EXP 4 - DATA FITTING

In [1]:

```
from pylab import*
```

In [2]:

```
from scipy.optimize import curve_fit
```

In [3]:

```
x_data,y_data =loadtxt('data.txt',unpack=True)
```

In [4]:

```
x_data
```

Out[4]:

```
array([8.299, 7.399, 6.899, 6.299, 5.499, 5.199])
```

In [5]:

```
y_data
```

Out[5]:

```
array([3.199, 2.599, 2.25 , 1.809, 1.309, 1.119])
```

In [6]:

```
def linearfunction(x,intercept,slope):
    y = intercept+slope * x
    return y
```

In [8]:

```
linearfunction(1,2,3)
```

Out[8]:

```
5
```

In [9]:

```
a_fit,cov=curve_fit(linearfunction,x_data,y_data)
```

In [10]:

```
print(a_fit)
```

```
[-2.40615359  0.67489826]
```

In [45]:

```
print(cov)
```

```
[[ 2.81352652e-03 -4.15417792e-04]
 [-4.15417792e-04  6.29516283e-05]]
```

In [46]:

```
inter = popt[0]
slope = popt[1]
```

In [47]:

```
d_inter = sqrt(cov[0][0])
d_slope = sqrt(cov[1][1])
```

In [48]:

```
d_slope
```

Out[48]:

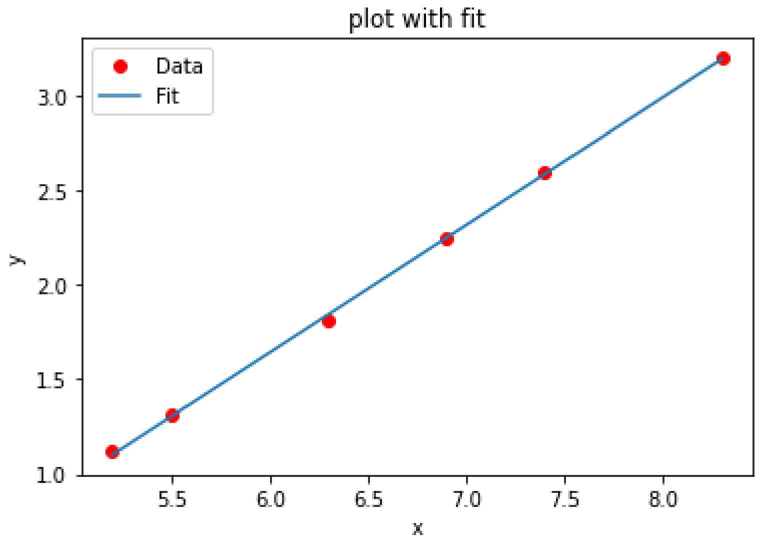
```
0.007934206216589924
```

In [49]:

```
plot(x_data,y_data,'ro',label='Data')
yfit = inter + slope*x_data
plot(x_data,yfit,label='Fit')
legend()
xlabel('x')
ylabel('y')
title('plot with fit')
```

Out[49]:

```
Text(0.5, 1.0, 'plot with fit')
```



In [50]:

```
print(f'The slope = {slope}, with uncertainty {d_slope}')
```

```
The slope = 0.6748982556897472, with uncertainty 0.007934206216589924
```

In [51]:

```
print(f'The intercept = {inter}, with uncertainty {d_inter}')
```

```
The intercept = -2.4061535892966415, with uncertainty 0.053042685870017224
```

DATA FIT WITH ERROR

In [52]:

```
def linearFunc(x,intercept,slope):
 y = intercept + slope * x
 return y
```

In [53]:

```
xdata,ydata,d_y = loadtxt('data_with_er.txt',unpack=True)
```

In [54]:

```
print(xdata)
```

```
[8.213 7.402 6.876 5.491 5.196]
```

In [55]:

```
print(d_y)
```

```
[0.0971 0.0559 0.0708 0.0683 0.0893]
```

In [56]:

```
a_fit,cov=curve_fit(linearFunc,xdata,ydata,sigma=d_y)
```

In [57]:

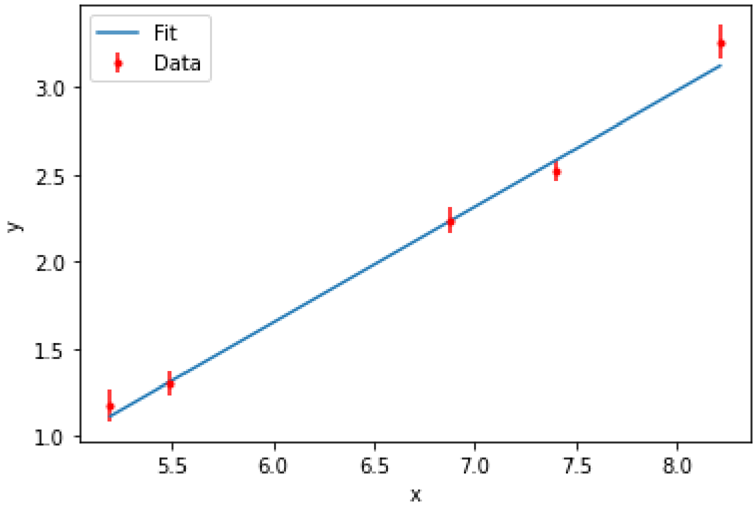
```
inter = a_fit[0]
slope = a_fit[1]
d_inter = sqrt(cov[0][0])
d_slope = sqrt(cov[1][1])
```

In [58]:

```
# Create a graph showing the data.
errorbar(xdata,ydata,yerr=d_y,fmt='r.',label='Data')
# Compute a best fit line from the fit intercept and slope.
yfit = inter + slope*xdata
# Create a graph of the fit to the data. We just use the ordinary plot
# command for this.
plot(xdata,yfit,label='Fit')
# Display a legend, label the x and y axes and title the graph.
legend()
xlabel('x')
ylabel('y')
```

Out[58]:

```
Text(0, 0.5, 'y')
```



In [59]:

```
print(f'The slope = {slope}, with uncertainty {d_slope}')
print(f'The intercept = {inter}, with uncertainty {d_inter}')
```

```
The slope = 0.6656028702881751, with uncertainty 0.03549213604200107
The intercept = -2.3430681719234285, with uncertainty 0.239532487804196
```

In [60]:

```
chisqr = sum((ydata-linearFunc(xdata,inter,slope))**2/d_y**2)
dof = len(ydata) - 2
chisqr_red = chisqr/dof
print(f'Reduced chi^2 = {chisqr_red}')
```

```
Reduced chi^2 = 1.2633310164063059
```

With DIfferefnt values

In [61]:

```
def linearFunc(x,intercept,slope):
 y = intercept + slope * x
 return y
```

In [62]:

```
xdata,ydata,d_y = loadtxt('dwe.txt',unpack=True)
```

In [63]:

```
xdata
```

Out[63]:

```
array([8.213, 7.402, 6.876, 5.491, 5.196, 4.873, 4.422, 3.991])
```

In [64]:

```
ydata
```

Out[64]:

```
array([3.261, 2.52 , 2.239, 1.299, 1.175, 0.911, 0.871, 0.661])
```

In [65]:

```
d_y
```

Out[65]:

```
array([0.071, 0.059, 0.088, 0.083, 0.039, 0.055, 0.033, 0.032])
```

In [66]:

```
a_fit,cov=curve_fit(linearFunc,xdata,ydata,sigma=d_y)
```

In [67]:

```
inter = a_fit[0]
slope = a_fit[1]
d_inter = sqrt(cov[0][0])
d_slope = sqrt(cov[1][1])
```

In [68]:

```
print(f'The slope = {slope}, with uncertainty {d_slope}')
print(f'The intercept = {inter}, with uncertainty {d_inter}')
```

```
The slope = 0.5799850077839609, with uncertainty 0.034895112324111959
The intercept = -1.7346908170011626, with uncertainty 0.18164974724457034
```

In [69]:

```
chisqr = sum((ydata-linearFunc(xdata,inter,slope))**2/d_y**2)
dof = len(ydata) - 2
chisqr_red = chisqr/dof
print(f'Reduced chi^2 = {chisqr_red}')
```

```
Reduced chi^2 = 6.714328193557582
```

Type *Markdown* and LaTeX: α^2