**HW4**

1. For this problem, we will re-work homework 2*a* using the scipy stats module rather than the Simpson method to find:

P(x<1|N(0,1)): probability x<1 given a normal distribution of x with μ=0, σ=1

P(x>μ+2σ|N(175, 3))

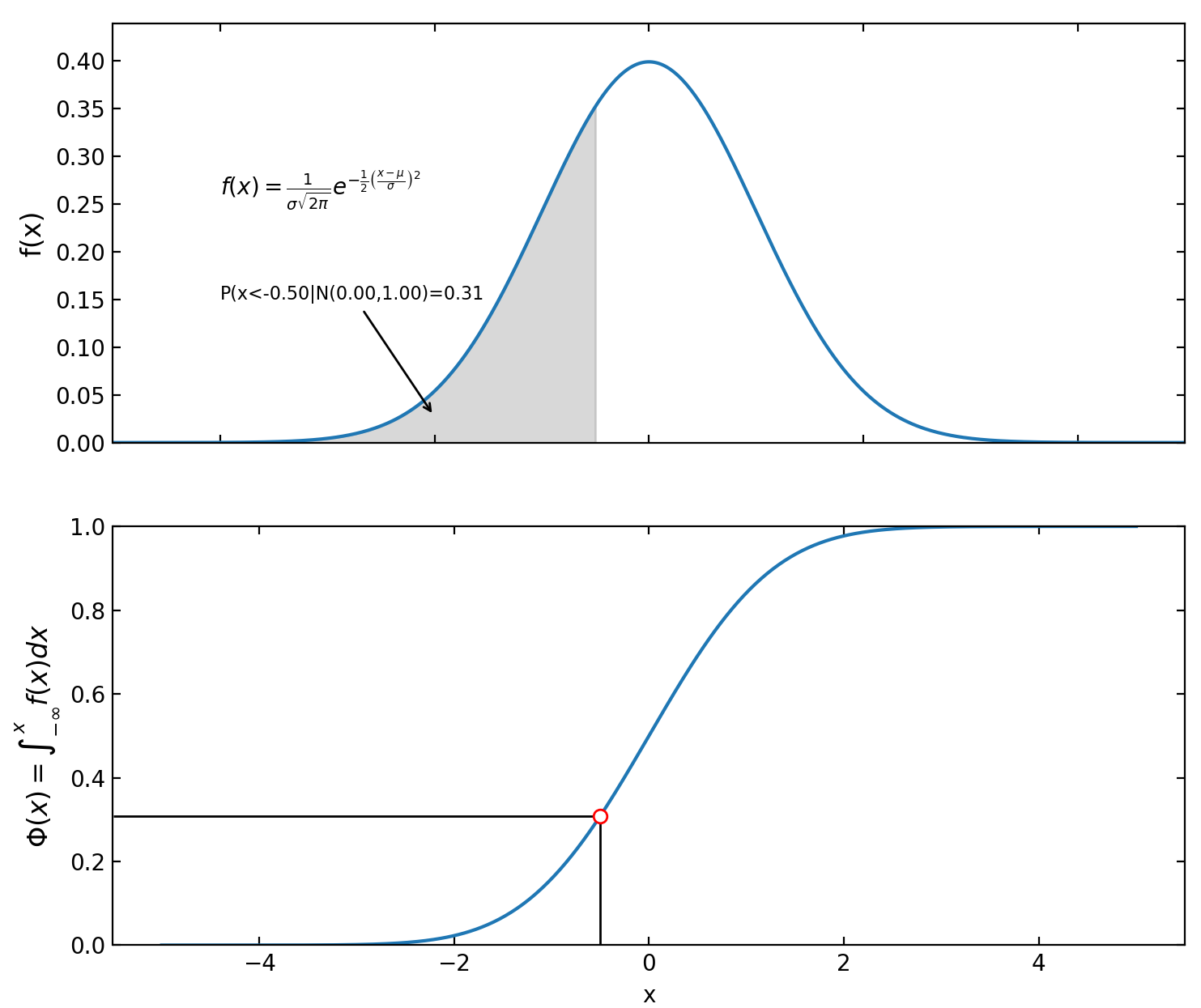
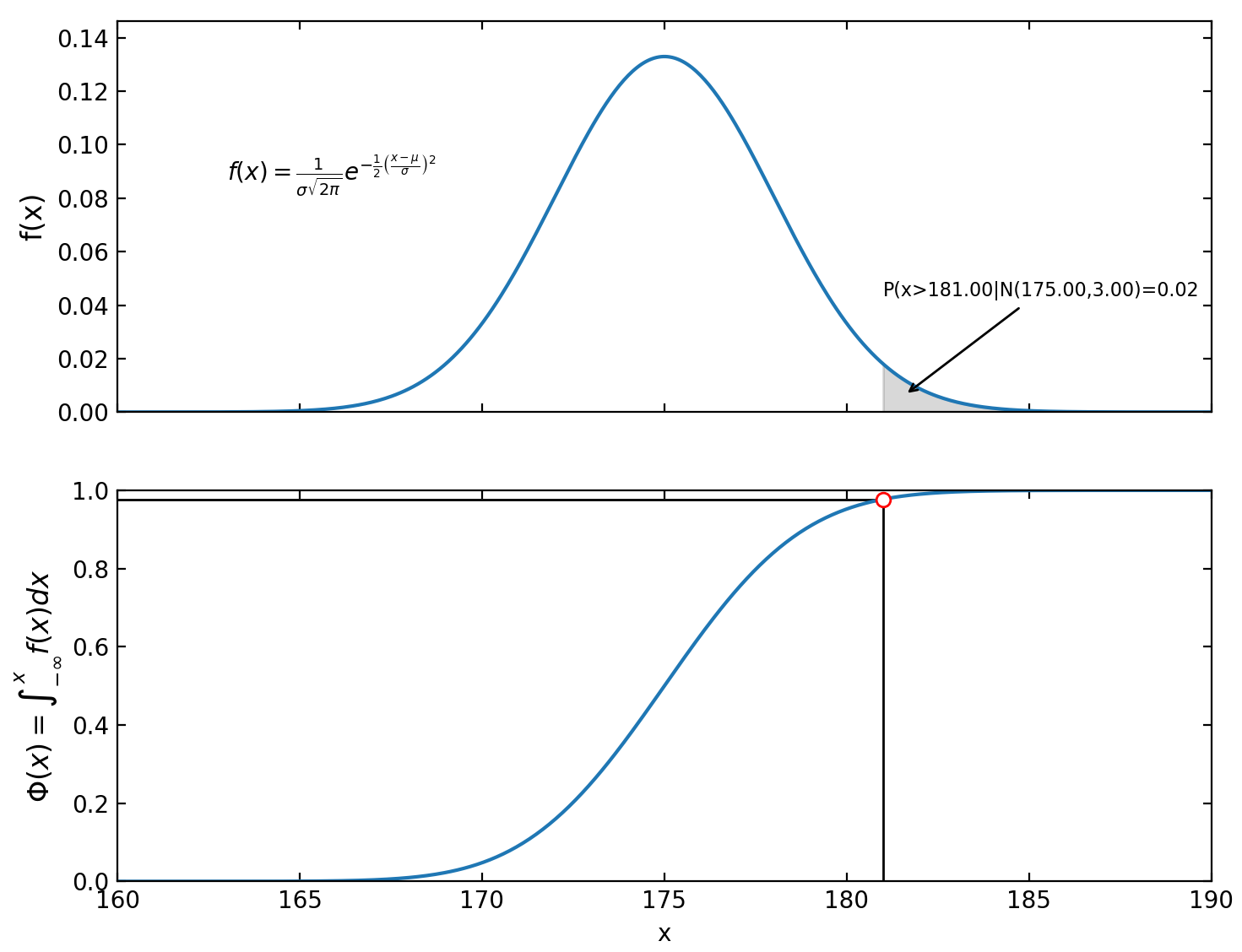
Specifically, you should import as: from scipy import stats

Within stats you should explore the functions stats.norm().pdf and stats.norm().cdf , which refer to the probability density function and cumulative distribution functions, respectively.

Rather than printing your findings to the console, we will use matplotlib.pyplot to produce nicely formatted plots such as shown below. Additional requirements are:

* You should use numpy arrays for all of your work on this problem where arrays are needed.

Note: a code stem for HW4a.py is available for download.

1. Re-work problem b) from homework 2, but use fsolve rather than the secant method to find the roots.
2. Re-work problem c) from homework 2, but use numpy and scipy rather than the Cramer function to solve the matrix equations.

**HW4a.py (This is a stem with #JES MISSING CODE when I broke the working program)**

# region imports  
import matplotlib.pyplot as plt  
import numpy as np  
from scipy import stats  
# endregion

# region functions  
def main():  
 '''  
 Calculates P(x<1|N(0,1)) and P(x>μ+2σ|N(175, 3)) and displays both the GNPDF   
and CDF  
 for each case  
 :return: nothing  
 '''  
 #part 1 P(x<1|N(0,1))  
 mu\_a = #JES MISSING CODE #mean  
 sig\_a = #JES MISSING CODE #standard deviation  
 c\_a = #JES MISSING CODE

p\_a = #JES MISSING CODE #calculate the probability P(x<1|N(0,1))

#create the illustrative plots for part a  
 x\_a=np.linspace(mu\_a-5\*sig\_a,mu\_a+5\*sig\_a,500) #create a numpy array using   
linspace between mu-5\*sigma to mu+5\*sigma with 500 points  
 cdf\_a = np.array([#JES MISSING CODE FOR LIST COMPREHENSION]) #create a numpy   
array filled with values of CDF  
 gnpdf\_a = np.array([#JES MISSING CODE FOR LIST COMPREHENSION]) #create a numpy  
array for f(x) from the GNPDF

plt.subplots(2,1,sharex=True) #create two, stacked plots using subplots with   
sharex=True  
 plt.subplot(2, 1, 1) #set subplot 1 for our focus by using plt.subplot  
 plt.plot(x\_a, gnpdf\_a) #plot the gndpf\_a vs x\_a  
 plt.xlim(x\_a.min(),x\_a.max())  
 plt.ylim(0, gnpdf\_a.max()\*1.1)  
 # fill in area below GNPDF in range mu\_a-5\*sig\_a to 1  
 x\_fill = np.linspace(mu\_a - 5 \* sig\_a, c\_a, 100) #create a numpy array of x   
values from mu-5\*sigma to 1 with 100 points  
 gnpdf\_fill = np.array([#JES MISSING CODE FOR LIST COMPREHENSION]) #calculate   
the GNPDF function for each x in x\_fill and store in numpy array  
 ax=plt.gca() #get the axes for the current plot  
 ax.fill\_between(x\_fill, gnpdf\_fill, color='grey', alpha=0.3) #create the filled  
region between gnpdf and x axis

#construct the equation to display on GNPDF using TeX  
 text\_x=mu\_a-4\*sig\_a  
 text\_y=0.65\*gnpdf\_a.max()  
 plt.text(text\_x,text\_y,r'$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\  
left(\frac{x-\mu}{\sigma}\right)^2}$')  
 arrow\_x=(c\_a-mu\_a+5\*sig\_a)\*2/3+(mu\_a-5\*sig\_a) #calculate the x coordinate for   
where the arrow should point  
 arrow\_y=(#JES MISSING CODE ) #calculate the y coordinate for where the arrow   
should point  
 plt.annotate('P(x<{:0.2f}|N({:0.2f},{:0.2f})={:0.2f}'.format(c\_a, mu\_a, sig\_a,   
p\_a),   
size=8,xy=(arrow\_x,arrow\_y),xytext=(text\_x,0.5\*text\_y),arrowprops=dict(arrowstyle='  
->', connectionstyle="arc3")) #draw the arrow with text  
 plt.ylabel('f(x)', size=12)  
 ax.tick\_params(axis='both', which='both', direction='in', top=True, right=True,

labelsize=10) # format tick marks  
 # ax.xaxis.set\_ticklabels([]) #erase x tick labels for the top graph  
 ax.yaxis.set\_label('f(x)')

#create the CDF plot  
 plt.subplot(2,1,2) #select the second plot  
 plt.plot(x\_a,cdf\_a) #plot cdf\_a vs x\_a  
 plt.ylim(0,1)  
 plt.ylabel('$\Phi(x)=\int\_{-\infty}^{x}f(x)\mathrm{d}x$', size=12)  
 plt.xlabel('x')  
 plt.plot(c\_a,p\_a,'o', markerfacecolor='white', markeredgecolor='red')  
 ax=plt.gca()  
 ax.tick\_params(axis='both', which='both', direction='in', top=True, right=True,  
labelsize=10) # format tick marks  
 ax.set\_xlim(ax.get\_xlim())

ax.hlines(p\_a,ax.get\_xlim()[0],c\_a, color='black', linewidth=1)  
 ax.vlines(c\_a, 0, p\_a,color='black', linewidth=1)  
 plt.show()

#part 2 P(x>mu+2\*sigma|N(175,3))

# endregion

# region function calls  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()  
# endregion