Python Excercises

In [74]:
```
from __future__ import division  #  0 بدلا من 0.5 يساوي 1/2 هذا السطر يجعل
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sympy as sp
from scipy.optimize import curve_fit
import time
from Tkinter import Tk
from tkFileDialog import askdirectory,askopenfilename,askopenfilenames
from ipywidgets import interact
pd.options.display.max_rows = 7   # 7 يجعل اقصى عرض للصفوف في الجدول
%matplotlib inline
```

1 - Solving physics Formula:

find in Hyderogen atom:

- Lyman series ($n_f = 1$)
- Balmer series ($n_f = 2$)
- Paschen series ($n_f = 3$)
- Brackett series ($n_f = 4$)
  giving that:
  $\frac{1}{\lambda} = R(\frac{1}{n_f^2} - \frac{1}{n_i^2})$
  $R = 1.097373 \times 10^7 m^{-1}$

In [2]:
```
R = 1.097373*10**7
for n_f in range(1,5): #  4 الى 1 سوف تأخذ في كل حلقة كبيرة قيمة من nf
    print 'n_f = ',n_f
    for n_i in range(1,8):# ni ستأخذ قيم من 1 الى 7 في كل حلقة صغيرة
        n_i += n_f  #  nf ستكون دائما اكبر من  ni هذا السطر يظمن لك أن
        v = R * ( (1/n_f**2) - (1/n_i**2) )
        lamda = 1/v * 10**9  # التحويل من المتر الى النانو متر
        print 'n = {}, lamda = {:.2f}nm'.format(n_i,lamda)
    print '---------------------------'
```

```
n_f =  1
n = 2, lamda = 121.50nm
n = 3, lamda = 102.52nm
n = 4, lamda = 97.20nm
n = 5, lamda = 94.92nm
n = 6, lamda = 93.73nm
n = 7, lamda = 93.03nm
n = 8, lamda = 92.57nm
--------------------------
n_f =  2
n = 3, lamda = 656.11nm
n = 4, lamda = 486.01nm
n = 5, lamda = 433.94nm
n = 6, lamda = 410.07nm
n = 7, lamda = 396.91nm
n = 8, lamda = 388.81nm
n = 9, lamda = 383.44nm
--------------------------
n_f =  3
n = 4, lamda = 1874.61nm
n = 5, lamda = 1281.47nm
n = 6, lamda = 1093.52nm
n = 7, lamda = 1004.67nm
n = 8, lamda = 954.35nm
n = 9, lamda = 922.66nm
n = 10, lamda = 901.25nm
--------------------------
n_f =  4
n = 5, lamda = 4050.08nm
n = 6, lamda = 2624.45nm
n = 7, lamda = 2164.95nm
n = 8, lamda = 1944.04nm
n = 9, lamda = 1816.93nm
n = 10, lamda = 1735.75nm
n = 11, lamda = 1680.20nm
--------------------------
```

Vectors & Matrics

1- Find the work that has been done on a box, where the Forse and distance are:

$\vec{F} = 2i + 5j - 3k$ (N)
$\vec{d} = -3i + 10j + 12k$ (m)
$W = \vec{F} \cdot \vec{d}$

2- Find the magnetic force on positive charge, that is moving in a magnatic field, given that:

$q = 2x10^{-19}$ (c)
$\vec{v} = (2i - 1.2j - 0.5k) \times 10^4$ (m/s)
$\vec{B} = 0.03i + 0.01j - 0.5k$ (T)
$\vec{F} = q(\vec{v} \times \vec{B})$

3- find the current in the curcuit, given that:

$3i_1 + 5i_2 - i_3 = 5$
$5i_1 - 1i_2 + 4i_3 = 2$
$2i_1 + 7i_2 + 2i_3 = 10$
or rewritten it as a matrix:
A X = b >> $\begin{bmatrix} 3 & 5 & -1 \\ 5 & -1 & 4 \\ 2 & 7 & 2 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \\ 10 \end{bmatrix}$
hence, $X = A^{-1}b$

In [3]:
```
# 1: w = f . d
f = np.array([2,5,-3])
d = np.array([-3,10,12])
w = np.dot(f,d)
print 'w =', w ,'Joles'
```

```
w = 8 Joles
```

In [4]:
```
# 2: F = q (v X B)
q = 2*10**-19
v = np.array([2,-1.2,-0.5]) *10**4
B = np.array([0.03,0.01,-0.5])
f = q*np.cross(v,B)
f
```

Out[4]:
```
array([  1.21000000e-15,   1.97000000e-15,   1.12000000e-16])
```

In [5]:
```
print 'F = {:.2e} i + {:.2e} j + {:.2e} k  (N)'.format(*f)
```

```
F = 1.21e-15 i + 1.97e-15 j + 1.12e-16 k  (N)
```

In [6]:
```
A = np.array([[3,5,-1],
              [5,-1,4],
              [2,7,2]])
b = np.array([[5],
              [2],
              [10]])
A_inv = np.linalg.inv(A)

X = np.dot(A_inv,b)
X
```

Out[6]:
```
array([[-0.04379562],
       [ 1.19708029],
       [ 0.8540146 ]])
```

In [7]:
```
print 'i_1 = ',X[0][0]
print 'i_2 = ',X[1][0]
print 'i_3 = ',X[2][0]
```

```
i_1 =  -0.043795620438
i_2 =  1.19708029197
i_3 =  0.85401459854
```

2- نابض يهتز بحركة توافقية بسيطة تعطى ازاحته كالاتي

$$y(t) = A\cos(\omega t + \phi)$$

أوجد سرعة النابض وتسارعه

In [8]:
```
sp.init_printing(use_latex='mathjax')
```

In [9]:
```
A,omega,t,phi = sp.symbols('A omega t phi')
y = A * sp.cos(omega*t+phi)
y
```

Out[9]: $A\cos\left(\omega t + \phi\right)$

In [10]:
```
# The Velocity is:
v = sp.diff(y,t)
v
```

Out[10]: $-A\omega\sin\left(\omega t + \phi\right)$

In [11]:
```
# The Acceleration is:
a = sp.diff(v,t)
a
```

Out[11]: $-A\omega^2\cos\left(\omega t + \phi\right)$

3) ما هي الطاقة التي اكتسبها المكثف بعد أن تم شحنه من 3 فولت إلى 7 فولت

$$E = \int_3^7 CV dV$$

Where: $C = 2.5\mu F$

In [12]:
```
# The Energy is:
c,v = sp.symbols('C V')
E = sp.Integral(c*v,[v,3,7])
E
```

Out[12]:
$$\int_{3}^{7} CV\,dV$$

In [13]:
```
E.doit()
```

Out[13]:
$$20C$$

In [14]:
```
_.subs(c,2.5e-6) # هذه العلامة تعني العملية السابقة _
```

Out[14]:
$$5.0 \cdot 10^{-5}$$

4 - Photon Energy:

$E = \frac{hc}{\lambda}$

You have to do the following:

- Import data from txt file
- Analyze the data and calculate the photon Energy
- Creat a table then put the date in it.
- Plot wavelength vs Photon Energy
- Save the figure as (Image) and the table as Excel file

In [15]:
```
Tk().withdraw()
path = askopenfilename()
L_nm = np.loadtxt(path)  # lambda.txt
```

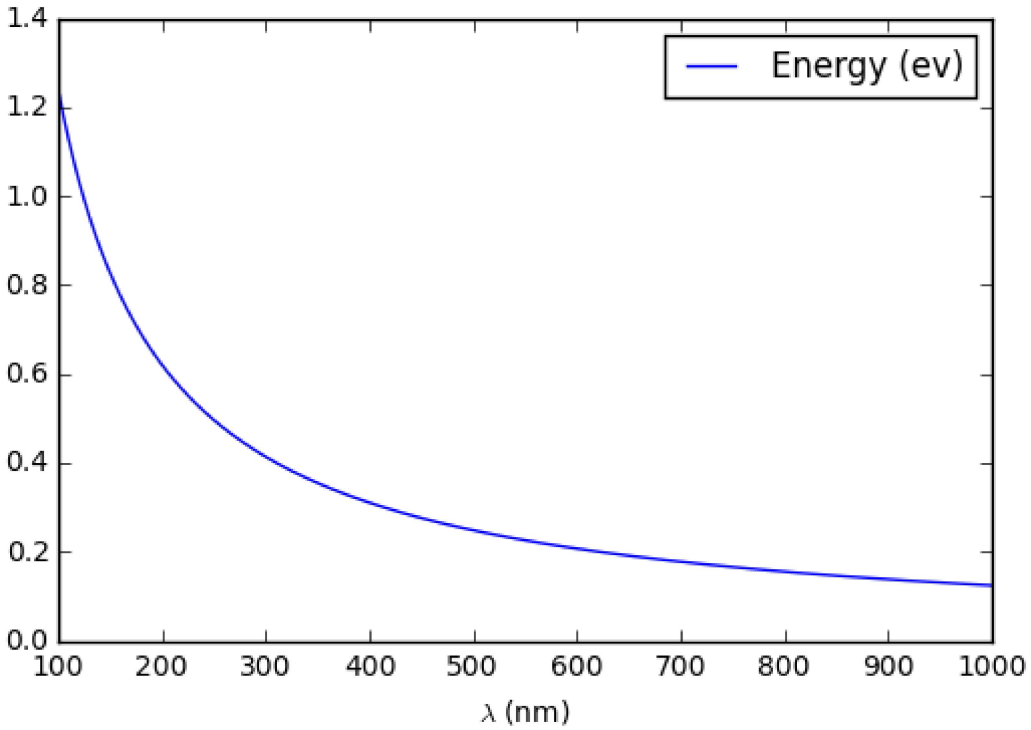
In [16]:
```
h = 6.626e-34
c = 3.0e8
e = 1.602e-19
L_m = L_nm * 10.0e-9  #التحويل من النانومتر الى المتر

E_j = (h * c)/L_m  # الطاقة بالجول
E_ev = E_j/e       # الطاقة بالإلكترون فولت

df = pd.DataFrame({'$\lambda$ (nm)':L_nm , 'Energy (J)':E_j, 'Energy (ev)':E_ev})
df
```

Out[16]:

	λ (nm)	Energy (J)	Energy (ev)
0	100	1.987800e-19	1.240824
1	101	1.968119e-19	1.228539
2	102	1.948824e-19	1.216494
...	...	...	...
898	998	1.991784e-20	0.124331
899	999	1.989790e-20	0.124207
900	1000	1.987800e-20	0.124082

901 rows × 3 columns

In [17]:
```
df.plot('$\lambda$ (nm)','Energy (ev)') # رسم بياني مباشرة من الجدول
plt.savefig('C:/Users/User/Desktop/wavelength_Energy.png',dpi = 200)
plt.show()
```



In [18]:
```
df.to_excel('C:/Users/User/Desktop/photon energy.xlsx')
```

3- تجربة إيجاد الجاذبية الأرضية من خلال السقوط الحر

في تجربة السقوط الحر نقوم بحساب زمن سقوط الاجسام في مسافة محددة، ثم نقوم برسم بياني بين المسافة المقطوعة على محور x و(مربع) الزمن على محور y. وبعد ذلك نقوم باستخراج الميل slope

$$t^2 = \frac{2d}{g}$$

$$g = \frac{2}{slope}$$

In [1]:

```
# اخذ النتائج يدويا
def results_manually():
    distence = []
    Time = []
    while True:
        raw_input('Press Enter to Start Time ') # اضغط انتر لبداية التجربة
        t0 = time.time()    #تسجيل بداية الزمن للتجربة
        raw_input('Press Enter to End Time ') # اضغط انتر عند انتهاء التجربة
        t = time.time()-t0    #  تسجيل الزمن المتسغرق من البداية الى النهاية
        d = input('d = ')    # تسجيل المسافة المقطوعة
        if d == 0:   # اذا انتهيت من اخذ القراءات تضع المسافة ب صفر حتى تخرج من البرنامج
            break
        distence.append(d)
        Time.append(t)
    return (distence,Time)  # اخيرا، المسافات و الازمنة على شكل قائمة لكي تستفيد منها في الحسابات لاحقا


# جلب النتائج من ملف خارجي
def import_results(path):
    distence,time = np.loadtxt(path).T # <=> distence,time = np.loadtxt(path)[:,0], np.loadtxt(path)[:,1]
    return (distence,time)



# حسابات تجربة السقوط الحر
def free_fall(distence,time,path,name = 'Free fall on Earth'):
    # path هو المكان الذي تريد حفظ النتائج فيه
    #name اسم ملفات النتائج

    #t^2
    t_2 = np.array(time)**2

    # تكوين جدول
    df = pd.DataFrame({'Distence (m)':distence,'t (sec)':time , 't^2 (sec)^2': t_2 })
    df.to_excel(path+'/%s.xlsx' % name)

    # fitting
    fit = np.polyfit(distence , t_2 , 1)

    # النتائج
    slope = fit[0]
    print 'Slope = %s sec^2/m'%slope
    g = 2 / slope
```

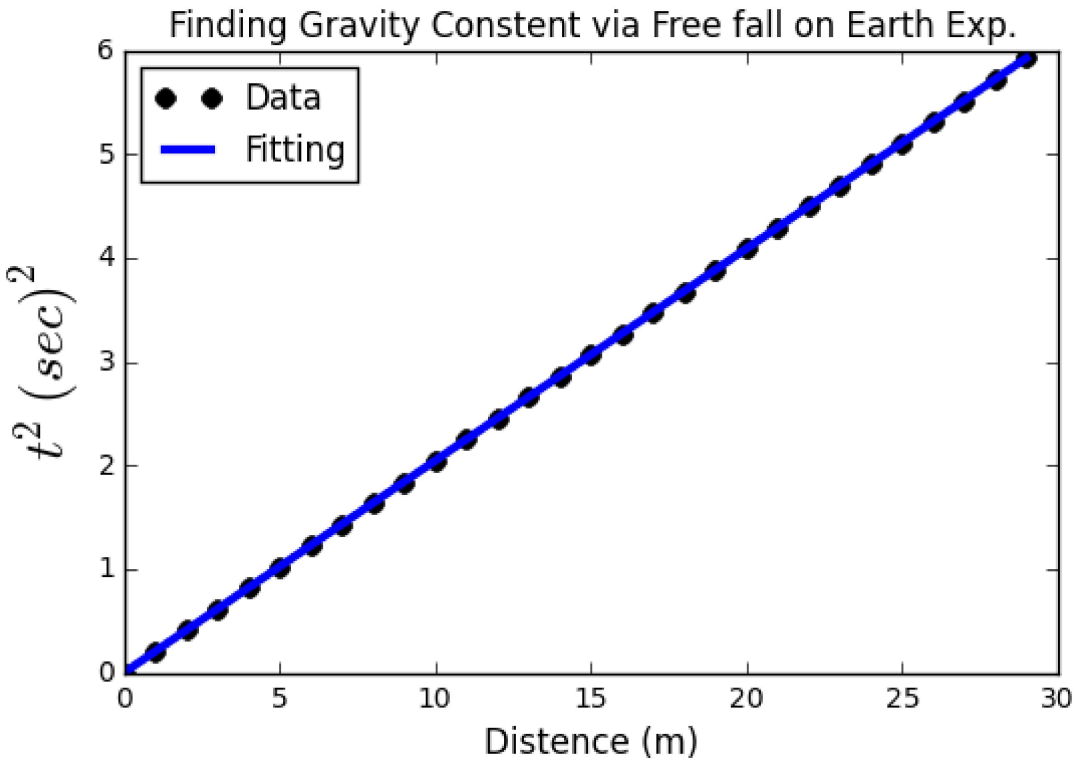
```
print 'g = %s m/s^2' %g

# الرسم البياني
plt.plot(distence,t_2 , 'ko' , label = 'Data' , markersize = 7)
y = np.poly1d(fit)
plt.plot(distence, y(distence),'b',lw = 3, label = 'Fitting')
plt.title('Finding Gravity Constent via %s Exp.'%name, fontsize = 12)
plt.xlabel('Distence (m)' , fontsize = 12)
plt.ylabel('$t^2 (sec)^2$' , fontsize = 20)
plt.legend(loc = 0)
plt.xlim(0)
plt.savefig(path+'/%s.png'%name, dpi = 800)
plt.show()
```

<font-size: 2em>

In [ ]:
```
#1 عمل التجربة يدويا
# اولاً اختيار مكان حفظ الملف
Tk().withdraw()
save_path = askdirectory()

# تسمية الملف
name = raw_input('Experiment name: ')

# ثانيا، الحسابات
d,t = results_manually()
free_fall(d ,t ,save_path ,name = name)
```

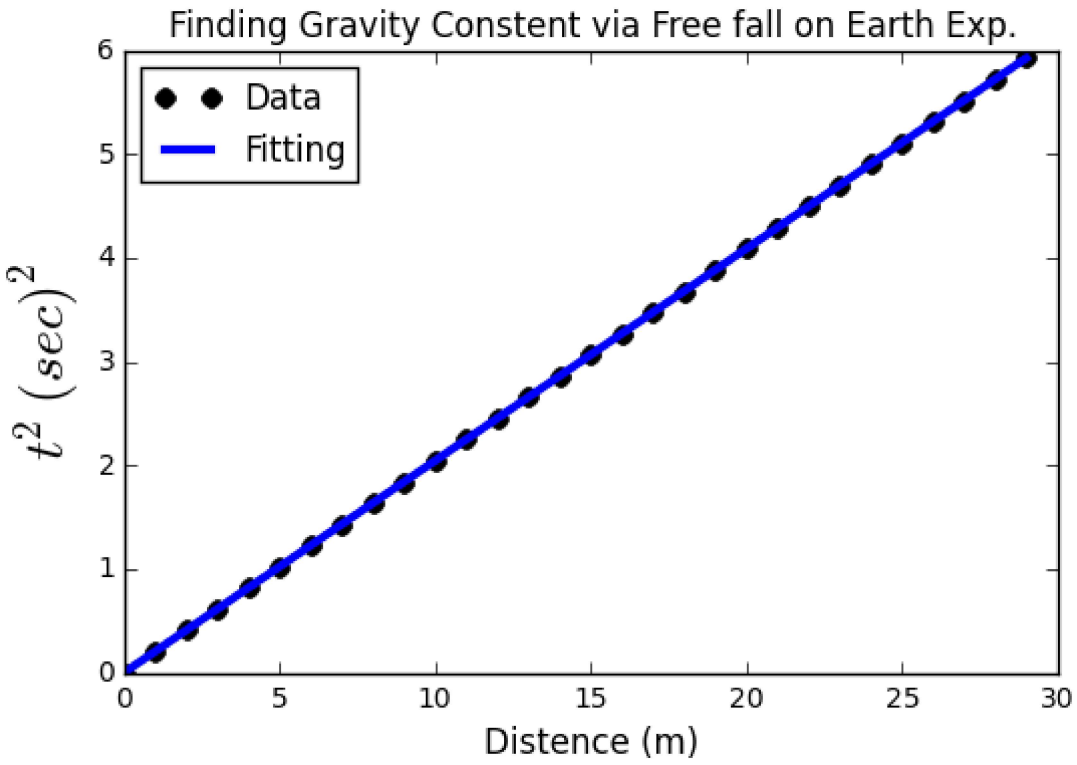
In [26]:
```
# اخذ البيانات من مصدر خارجي
# في حال وجود ملف واحد فقط

# اولاً اختيار مكان حفظ الملف
Tk().withdraw()
save_path = askdirectory()

# ثانيا، اختيار ملف البيانات
Tk().withdraw()
file_path = askopenfilename()

# ثالثا، تسمية الملف تلقائيا من المسار
name = file_path.split('/')[-1]
name = name[: name.index('.')]

# رابعا اخذ البيانات
d,t = import_results(file_path)

# أخيرا الحسابات
free_fall(d ,t ,save_path , name = name)
```

```
Slope = 0.204290091931 sec^2/m
g = 9.79 m/s^2
```

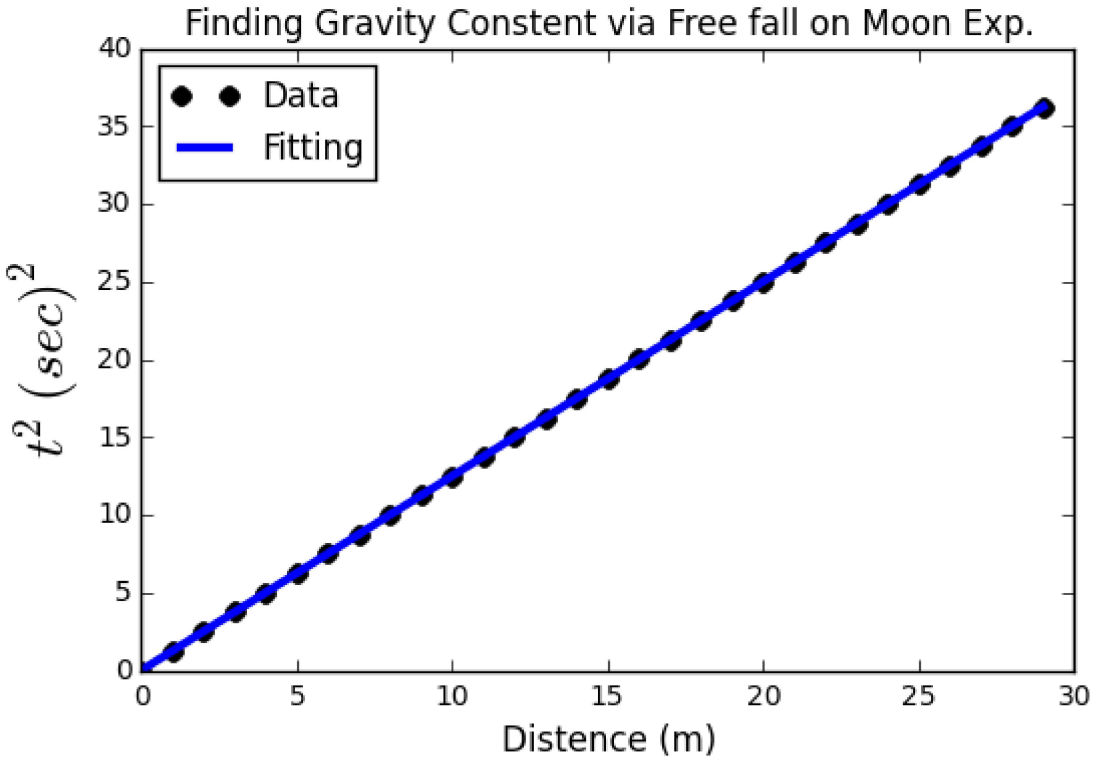


In [27]:
```
# اخذ البيانات من مصدر خارجي
# في حال وجود ملف واحد أو أكثر

# اولاً اختيار مكان حفظ الملف
Tk().withdraw()
save_path = askdirectory()

# ثانيا، اختيار ملفات البيانات
Tk().withdraw()
file_paths = askopenfilenames()

for file_path in file_paths:
    # ثالثا، تسمية الملف تلقائيا من المسار
    name = file_path.split('/')[-1]
    name = name[: name.index('.')]

    # رابعا اخذ البيانات
    d, t = import_results(file_path)

    # أخيرا الحسابات
    free_fall(d ,t ,save_path, name = name)
```

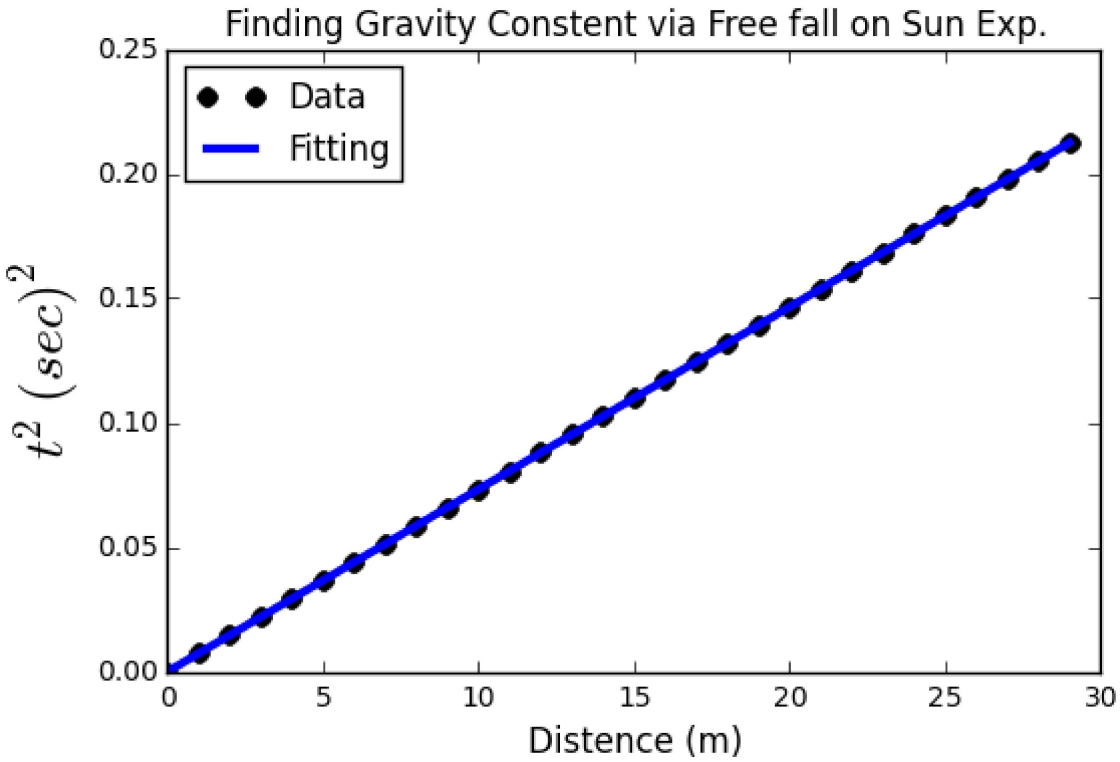
```
Slope = 0.204290091931 sec^2/m
g = 9.79 m/s^2
```




```
Slope = 1.25 sec^2/m
g = 1.6 m/s^2
```




```
Slope = 0.00732600732601 sec^2/m
g = 273.0 m/s^2
```

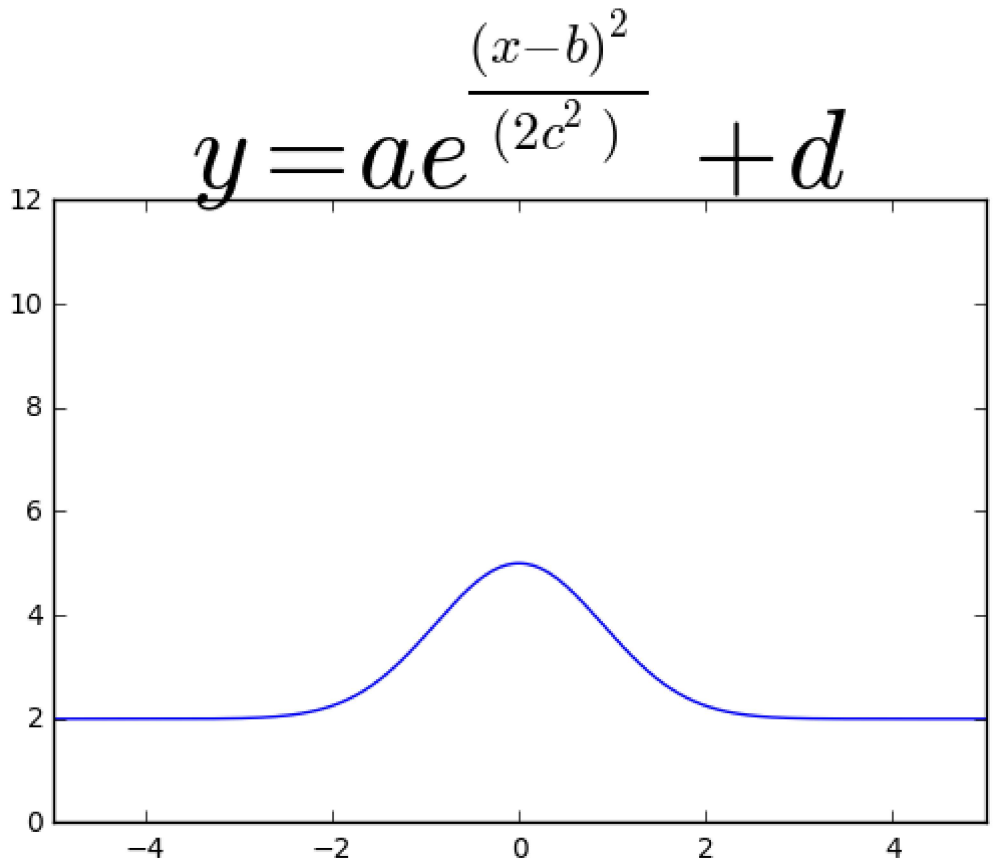


تجربة اثر الاضاءة لمصباح مع اختلاف درجة الحرارة

Quick Review about Gaussian function:

In [70]:
```
def gaussian(x,a,b,c,d): # دالة جاوس
    y = a * np.exp(-(x-b)**2/(2*c**2)) + d
    return y

def gausinteract(a,b,c,d):  # دالة ترسم معادلة جاوس
    x = np.linspace(-10,10,1000)
    y = gaussian(x,a,b,c,d)
    plt.plot(x,y)
    plt.title('$y = a e^{\\frac{(x-b)^2}{(2c^2)}}+d $',fontsize = 40)
    plt.ylim(0,12)
    plt.xlim(-5,5)
    plt.show()
```

In [129]:
```
interact(gausinteract, a = (0,10,1), b = (-5,5,1),c = (0.1,2,0.1),d = (0,5,1)) # تجعل الدالة تفاعلية
```



In [2]:
```
Tk().withdraw()
path = askopenfilename()
```

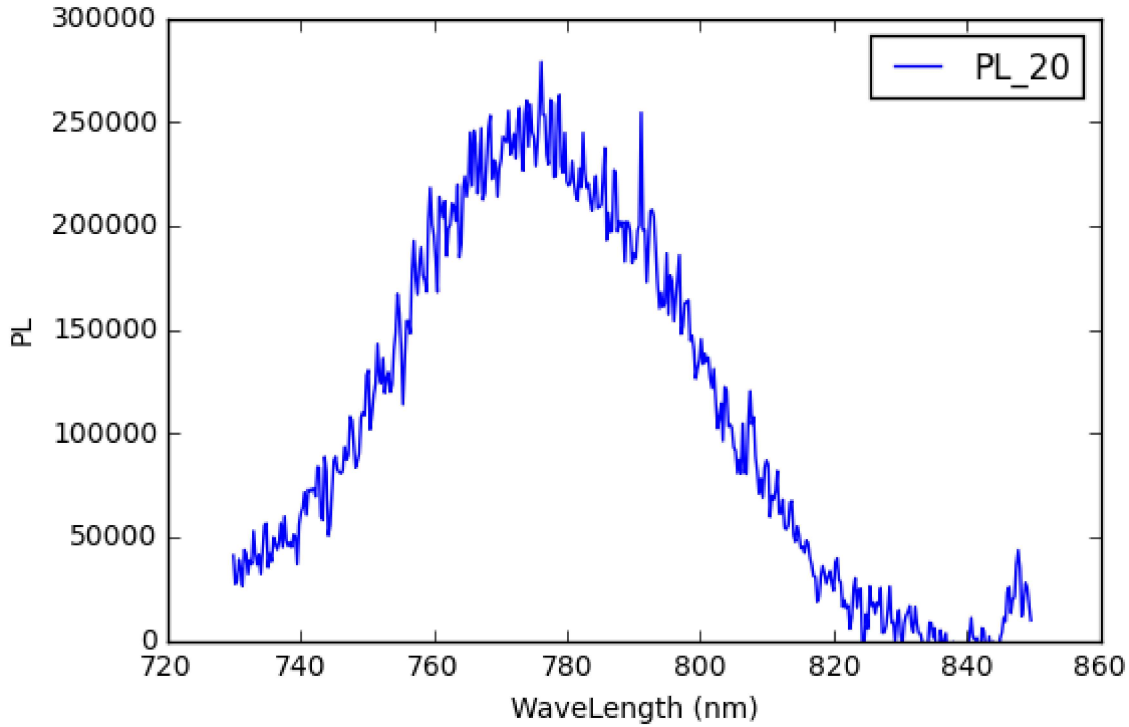
In [3]:
```
df = pd.read_excel(path) # Temp.xlsx
df
```

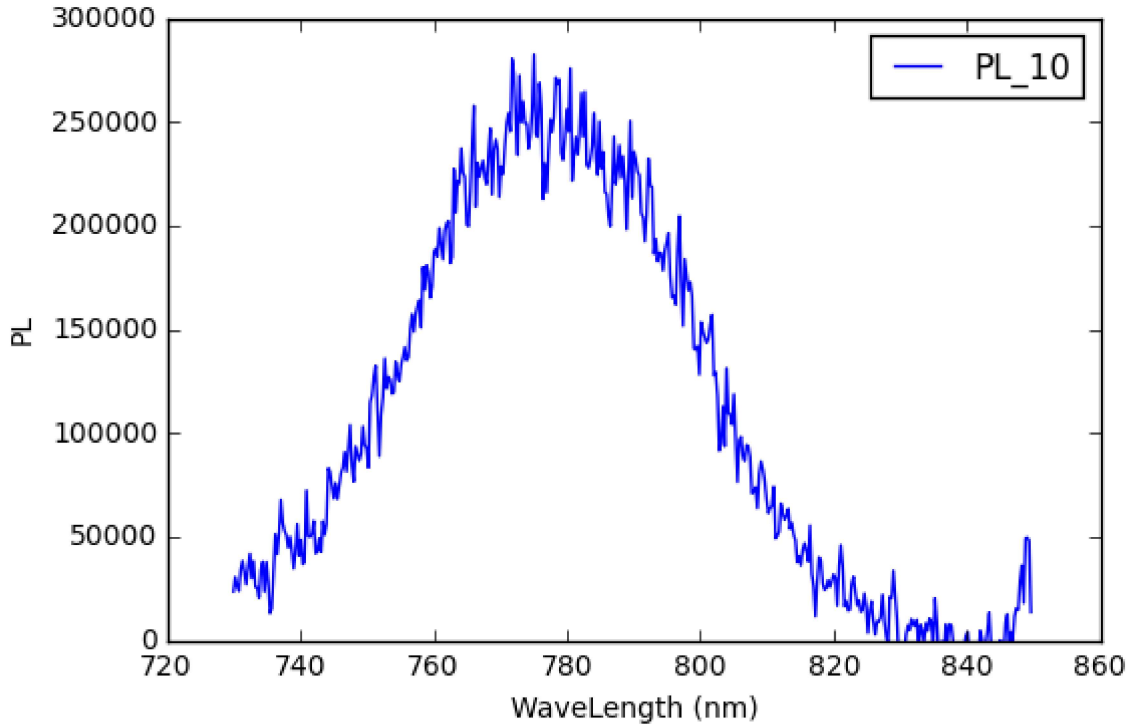
Out[3]:

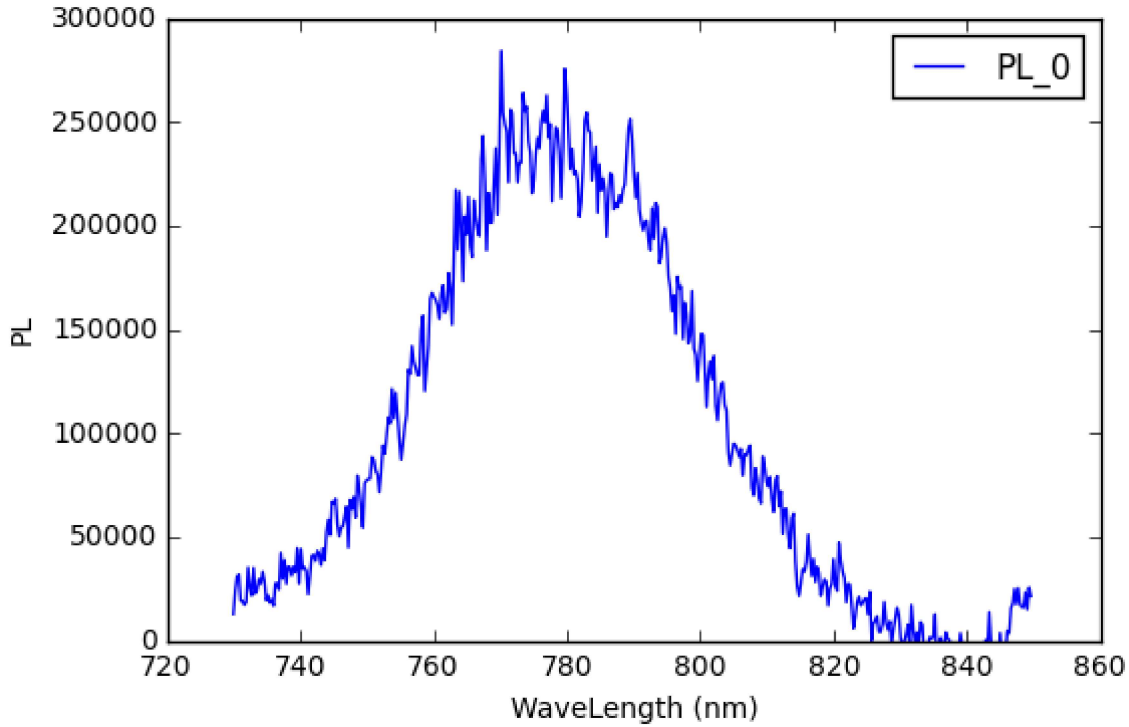
	L (nm)	PL_0	PL_-10	PL_10	PL_-20	PL_20
0	729.97430	12791.452146	6640.200792	23518.935583	12095.9447	41078.697143
1	730.24786	23921.452146	18798.200792	30575.935583	28391.9447	27318.697143
2	730.52142	30802.452146	16123.200792	26013.935583	14213.9447	28512.697143
...	...	...	...	...	...	...
437	849.05475	14998.452146	15195.200792	49561.935583	22477.9447	25640.697143
438	849.32593	25573.452146	17103.200792	47994.935583	20377.9447	17091.697143
439	849.59717	21373.452146	15361.200792	13656.935583	7144.9447	9799.697143

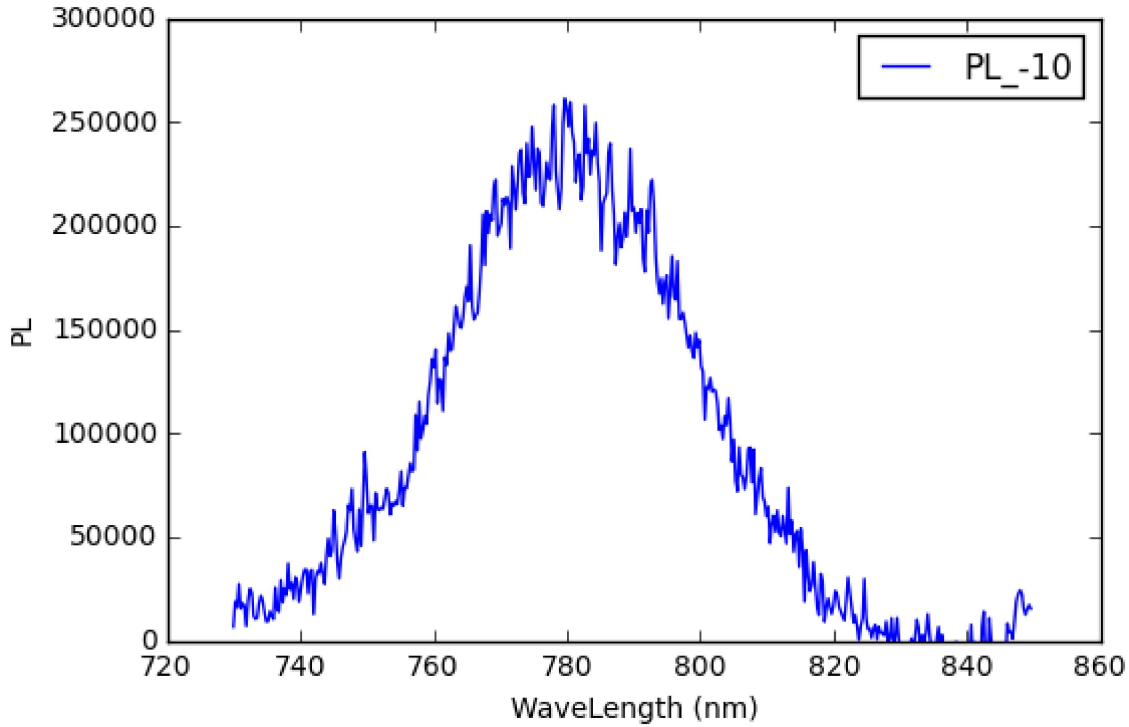
440 rows × 6 columns

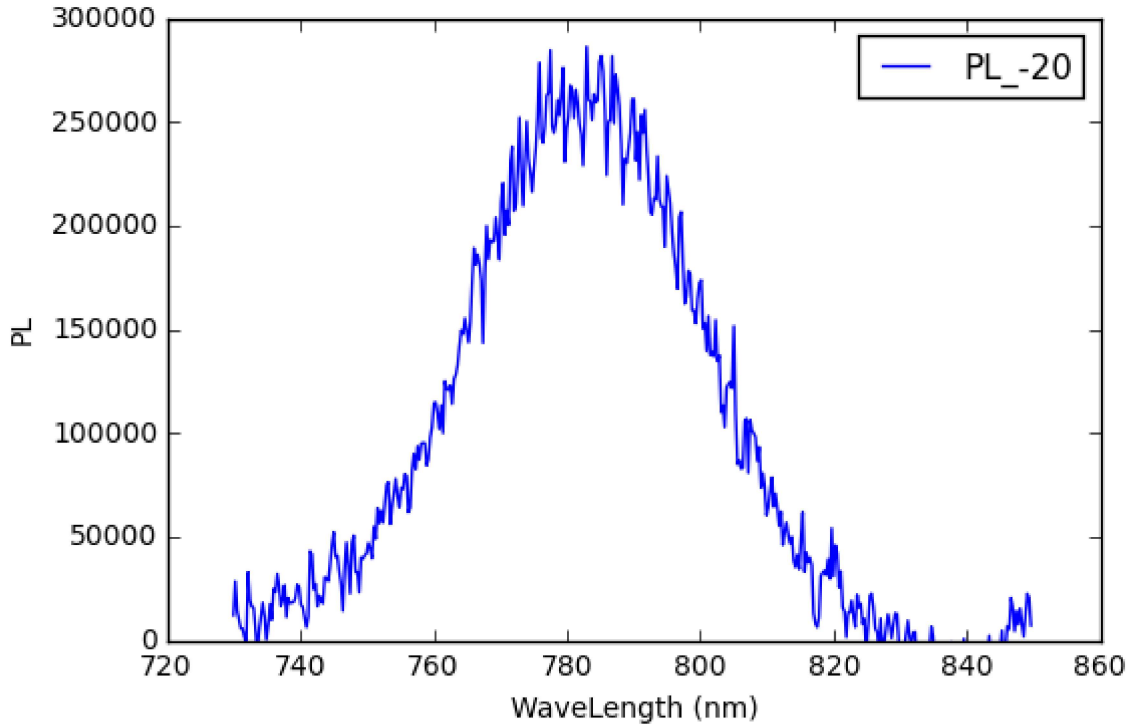
In [10]:
```
x = df['L (nm)'] # دائما محور x ثابت
for y in ['PL_20','PL_10','PL_0','PL_-10','PL_-20']: # محور y يتغير مع كل حلقة
    plt.plot(x,df[y],label = y)
    plt.ylim(0)
    plt.xlabel('WaveLength (nm)')
    plt.ylabel('PL')
    plt.legend()
    plt.show()
# قمنا هنا برسم كل رسمة على حدة،
```


PL_20
PL
WaveLength (nm)
300000
250000
200000
150000
100000
50000
0
720
740
760
780
800
820
840
860

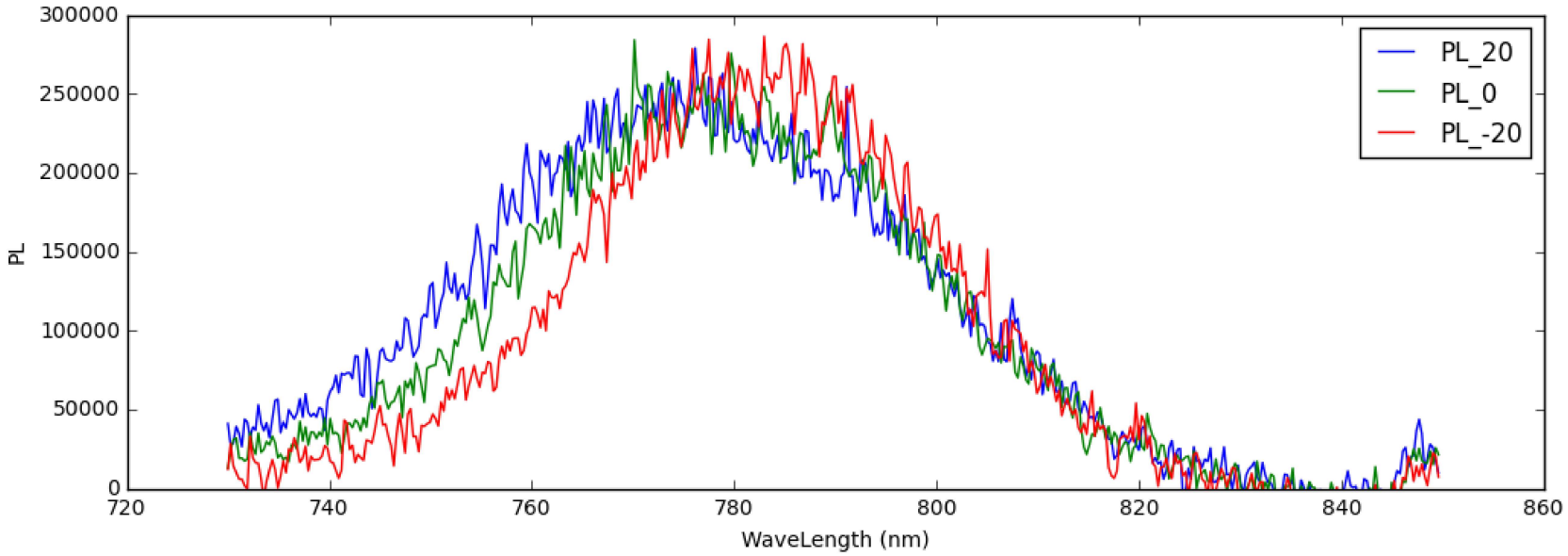

PL_10
PL
WaveLength (nm)
300000
250000
200000
150000
100000
50000
0
720
740
760
780
800
820
840
860


PL_0
PL
WaveLength (nm)
300000
250000
200000
150000
100000
50000
0
720
740
760
780
800
820
840
860


PL_-10
PL
WaveLength (nm)
300000
250000
200000
150000
100000
50000
0
720
740
760
780
800
820
840
860


PL_-20
PL
WaveLength (nm)
300000
250000
200000
150000
100000
50000
0
720
740
760
780
800
820
840
860

In [26]:
```
x = df['L (nm)']
plt.figure(figsize=(12,4))
for y in ['PL_20','PL_0','PL_-20']:
    plt.plot(x,df[y],label = y)
plt.ylim(0)
plt.xlabel('WaveLength (nm)')
plt.ylabel('PL')
plt.legend()
plt.show()
# قمنا هنا برسم 3 رسمات في اطار واحد
```



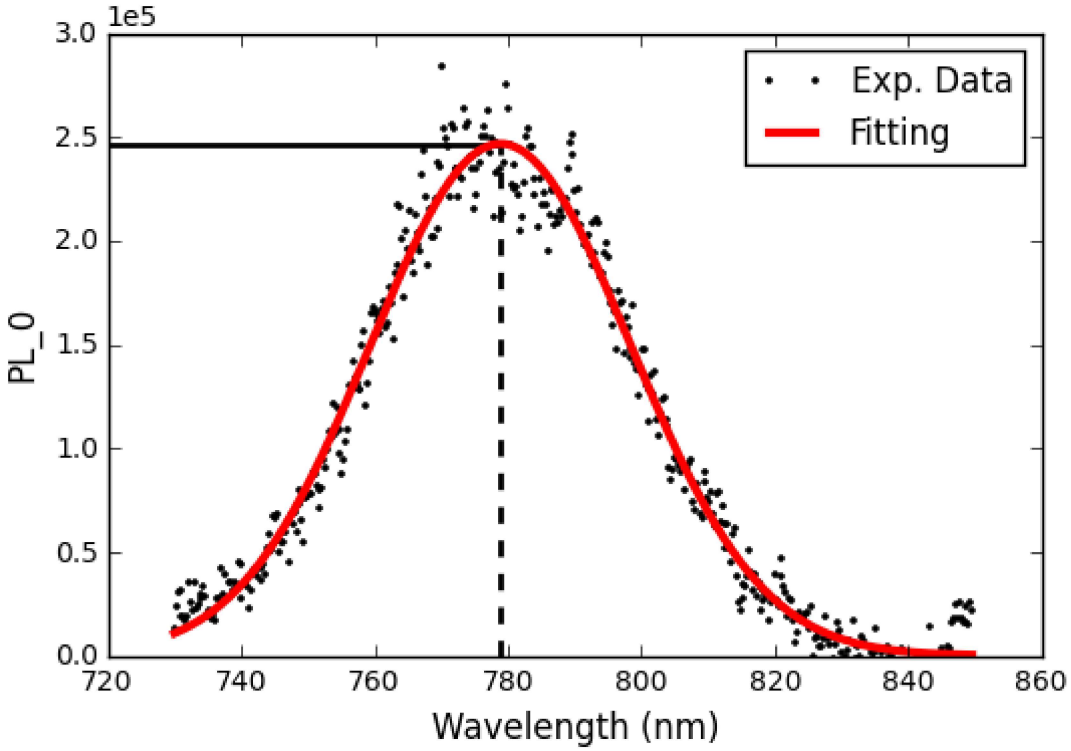
In [76]:
```
# d والارتفاع c و العرض b و مكان القمة a لاستخراج المعلومات المهمة من الرسومات البيانية مثل القمة
# كالاتي fitting يجب ان نعمل
x,y = df['L (nm)'],df['PL_0']
p0 = [max(y),780,20,0] # بالترتيب a,b,c,d تخمين لقيم
fit,cov = curve_fit(gaussian,x,y,p0)
fit
```

Out[76]:
```
array([  2.46721126e+05,   7.78874945e+02,   1.95259920e+01,
        -7.89471463e+00])
```

In [77]:
```
a,b,c,d = fit
print 'a = ' , a
print 'b = ' , b
print 'c = ' , c
print 'd = ' , d
print 'FWHM = ' , 2.35 * c  # عرض نصف ارتفاع القمة
```

```
a =  246721.125545
b =  778.874944629
c =  19.5259919749
d =  -7.89471462657
FWHM =  45.8860811411
```

In [120]:
```
def plot(x,y,fit): # x,y,fitting دالة تقوم بالرسم البياني بسرعة بمجرد ادخالك قيمة
    plt.plot(x,y, 'ko',label = 'Exp. Data', markersize = 2)
    plt.plot(x,gaussian(x,*fit) ,'r' ,lw = 3 , label = 'Fitting')
    plt.xlabel('Wavelength (nm)', fontsize = 12)
    plt.ticklabel_format(axis = 'y',scilimits = (0,0))
    plt.legend(loc = 0)
    plt.ylim(0)

plot(x,y,fit)
plt.ylabel('PL_0', fontsize = 12)
plt.vlines(b,0,a,lw = 2 ,linestyles='--')
# x, ymin, ymax تقوم برسم خط عمودي يجب أن تدخل قيمة
plt.hlines(a,720,b,lw = 2,linestyles='-')
# y, xmin, xmax تقوم برسم خط افقي يجب أن تدخل قيمة
plt.show()
```



In [128]:
```
# لكي نقوم بعمل تحليل كامل للبيانات مرة واحدة يجب ان نكتب التالي
Tk().withdraw()
path = askopenfilename() # اختيار مكان الملف الذي به البيانات
df = pd.read_excel(path)
save_path = askdirectory() # اختيار مكان الحفظ

PL = ['PL_20','PL_10','PL_0','PL_-10','PL_-20']
A = []
B = []
C = []
D = []
FWHM = []

x = df['L (nm)'] # محور x دائما ثابت
for pl in PL:
    y = df[pl]
    p0 = [max(y),780,20,0] # تخمين
    fit,cov = curve_fit(gaussian,x,y,p0)
    a,b,c,d = fit
    fwhm = 2.35 * c
    A.append(a)
    B.append(b)
    C.append(c)
    D.append(d)
    FWHM.append(fwhm)

    plot(x,y,fit) # نستفيد من الدالة التي عرفناها سابقا لكي تختصر علينا الرسم
    plt.ylabel(pl,fontsize = 12)
    plt.savefig(save_path + '/%s.png' % pl ,dpi = 200)
    plt.clf()

res = pd.DataFrame({'PL':PL , 'a':A , 'b (nm)':B, 'c (nm)':C , 'd':D , 'FWHM (nm)':FWHM})
res = res[['PL','a','b (nm)','c (nm)','d','FWHM (nm)']]
res.to_excel(save_path + '/PL with Temperature.xlsx')
```

```
<matplotlib.figure.Figure at 0x170f4be0>
```