In [41]:
```
reset()

# Problem 1

# a
distr = RealDistribution('gaussian', 1)
z1 = (80-78)/8
a1 = 1 - distr.cum_distribution_function(z1)
show(a1)

# b
z2 = (40-78)/8
a2 = distr.cum_distribution_function(z2)
show(a2)

# c
z3 = (55-78)/8
z4 = (75-78)/8
a3 = distr.cum_distribution_function(z4) - distr.cum_distribution_function(z3)
show(a3)

# d
var('x')
f = -0.5 == (x-78)/8
show(solve(f,x))

# e
sample2 = [distr.get_random_element()+78 for j in range (500)]
# print(n(mean(sample2)))
# print(n(variance(sample2)))
histogram(sample2, bins=50)
```

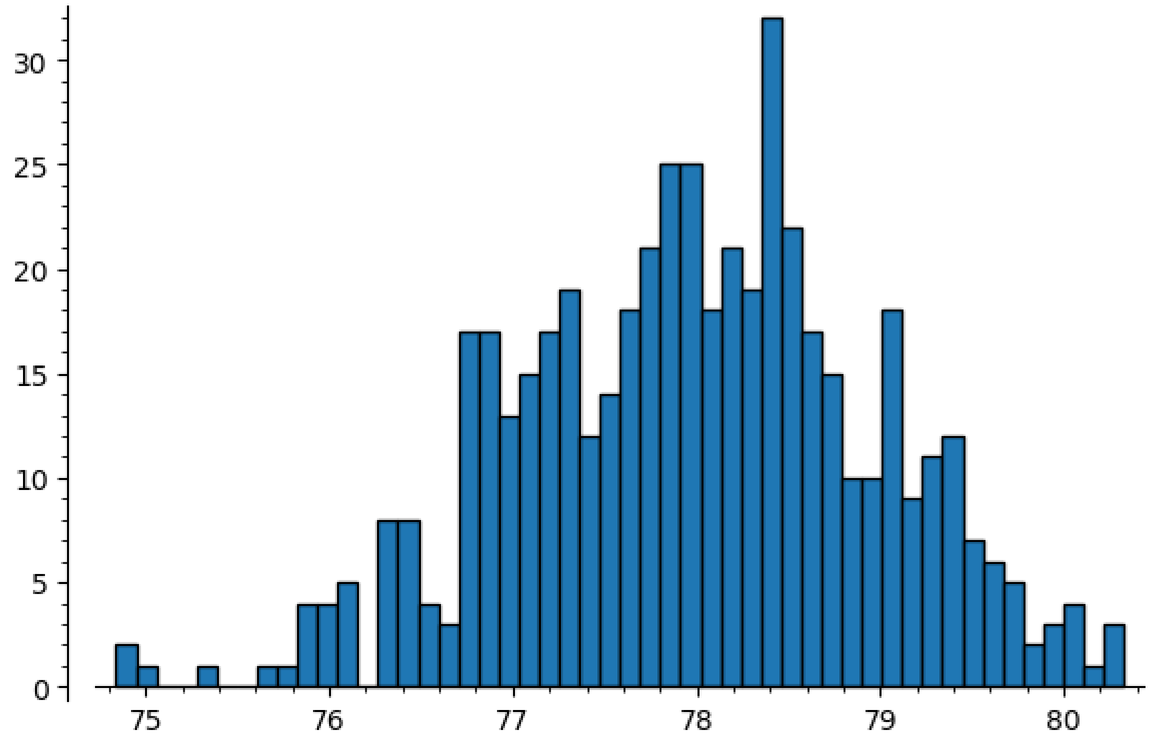
0.4012936743170763

$1.0170832425687034 \times 10^{-06}$

0.3518100958373302

$[x = 74]$

Out[41]:



In [42]:
```
reset()

# Problem2
file = open("project5-data.txt", "r")
file.readline()  # don't read the first line
L = []
for x in file:
    x = x.rstrip('\n')
    temp = x.split(',')
    L.append((temp[0], temp[1]))
file.close()
# print(L)

# a
var('s,x,m')
f(x) = 1/(s*sqrt(2*pi))*exp((-1/2)*((x-m)/s)**2)
# show(f(x))
data_fit = find_fit(L,f,solution_dict=True)
show(data_fit)

# b
plot(f.subs(data_fit),(x,0,50)) + points(L,size=50,color='red')
fxx = diff(f.subs(data_fit),x,2)
# print(f.diff(2))
points = solve(fxx==0,x,solution_dict=True)
points[0][x]
integral(f.subs(data_fit), x, points[1][x], points[0][x])
```

$\{m : 30.0, s : 7.000000000000004\}$

Out[42]:
```
0.00010270256461965642*erf(1/2*sqrt(2))*e^(450/49)
```

In [43]:

```
reset()

# Problem3

def problem3_a(G):
    A = G.adjacency_matrix()
#     show(A)
    size = len(G.vertices())
    D = matrix(size)
    for i in range(size):
        count = 0
        for j in range(size):
            if A[i][j] == 1:
                count += 1
        D[i,i] = count
#     show(D)
    return D-A

G = graphs.PetersenGraph()
show(problem3_a(G))
```

$$\begin{pmatrix} 3 & -1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 3 & -1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & 0 & 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & -1 & 3 & 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 0 & 3 & 0 & -1 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 3 & 0 & -1 & -1 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 3 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & -1 & -1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & 0 & 3 \end{pmatrix}$$

In [44]:
```
reset()
M = matrix([[2,-1,-1,0,0,0],[-1,3,-1,0,-1,0],[-1,-1,4,-1,-1,0],[0,0,-1,2,0,-1
],[0,-1,-1,0,3,-1],[0,0,0,-1,-1,2]])
def problem3_b(L):
    matrix_d = []
    length_l = len(L[0])
    for i in range(length_l):
        vector_d = [0 for i in range(length_l)]
        for j in range(length_l):
            if i == j:
                vector_d[i] = L[i,j]
        matrix_d.append(vector_d)
    D = matrix(matrix_d)
    A = Graph(D - L, format='adjacency_matrix')
    return A

show(problem3_b(M))
```