>
```



B. 5. PLOTTING THE ORIGINAL AND BLUR

In [17]:
```
plt.subplot(121), plt.imshow(img, cmap='gray'),plt.title('Original')
plt.subplot(122), plt.imshow(img_convolved_one, cmap='gray'),plt.title('Convolved (One Times)')
```

Out[17]:
```
(<matplotlib.axes._subplots.AxesSubplot at 0x1c46edb96a0>,
 <matplotlib.image.AxesImage at 0x1c46b785130>,
 Text(0.5, 1.0, 'Convolved (One Times)'))
```

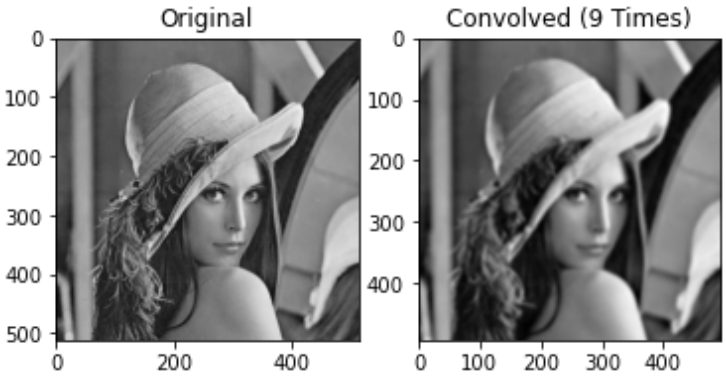


B. 6. APPLYING THE FILTER MULTIPLE TIMES

In [18]:
```
#convolution repeated 9 times
img_convolved_two = convolution_function(img_convolved_one) #second convolution
img_convolved_three = convolution_function(img_convolved_two) #third convolutuon
img_convolved_four= convolution_function(img_convolved_three) #fourth convolution
img_convolved_five = convolution_function(img_convolved_four) #5th convolution
img_convolved_six = convolution_function(img_convolved_five)  #6th the convolution
img_convolved_seven = convolution_function(img_convolved_six) #7th convolution
img_convolved_eight = convolution_function(img_convolved_seven) #8th convolution
img_convolved_nine = convolution_function(img_convolved_eight)  #9th  convolution
```

In [19]:
```
#comparison between original image and 9 times filtered image
plt.subplot(121), plt.imshow(img, cmap='gray'),plt.title('Original')
plt.subplot(122), plt.imshow(img_convolved_nine, cmap='gray'),plt.title('Convolved (9
 Times)')
```

Out[19]:
```
(<matplotlib.axes._subplots.AxesSubplot at 0x1c46b7babb0>,
 <matplotlib.image.AxesImage at 0x1c46ee0ed30>,
 Text(0.5, 1.0, 'Convolved (9 Times)'))
```



EXERCISE 2: LINEAR REGRESSION

In [20]:
```
#needed packages
import random
import numpy as np
import matplotlib.pyplot as plt
import random
```

1. Generating matrix X

In [21]:
```
#mean=2 and sigma=0.01
x = np.random.normal(2, 0.01, 200).reshape(100,2)
#checking the created matrix's dimension
print('Dimension of x :\n',x.shape)
```

```
Dimension of x :
 (100, 2)
```

2. Generating matrix Y

In [22]:
```
#creating a random matrix with uniform distribution
y =np.random.uniform( 0,10,100).reshape(100,1)

#checking the dimension of created matrrix
print('Dimension of y:\n',y.shape)
```

```
Dimension of y:
 (100, 1)
```

3. Learn Simple Linear Regression

First we have to solve this linear equation problem.

$min \sum_{i=1}^{n} (\beta_0 x_i^{(1)} + \beta_1 x_i^{(2)} - y_i)^2$

If we use

$$\frac{\partial}{\beta_0} = 0$$

and

$$\frac{\partial}{\beta_1} = 0$$

then we obtain

$$\beta_1 = \frac{\sum_{i=1}^{N} y_i x_i^{(2)} - \beta_0 \sum_{i=1}^{N} x_i^{(1)} x_i^{(2)}}{\sum_{i=1}^{N} x_i^{(2)^2}}$$

and

$$\beta_0 = \frac{\sum_{i=1}^{N} x_i^{(1)} y_i \sum_{i=1}^{N} x_i^{(2)^2} - \sum_{i=1}^{N} y_i x_i^{(2)} \sum_{i=1}^{N} x_i^{(1)} x_i^{(2)}}{\sum_{i=1}^{N} x_i^{(1)^2} \sum_{i=1}^{N} x_i^{(2)^2} - (\sum_{i=1}^{N} x_i^{(1)} x_i^{(2)})^2}$$

In [23]:
```
sum_xone_y=0
for i in range(0,x.shape[0]):
     sum_xone_y= sum_xone_y+x[i][0]+y[i][0]


sum_xtwo_y=0
for i in range(0,x.shape[0]):
    sum_xtwo_y=sum_xtwo_y + x[i][1]+y[i][0]


sum_xone_xtwo=0
for i in range(0,x.shape[0]):
    sum_xone_xtwo=sum_xone_xtwo + x[i][0]+x[i][1]

sum_xone_xone=0
for i in range(0,x.shape[0]):
    sum_xone_xone=sum_xone_xone+ x[i][0]+y[i][0]


sum_xtwo_xtwo=0
for i in range(0,x.shape[0]):
   sum_xtwo_xtwo=sum_xtwo_xtwo + x[i][1]+x[i][1]
```

In [24]:
```
def beta_zero(x,y):
    return (sum_xone_y*sum_xtwo_xtwo-sum_xtwo_y*sum_xone_xtwo)/(sum_xone_xone*sum_xtwo
_xtwo-sum_xone_xtwo**2)

def beta_one(x,y):
    return (sum_xtwo_y - beta_zero(x,y)*sum_xone_xtwo)/sum_xtwo_xtwo
```

In [25]:
```
print(beta_zero(x,y))
print(beta_one(x,y))
```

```
0.00013546686660821148
1.8550424007007245
```

4. Inverting Matrix A

Here we will calculate the $A = X^T X$.

In [26]:
```
#defining a function which gives a matrix multiplication of an matrix and its transpos
e:
def matrix_A(x):
    a_one_one = 0
    a_one_two = 0
    a_two_one = 0
    a_two_two = 0

    A = np.zeros((x.shape[1],x.shape[1]))
    for i in range(0,x.shape[0]):
        a_one_one = a_one_one+ x[i][0]**2
        A[0,0] = a_one_one

    for j in range(0,x.shape[0]):
        a_one_two = a_one_two + +x[j][0]*x[j][1]
        A[0,1]=a_one_two


    for k in range(0,x.shape[0]):
        a_two_one = a_two_one + x[k][1]*x[k][0]
        A[1,0] = a_two_one

    for l in range(0,x.shape[0]):
        a_two_two = a_two_two+x[l][1]**2
        A[1,1]=a_two_two

    return A
```

In [27]:
```
# A = $x^Tx$
A=matrix_A(x)
print(A)
```

```
[[400.22289267 400.32146678]
 [400.32146678 400.43892379]]
```

5. Predict Simple Linear Reg

Now we will use $\beta = (X^T X)^{-1} X^T Y$ formula. First we will calculate $A^{-1} = (X^T X)^{-1}$.

In [28]:
```
#det function for 2x2 dimension matrx:
def det(A):
    return A[0,0]*A[1,1]-A[0,1]*A[1,0]

#defining a function which gives the inverse a 2x2 matrix:
def inverse(A):
    R = np.zeros((2,2))
    R[1,1]= A[0,0]
    R[0,1] =- A[0,1]
    R[1,0]=-A[1,0]
    R[0,0]=A[1,1]

    return R*(1/det(A))
```

In [29]:
```
inverse(A)
```

Out[29]:
```
array([[ 53.05476741, -53.03920535],
       [-53.03920535,  53.02614511]])
```

First, we will calculate the $B = (X^T X)^{-1} X^T$

In [30]:
```
B=np.zeros((2,100))
for i in range(0,x.shape[0]):
    B[0,i]=inverse(A)[0,0]*x[i][0]+inverse(A)[0,1]*x[i][1]
    B[1,i]=inverse(A)[1,0]*x[i][0]+inverse(A)[1,1]*x[i][1]
```

Second, we will calculate $BY = (X^T X)^{-1} X^T Y$

In [31]:
```
b_one = 0
b_two = 0
for i in range(0,B.shape[1]):
    b_one = b_one+ B[0][i]*y[i]
    b_two = b_two+ B[1][i]*y[i]
```

Then, we would obtain Beta =(β_1, β_2).

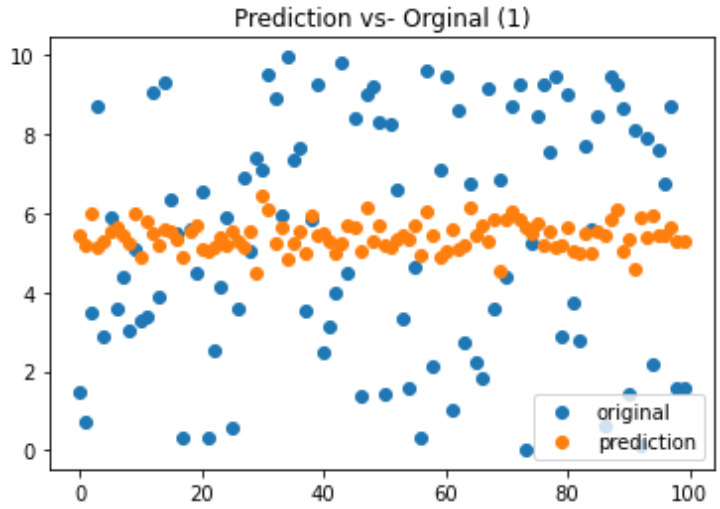
In [38]:
```
BETA_first = (b_one,b_two)
print('beta_1, beta_two= \n',BETA_first)
```

```
beta_1, beta_two=
 (array([-25.47134105]), array([28.1742986]))
```

In [33]:
```
# prediction is the set of predictions
prediction = x@BETA_first
```

6. Plotting

In [34]:
```
plt.title('Prediction vs- Orginal (1)')
plt.scatter(list(range(100)),y , label="original")
plt.scatter(list(range(100)), prediction, label="prediction")
plt.legend()
plt.show()
```



7. Numpy Linealg Lstsq

In [35]:
```
#using np.linalg.lstsq to fund beta
BETA_second =np.linalg.lstsq(x, y, rcond=None)[0]
print(BETA_second)
```

```
[[-25.47134105]
 [ 28.1742986 ]]
```

In [36]:
```
prediction_second = x@BETA_second
```

Now, I will show the graphical result of my first calculation.

In [37]:
```
plt.title('Prediction vs- Orginal (2)')
plt.scatter(list(range(100)),y , label="original")
plt.scatter(list(range(100)), prediction_second, label="prediction")
plt.legend()
plt.show()
```