

```
{
  "cells": [
    {
      "cell_type": "code",
      "execution_count": 9,
      "metadata": {},
      "outputs": [
        {
          "ename": "SyntaxError",
          "evalue": "invalid syntax (<ipython-input-9-47d2559f80ac>, line 13)",
          "output_type": "error",
          "traceback": [
            "\u001b[0;36m File \u001b[0;32m\"<ipython-input-9-47d2559f80ac>\"\n\u001b[0;36m, line \u001b[0;32m13\n\u001b[0m\n\u001b[0;31m{ \u001b[0m\n\u001b[0m\n\u001b[0m\n^ \u001b[0m\n\u001b[0m\n\u001b[0;31mSyntaxError\n\u001b[0m\n\u001b[0m invalid\nsyntax\n"
          ]
        }
      ],
      "source": [
        "{\n",
        "  \"cells\": [\n",
        "    {\n",
        "      \"cell_type\": \"markdown\",\n",
        "      \"metadata\": {},\n",
        "      \"source\": [\n",
        "        \"# Statistics 3: Peak Integration and Internal Standards\\n\",\n",
        "        \"\\n\",\n",
        "        \"This Python exercise will cover how to use Python to do peak integration and quantitation using internal standards. The concepts are introduced in the context of separations, but they could also be applied to other analytical techniques and data analysis problems. We will start by importing all the libraries we need. Don't forget to run this line of code with Ctrl+Enter or Shift+Enter\\n\",\n",
        "        ],\n",
        "      },\n",
        "    ]\n",
        "  }\n",
        "]\n",
        "{\n",
        "  \"cell_type\": \"code\",\n",
        "  \"execution_count\": 1,\n",
        "  \"metadata\": {},\n",
        "  \"outputs\": [],\n",
        "  \"source\": [\n",
        "    \"# Import Everything you Need\\n\",\n",
        "    \"import numpy as np\\n\",\n",
        "    \"import matplotlib.pyplot as plt\\n\",\n",
        "    \"import matplotlib.patches as patches\\n\",\n",
        "  ],\n",
        "},\n",
        "{\n",
        "  \"cell_type\": \"markdown\",\n",
        "  \"metadata\": {},\n",
        "  \"source\": [\n",
        "    \"What is the area under the peak? The simplest explanation is that it is sum of the area from each data point. Remember that area is length times height for a rectangle. For discrete data points, we can calculate the\"",

```

area between two points from the [trapezoidal rule] (https://en.wikipedia.org/wiki/Trapezoidal_rule). Here, the area of each data point y_k is equal to the average y value times the difference in x : $\frac{y_k + y_{k+1}}{2} \times \Delta x$. Let's look at an example with two data points (x_1, y_1) and (x_2, y_2) . The blue trapezoid shows the area between these two data points. We calculate this by calculating the area of the red rectangle, which is the average y value times the difference in x values. You can see that the area of the blue trapezoid is the same as the red rectangle. You can read more from the link above on the trapezoidal rule.

```

"    ]\n",
"  },\n",
"  {\n",
"    \"cell_type\": \"code\",\n",
"    \"execution_count\": 68,\n",
"    \"metadata\": {},\n",
"    \"outputs\": [\n",
"      {\n",
"        \"name\": \"stdout\",\n",
"        \"output_type\": \"stream\",\n",
"        \"text\": [\n",
"          \"The integral is: 13.0\\n\\n\\n\",\n",
"        ]\n",
"      },\n",
"    ]\n",
"  },\n",
"  {\n",
"    \"data\": {\n",
"      \"image/png\":

```

```

\"iVBORw0KGgoAAAANSUgAAAXwAAAEKCAyAAARnO4WAAABHNCSVQICAgIfAhkiAAAAAlwSFlz
AAALEgAACxIB0t1+/AAAGBhJREFUeJzt3XuQVdY3H3l+UOgkWW6IjYVnIk2bpYgu64rtWmiUkc
WXGiyUw6I5ekaZSGTJs6TqbjThymduOOZjLTjMfJtEnK2GqcGMv2OFbHdm1HqhKPazuWy/OiQELch
EBiJWC5CcGy3/5xDuWk7S67wHme3ee8XzM7+zznnH1+Xw4/Pvz2XH4nMhNJUuub1OwCJEmNYeBLUk
0Y+JJUEwa+JNWEgS9JNWHgS1JNGPiSVBMGviTVhIEvSTUxudkFDLZgwYLS6upqdhmSNGGsW7futcZ
sGM224yrwu7q6WLT2bbPLkKQJIyL2jXZbD+liUk0Y+JJUEwa+JNWEgS9JNWHgS1JNGPiSVBMGviTV
RKWBHG/FxHbImJrRDwREdOrbE+SNLzKAj8iFgG/A3Rn5j1AG/BIVelJ0kSzZg10dcGkScX3NWuqb
a/qO20nAzMi4hwwEzhYcXuSNCgSWQMrV8Lp08X7ffuK9wArVlTTZmUj/Mx8GfhjYD9wCOjNzKeqak
+SJPjVqwbC/oLTp4v1VanykM484GHgVuBmYFZE/MoQ262MiLURsbanp6eqciSp6fr74dAh+MEPiH
9UPbvr679Kg/p/AywJzn7ACLiA8D7gC8M3igzVwOrAbq7u7PCeiSpotLh1Vdh927YvBm2bBkY1c+e
DSdPvvlnoJurq6fKwN8PvDciZgJvAMsBp8KU1LIy4fBh2LSXNm0qQv7UqWLDjBnQ3g4d5UTGH/gAf
OMbcO7cwM/PnAmPPVZdfZUFfmY+GxFfBdYdfcAGypG8JLWCTOjPqGj4LVuKUXsmTJ9eBPYCBUP/7L
33Ft+feqr4mSVLirCv6oQtVHyVTmZ+EvhklW1IUqNkwmuVfQG/eXPxdeLEQMDPnQvXXz/6z7v3Xrj
hBujuhkcfrazzi8bVA1AkaTzJhNdfLwJ+61bYuBGOHy+WT5tWjODnzWt21aNN4EtSKROOHBkI+E2b
4NixYvnuUW4t7c3u8orZ+BLqq1MOHq0CPht24oR/JEjEAFTphThvnhx8b4VGPiSauVCwD/3HGzYU
ByyiYDjk4uA7+xsnyC/liEvqAX19sKePbB9exHwPT31CfhLGfiSWsrX4wMBv3FjceNTBLS11S/gL2
XgS5rQTpwoDtFcGMG/8ooBPxwDX9KEcvJkEfa7dhQBf7Ccg9eAvzWDX9K4durUWwP+wIFI/vhJkwz
4sTLwJY0rp04VM0nu3Anr18PLLxfLJ00q7mRdssSAv1IGvqSmOn26CPjnnny8Cfv/+ItAjWu86+GYz
8CU11BtvFAH/wguwbt1b53/3EE21DHxJlTpZiDgN2woLpmEItTnzIFbbikO16h6Br6ka+rNN98a8
Lt3D6yb09eAbyYDX9JvEPN4rDMrl3FMfjdu4s5asAR/Hhj4Esak7NnBwJ+w4bi++CAX7TIgB+vDH
xJIzP3D156aSDgX3iHeBg3wHXXwc03Fzc9afyrLPAj4k7gy4MW3QZ8IjM/U1Wbkq7euXPFzU27dhV
z0ezc+daAv+kmA36iqvKZtjuB+wEiog14GXiyqvYkXZm+viLgX3yxGMHv3AnnzxfzS824FtJow7p
LAdezMx9DWp0jD6+oq7V198sRjB79hRLMssRvA33FBMHazW06i/1keAJxrUlqRBzp8vAn737mIEv
337QMDPng0LFxrdVH5X3NETAV+Efj4M0tXAisBOjs7qy5HannnzxczS07eXYzgn3uuOC5vwKsRf+
0fBNZn5qtDrczMlcBqg07u7mxAPVJL6e8vAn7PniLgt20rLp3MhFmzYMGc4vmsUiMC/8N4OE6Zvr
74dChIuA3bYKtW4ubnzJh5kyYP9+A19AqDfyImAn8LPBbVbYjtbL+/uIpToMD/syZgYC//noDXqNT
aeBn5mlgfpVtSK0mcyDgN2+GLVuKGSyZyCaMYkbJqVOBXaUmIk/dSE2WCYcPD4zgt2wp5ogfHPAdH
c2uUq3AwJcaLBN6egZG8Js3F095yoTp04uAX7Cg2VWqFRn4UsUy4bXXiueyXgj448eLdRcCfr4HPT
UABr50jWXC668XAb91S3Gp5IWAnzatCPh585paomrKwJeuUiYcOTIQ8Js2wbFjxToDXuOJgS+NZM0
aWLWqmAC+sxMee4z81ys4erQI+G3biukKjh4tNp861eeyavwy8KXhrFkDK1cW18wA7NtH36Mr+fIX

```

4f8sXAEY8JpYfC6NNIRjx+DsxlYNhH1p8tnTPPT9VXR2wpIlxdTBM2YY9poYWmeEf+ON8OqQ0/Vl1
9XLHPbSxXbexQbu5zALeZyhZ/NuP76P+NR/bmyBamELyC7z8Gj1z4ZqncA37DUGx7muDPh3spGlvM
KNBP200U87R+1kP73MpZ3et/1sL3ObULFa2vG397MqtE7gSyM4wWz20sUO7mQDSznEzQT9TKKfdo7
RyT4uPSrzDMv5Bb7BVM5dXHaWKTzD8sYWL10jBr5a0klmsZcudnIn61nKQW4myDLge4cM+Ett4V4A
lvMMc+mll17k8w/KLy6WJxsBXSzjFzDLg72ADyzjAaosBP5deOnmJYOyPW9jCvQa8WoaBrwnpNDPYx
xJ2cgfrWcYBFgNJkOUI/soCXmplBr4mhDeYzj6W8Dx3sJ617GMJAUQ5gl9swEuXZeBrXDrDNPaxhB
e4nfUsYy9dwIWAP85iXmKSAS+NiYGvceEM09hPJy/we6xnGXu4DYAgmUMvtxjw0lWr+hGH7cBngXu
ABB7NzH+osk1NDG8ylf10sot3sJ5/xG5uI8vrZuYa8FI1qh7h/wnwncz85YiYCsysuD2NU2eZcjHg
N7CUXfwEYSQgmcNxFvEyK+hvdplSS6ss8CNIvCTwL8GyMyzwNmQ2tP4cpYpvMRiXuQdrGcpu7id/
nLMPofj3MxB2gx4qaGqHOHfBvQA/zMi7gPWAR/NzFODN4qIlcBKgM7OzgrLUZXOMfliwG/gfp7nDv
qZBATXcYKbDHip6aoM/MnAMuAjmflsRPwJ8PvAfxq8UWauBLYDdHd3e9B2guijrQz429jAUp7nTs4
zibwY8K/QxvmlmlylpkCoD/wBwIDOfLd9/1SLwNQh10cbLLGIX72Aj97ODd3KetjLgT3KjAS+Ne5UF
fma+EhEvRcSdmbkTWA48V1V7urbOM4mXWRxBL+Dd9FHGXDM4iQ38CqTDXhpQqn6Kp2PAGvKK3R2A
79ecXu6QueZxEFu5kVuYxP3s4276GMKCczmFAs5zGT6ml2mpKtQaeBn5kagu8o2dGXOM4lD3MRubm
Uj9/Mcd3OWqSQwilMspMeAl1qMd9rWRD/BIW5iD11sZClbuZuzTCsD/jTzeY0pBrzU0gz8FtVP8Ao
3socuNnEfW7mHM8ygn2AWp5jPEayMerCHpNZn4LeIfOjXuYE9dLGZ+9jCvzbBDJJGJqdp5xhT6Wl2
mZKayMCfoBIGbfy72cK7OcVMIJjBadrppcOAlzSIgT9BJHCYhewtA34T7+Y0M0mC6ZyhnWMS4LVml
ylpHDPwx6kEeugoA/5eNnMfJ5lNAtN5swz415tdpqQJxMAfJxJ4nfns4Va2cA+buI8TzCGBaWAXX8
+RZpcpaQIz8JskgSNczx5uZSv3sJH76GUuANM4SzvHmMex5hYpqaUY+A2SwFHmsYdb2cbdbOR+jjI
PSKaWAb+E/c0uU1ILM/ArksAx2t1LF1vLgD/CfIjKcudo5xid7Cuf8SRJ1TPwr6FjzGUvXWzjLjaw
lNdZQJBMNuAljQMG/1XoZQ576WI772I9yzhMRzmC7zPgJY07Bv4YHOe6MuDfyQaW8Qo3lCP4vvIYv
AEvafwy8EdwgtntspYsd3Ml6lvEKNwHJZM4b8JImHAN/kJPMYi9d7ORO1rOUgywCkjb6aeoeh2gkTW
i1DvxtZLwY8BtYygEWEcAkztNORwEvqaVUGvgRsRc4AZwH+jKzqQ9D0c2MMuDVkAN+MQBBf3mS9SU
Cn6MuqTU1YoT/05lZ2axea9bAqlWwn/N0sp/H+ANW8AQAbzCdvXTxPLezgWXso5OgCPi59LLYgJdU
IxP6kM6aNbByJZw+DTCJfXTxb/gc61nGHI6zjy4oA704RGPAS6qvSRV/fgJPRcS6iFh5rT981aoLY
T/gDDP4H/wWx2jnFl6is/yaw3HDXlKtVT3CfyAzD0bEQUdPiNiRmd8bvEH5H8FKgM7OzjF9+P5hpp
45xWzmcvYKCPakVlXpCD8zD5bfDwNPAu8ZYpVmdmdmd0dHRl1j+vzh/n+YS++Ya5WkVldZ4EfErIi
47sJr4APAlmvZxmOPwcyZb102hbMs55lr2YwktYQqD+ncADwZERfa+WJmfudaNrBiRfH9Ix+Bo0eT
ufSynGe4ly3XshlJagmVBX5m7gbuq+rzLl1xao4cged/578x30f+SdKwqr5KR5I0Thj4klQTB4kl
YSBL0k1YeBLUk0Y+JJUEwa+JNWEgS9JNWHgS1JNGPiSVBMGviTVxLCBHxHfioiuxpUiSarSSCP8v6
R4WtWqiJjSoHokSRUZdrbMzPxKRPxv4BPA2oj4a6B/0PpPN6A+SdI1crnpkc8Bp4BpwHUMCnxJ0sQ
ybOBHxIPAp4GvA8sy8/Rw20qSxr+RRvirgH+RmdsaVYwkqTojHcP/Z9eigYhoA9YCL2fmh67FZ0qS
xq4R1+F/FNjegHYkSSOoNPAj4hbg54HPVtmOJOnyqh7hfwB4j4xwdU9ErIyItRGxtqenp+JyJKm+K
gv8iPgQcDgz1420XWauzszuzOzu6OioqhXjqr0qR/gPAL8YEXuBLwHvj4gvVNIEJGkElQV+Zn48M2
/JzC7gEeDvMvNXqmpPkjQyZ8uUpJq43NQK10Rmfhf4biPakiQNzRG+JNWEgS9JNWHgS1JNGPiSVBM
GviTVhIEvSTVh4EtSTRj4klQTB4klYSBL0k1YeBLUk0Y+JJUEwa+JNWEgS9JNWHgS1JNGPiSVBNV
PsR8ekt8OCI2RcS2iPjDqtqSJFlelU+8ehN4f2aejIgpwPcj4tuZ+aMK25QkDaOywm/MBE6Wb6eUX
11Ve5KkkVV6DD8i2iJiI3AYeDozn62yPUnS8CoN/Mw8n5n3A7ca74mIey7dJiJWRsTaiFjb09NTZT
mSVGsNuUonM48B3wUeHGLd6szszszujo6ORpQjSbVU5VU6HRHRXr6eAfwMsKOq9iRJI6vyKp2bgM9
HRBvFfyxfycxvTtieJGkEVV6lsxlYwtXnS5LGxjttJakmDHxJqgkDX5JqwsCXpJow8CWpJgx8SaoJ
A1+SasLAl6SaMPAlqSYMfEmqCQNfkmrCwJekmjDwJakmDHxJqgkDX5JqwsCXpJow8CWpJqp8pu3ii
Pj7iNgeEdsi4qNVtSVJurwqn2nbB/yHzFwfEdcB6yLi6cx8rsI2JUnDqGyEn5mHMnN9+foEsB1YVF
V7kqSRNeQYfkr0UTzQ/NlGtCdJervKAz8iZgN/A/xuZh4fYv3KiFgbEWt7enqqLkeSaqvSwI+IKRR
hvyYzvzbUNpm5Oj07M7O7o6OjynIkqdaqvEongM8B2zPz01WlI0kanSpH+A8Avwq8PyI2l18PVdie
JGkElV2WmZnfB6Kqz5ckjY132kpSTRj4klQTB4klYSBL0k1YeBLUk0Y+JJUEwa+JNWEgS9JNWHgS
1JNGPiSVBMGviTVhIEvSTVh4EtSTRj4klQTB4klYSBL0k1YeBLUk1U+UzbxyPicERsraoNSdLoVT
nC/0vgwQo/X5I0BpUfFmZ+DzhSledLksam6cfwI2JlRKYNiLU9PT3NLkeSWlbtAz8zV2dmd2Z2d3R
ONLscSWpZTQ98SVJjGPiSVBNVXpb5BPAPwJ0RcSAifqOqtirJlze5qg/Oza9X9dmSpLHzkI4klYSB
L0k1YeBLUk0Y+JJUEwa+JNWEgS9JNWHgS1JNGPiSVBMGviTVhIEvSTVh4EtSTRj4klQTB4klYSBL
0k1YeBLUk0Y+JJUE5UGfkQ8GBE7I2JXRPx+lw1JkkZW5SMO24D/DnwQuAv4cETcVVV7kqSRVTnCfw
+wKzN3Z+ZZ4EvAwXW2J0kaQWXPtAUWAS8Nen8A+MdVNDRtGhyLndPMqOLjJakyZ5jOFPOa01aVgR9
DLMu3bRSxElhZvj0ZETuvsL0FwGtX+LNVsq6xxsa6xxsa6xGZdlfREW/Kv4wpXWtWS0G1YZ+AeAXYPE
3wIcvHSjzFwNrL7axiJibWZ2X+3nXGvWNTbWNTbWNTZ1r6vKY/j/D7g9Im6NiKNAI8DXK2xPkjSCy
kb4mdkXEF8e+FugDXg8M7dV1Z4kaWRVHtIhM78FfKvKNga56sNCFbGusbGusbGusal1XZH5tvOokq
QW5NQKklQT4z7wi+LxiDgcEVuHWR8R8af19A2bI2LZoHW/FhEvlf+/1uc6VpT1bI6IH0bEfYPW7Y2
ILRGxMSLWNriun4qI3rLtjRHxiUHRKpsKYxR1fWxQTVsj4nxEXF+uq3J/LY6Iv4+I7RGxLSI+OsQ2
De9jo6yr4X1slHU1vi+Nsq6G97GImB4RP46ITWVdfzjEntMi4svlPnk2IroGrft4uXxnRPzcVReUm

eP6C/hJYBmwdZj1DwHfprju/73As+Xy64Hd5fd55et5Dazrfrfao5he4tlB6/YCC5q0v34K+OYQy9
uAF4HbgKnAJuCuRtVlyba/APxdg/bXTcCy8vVlwPOX/rmb0cdGWVfD+9go62p4HxtNXc3oY2WfmV2
+ngI8C7z3km3+HfAX5etHgC+Xr+8q99E04NZy37VdTt3jfoSfmd8DjoywycPAX2XhR0B7RNwE/Bzw
dGYeycyjwNPAG42qKzN/WLYL8COK+xAqN4r9NZxKp8IYY10fBp64Vm2PJDMpZeb68vUJYDvFXeKDN
byPjaauZvSxUe6v4VTWx66grob0sblPnCzfTim/Ljl1x+jDw+fLlV4HlERHl8i9l5puZuQfYRbEPr9
i4D/xRGGoKh0UjLG+G36AYIV6QwFMRsS6K040b7Z+Uv2J+OyLuLpeNi/0VETMpQvNvBi1uyP4qf5V
eSjEKG6ypfWyEugZreB+7TF1N62OX21+N7mMR0RYRG4HDFAOEYftXZvYBvcB8KthflV6W2SDDTeEw
qqkdqhYRP03xj/GfDlr8QGyejIiFwNMRSaMcATfCemBJZp6MiIeA/wXczjjZXxS/av8gMwf/NlD5/
oqI2RQB8LuZefzS1UP8SEP62GXqurBNw/vYZepqWh8bzf6iwX0sM88D90dEO/BkRNYtmYPPZTWsf7
XCCH+4KRxGNbVDlSLi3cBngYcz8/ULyzPzYPn9MPAkV/lr2lhk5vELv2JmcZ/ElIhYwDjYX6VHuOR
X7ar3V0RMoQiJNZn5tSE2aUofG0VdTelj16urWX1sNPur1PA+Vn72MeC7vP2w38X9EhGTgbkUh+zv
/f66licoqvoCuhj+JOTP89YTaj8ul18P7KE4mTavfHl9A+vqpDjm9r5Lls8Crhv0+ofAgw2s60YG7
r94D7C/3HeTKU463srACbW7G1VXuf5CR5/VqP1V/tn/CvjMCNs0vI+Nsq6G97FR1tXwPjaauprRx4
AOoL18PQP4v8CHLtnmt3nrSduvlK/v5q0nbXdzlSdtX/0hnYh4guKs/4KIOAB8kuLEB5n5FxR38j5
E0fFPA79erjsSEf+FYk4fgE/lW3+Fq7quTlAch/uz4vwLfvLMjnQDxa9lUPwD+GJmfqeBdf0y8G8j
og94A3gki95V6VQYo6gL4JeApzLz1KAfrXR/AQ8AvwpsKY+zAvwBRZg2s4+Npq5m9LHR1NWMPjaa
qDxfewm4PNRPBBqEkWYfzMiPgWszcyvA58D/joidlH8Z/RIWfO2iPgK8BzQB/x2FoeHrph32kpSTb
TCMXxJ0igY+JJUEwa+JNWEgS9JNWHgS1JNGPjSMMoZGpCmmlFvXv1+1A+NlsYTA18aRma+BPw58Ef
loj8CVmfmvuZVJV05r8OXr1Derr8Oebz4TWBpFjM9ShPOuL/TVmqmzDwXER8DvgN8wLDXROYhHeny
PggcAu5pdiHS1TDwpRFExP3Az1JmmvZ75YNPpAnJwJeGUT516M8p5lbfD/xX4I+bW5V05Qx8aXi/C
ezPzKfL938GvDMi/nkTa5KumFfpSFJNOMKXpJow8CWpJgx8SaoJA1+SasLAl6SaMPAlqSYMfEmqCQ
Nfkmri/wO9zx0TfZmNlwAAAABJRu5ErkJggg==\n\n",\n",

```
"    \"text/plain\": [\n",\n"    \"<matplotlib.figure.Figure at 0x7f2b41557fd0>\"\n",\n"    ]\n",\n"    },\n",\n"    \"metadata\": {},\n",\n"    \"output_type\": \"display_data\"\n",\n"  },\n",\n" ],\n",\n" \"source\": [\n",\n"   \"# Define the data points\n",\n"   \"x1 = 1\n",\n"   \"x2 = 3\n",\n"   \"y1 = 5\n",\n"   \"y2 = 8\n",\n"   \"\n",\n"   \"# Calculate the average y and change in x\n",\n"   \"average_y = (y1 + y2)/2\n",\n"   \"delta_x = (x2-x1)\n",\n"   \"\n",\n"   \"# Calculate the integral\n",\n"   \"integral = average_y * delta_x\n",\n"   \"print('The integral is:', integral)\n",\n"   \"\n",\n"   \"# Plot the results\n",\n"   \"plt.plot(x1, y1, marker='o', color='blue')\n",\n"   \"plt.plot(x2, y2, marker='o', color='blue')\n",\n"   \"plt.plot(2, average_y, marker='o',\n",\n"color='red')\n",\n"   \"plt.gca().add_patch(patches.Rectangle((x1, 0), delta_x, average_y,\n",\n"color='red'))\n",\n"   \"x = [x1, x2, x2, x1]\n",\n"   \"y = [0, 0, y2, y1]\n",\n"   \"plt.gca().add_patch(patches.Polygon(xy=list(zip(x,y)), color =\n",\n"   \"blue\", alpha = 0.5))\n",\n"   \"plt.ylim(0,8.5)\n",\n"
```

```

"      \"plt.xlabel(\\\\"X\\")\\n\\",\\n",
"      \"plt.ylabel(\\\\"Y\\")\\n\\",\\n",
"      \"plt.show()\\n\\n",
"    ]\\n",
"  },\\n",
"  {\\n",
"    \"cell_type\\\": \"markdown\\\",\\n",
"    \"metadata\\\": {\\n",
"      \"slideshow\\\": {\\n",
"        \"slide_type\\\": \"fragment\\\"\\n",
"      }\\n",
"    },\\n",
"    \"source\\\": [\\n",
"      \"For an array, we just need to sum all of these little trapezoids
together. First, let's generate a Gaussian peak:\\n\\n\",
"    ]\\n",
"  },\\n",
"  {\\n",
"    \"cell_type\\\": \"code\\\",\\n",
"    \"execution_count\\\": 5,\\n",
"    \"metadata\\\": {},\\n",
"    \"outputs\\\": [\\n",
"      {\\n",
"        \"data\\\": {\\n",
"          \"image/png\\\":
\\\"iVBORw0KGgoAAAANSUHEUgAAAX0AAAD8CAYAAACb4nSYAAAABHNCSVQICAgIfAhkiAAAAAlwSF1
zAAALEgAACxIB0t1+/AAAIABJREFUeJzt3Xl0XHD99/H3d0abZWu1ZMvWYsmObEdJvMo2SchCFuJA
sIEG4rT0CZDWhIcQOLRPSQoNfdL2HJYS2kMDxQ+EUiAYk1BwgolJyEJC4kWyvMmObXnR4lW2dmvXf
J8/ZsZMFm6kka6s3xf58ieuf07d77XHn109bv3/n6iqhhjjIkPHrcLMMYYM3ks9I0xJo5Y6BtjTB
yx0DfGmDhioW+MMXHEQt8YY+KIhb4xxsQRC31jjIkjFvrGGBNHETwuYLCcnBwtLi52uwxjjIkqlZW
V51Uld6R2ERf6xcXFVFRUuF2GMcZEFRGpddLOuneMMSaOWOgbY0wcsdA3xpg4YqFvjDFxxFHoi8hq
ETkkIjUi8vBl2t0tIioi5SHLHgmsd0hE7ghH0cYYY8ZmxKt3RMQLPAHcDjQA00Vks6oeGNQuDXgI2
B6yrAxYB1wFzAZeFJH5qjoQv10wxhj1JmJ/ZVAjaoeU9VeYCOwdoh2/wR8HegOWbYW2KiQPap6HK
gJbM8YY4wLnIR+PlAf8rwhsOwSEVkkFKRqc6Nd15ho0dvv4/n9Z9i4o47Wzj63yzFmTJzcnCVDLLs
0sa6IeIBvAr8f7boh21gPrAcoKipyUJIxk6e7b4BvvXiYpysauHCxF4BHN1ez+qo8/m71AgqyUl2u
0BjnnIR+AlAY8rwaOBXyPA24GnhFRADygM0issbBugCo6gZgA0B5ebnN1G4ihs+n/M0v9vCbvadZf
VUe96woJGdaMr+orOeZygb2n2zlmU9fR9bUJLdLNCyRJ6G/EygVkrLgJP4Ts38effVFW4Gc4HMReQ
X4W1WtEJEu4CkReRz/idxSYEf4yjdYnlt61v8Zu9pHrlzIZ+6ad6l5dcUZPCBxbP5i+9vZ/2PK/j
x/atISfS6WKkxzozYp6+q/cCDwFbgILBJVatF5LHA0fz11q0GNgEHgOeBz9iVOyZaPLW9ju+9eoy/
fNcc1t849x2vryjO5vGPLmbniWb+z9N7UbVfUk3kk0j7oJaXl6sNuGbcdqGjh5u+8QpLizL54cdXk
OAd/vjo278/wjdfOMyPPrmSm+aPOMihMRNCRCPvtXykdNZHrjFDeOLlo3T29vOVD5RdNvABPNXTPA
qzp/C1376FzxdZB1HGDGahb8wgDc2d/GRbLR9ZXsgVM9JGbj+U40Fvbl/AgdNtPLv3HdcpGBNRLPS
NGeTxFw6DwOdvL3W8zprFs1mYl8Y3f3eY3n7fBFZnzPhY6BsT4sjZdv6n6iQfv66YWRlTHK/n8Qhf
XL2QuqZONlXUj7yCMS6x0DcmxE+21ZLo8fBAYOWZTt28IJdFBRn895sn7EoeE7Es9I0J604b4H+qT
rL66jyyx3Cz1Yhw78oiDp/toKq+ZQIQNgB8LPSNCFjt/to0dfezbmXhyI2H8YHF501N8rJxR10YKz
MmfCz0jQn42Y56iqencu3c6WPexrTkBNYsns2ze07T3m2DspnIY6FvDHC0sYMdx5v46IpCamNIjdk
9Kwrp6htg8x67fNNEHgt9Y4BNO+tJ8Ah3Ly8Y97aWFGayMC+Nn++0q3hM5LHQ3HP51N+WXSWxbO
YEZayri3JyLcs6KQvQ2tHDnbHoYKjQkfc30T96rqm21s7+H9i2aFbZvvu8a/ra3VZ8K2TWPCwULfx
L3n958hyevhLoUzwrBnmeKpLCvK5HkLFRNhLPRNXFNvNq8+w3VXTCctJTGs277jqjz2n2yjjvqkzrN
s1Zjws9E1cO3i6nfqMLlZflRf2bd8R2ObvDpwn+7aNGSsLFRPXnq8+g0fgtrKZyD92cc5UFualsXW
/dfGYyOEo9EVktYgcEpEaEXl4iNcfEJF9IrJbRF4XkbLA8mIR6Qos3y0i/xnuHTBmPH5XfYby4mxy
piVPyPbvucQpnbVNNLb3TMj2jRmtEUNfRLzAE8CdQB1wbzDUQzylqteo6hLg68DjIa8dVdUlgA8Hw
lW4MeN14vxF3jrTPiFdO0Grr85DFV48aF08JjI4OdJfCdSo6jFV7QU2AmtDG6hqW8jTqYANMWgiXj
CI33tV+Lt2ghbmvVGUncoLlq9vIoST0M8HqM8tbAgsexsR+YyIHMV/pP9QyEs1l1l1lIq+KyAlDvYG
IrBerChGpaGxsHEX5xozdq4cbKZ0xjYKs1Al7DxHh5gW5vHn0Aj39AxP2PsY45ST0hxqI5B1H8qr6
hKROA74IfDmw+DRQpKpLgS8AT4lI+hDrblDVclUtZ821iaXNxOvuG2DH8SZuKJ34z9sNpbl09Q2wq
9aGWzbucxL6DUDoWLMFwOVGktoIfBBAVxtU9ULgcSVwFJg/tlKNCZ+dJ5ro6fdxw/ycCX+vd83NJs

```

EjvHbEfosl7nMS+juBUhEpEZEKBYB2wObSBiIROJvp+4EhgeW7gRDAiMhcoBY6Fo3Bjxu01I+dJ8np
YVZI94e+VlpLIsqIsXjtyfsLfy5iRjBj6qtoPPAhSBQ4Cm1S1WkQeE5ElgWYPiki1iOzG341zX2D5
jcBeEdkDPA08oKpNYd8LY0bpD4cbKS/OIjUpYVLe74bSHPafauVCh126adzl6BOvqluALYOWPRry+
HPDrPcM8Mx4CjQm3M61dfPWmXa+uHrhpL3nDfNz+eYLh/nj0QusWTx70t7XmMHsjlwTdl6v8Xez3F
A68f35QdfkZ5CZmshrh61f37jLQt/EndeOnCdnWhJls95xIdmE8XqE66/I4bUj51G121iMeyz0TVx
RVV47cp7rr8jB4xnftIijdWNpDmfaujlyrmNS39eYUBb6Jq7UnOvgfEcP18+bvK6doOsC77nt2IVJ
f29jgiz0TVzZdtx/8diqurN/qeZgBV1TmJ2RwvZjdgGbcY+Fvokr249dIC89haLsiRt6YTgiwqq50
91+/IL16xvXWOibuKGqbD/exKq52YhMbn9+0KqSbM539HK08aIr72+Mhb6JG8fPX6SxvYdVJdNdq2
HVXP97bz9u/frGHRb6Jm5sd7E/P6h4eioz0pKtX9+4xkLfxI3txy6QMy2ZuTlTXavB+vWN2yz0TVy
IhP78oFUL2Zxt66H2QqerdZj4ZKFv4kJ9UxenW7t51ySMqjmSdwW616xf37jBQt/EhW2BgA2eSHXT
vNxp5ExLsn594woLFRMXdh5vIislkdIZ09wuBRFhRXE2005Y6JvJZ6Fv4kJlXTPL52S53p8ftHxOF
g3NXZxt63a7FBNnLPRNzGu62Muxxossn+N+f35QebG/lSraZpcrMfHGUEiLyGoROSQiNSLy8BCvPy
Ai+0Rkt4i8LiJlIa89EljvkIjcEc7ijXEiGKzL52S5XMmflM1KJznBY6FvJt2IoR+Y4/YJ4E6gDLg
3NNQDnlLVa1R1CfB14PHAumX459S9ClgNfCc4Z64xk6WytplEr7CoIMPTUi5JSvCwuDCTCgt9M8mc
HOMvBGpU9Ziq9gIbgbWhDVS1LeTpVCB418laYKQq9qjqcaAmsD1jJk11bRNx52eQkhhZxxv1c7KoP
tlKd9+A26WYOOIk9POB+pDnDYflbyMinxGRo/iP9B8a5brrRaRCRCoaG206ORM+Pf0D7GloZX1R5H
TtBC2fk0W/T91T3+J2KSAOOAn9oS53eMf946r6hKrOA74IfHmU625Q1XJVLc/NzXVQkjHOVJ9qo7f
fR31x5IX+ssAPIuviMZPJSeg3AIUhzwuAU5dpvxH44BjXNSasKk/4A3VZBJ3EDcqamsS83KnsstA3
k8hJ608ESkWKRESS8J+Y3RzaQERKQ56+HzgSeLwZWCCiYJSASpQCO8ZftjHOVNY2U5Sdyoy0FLdLG
VL5nGwq65rx+WzwNTM5Rgx9Ve0HHgS2AgeBTapaLSKPiciaQLMHRaRaRHYDXwDuC6xbDwWCDgDPA5
9RVTtrZSaFqlJR2xxrL2oOtnxOFi2dfRw7b5OqmMmR4KSRqm4Btgxa9mjI489dZt1/Af51rAUaM1b
1TV2c7+iJyK6doOWBcw27apu5IgKGiDCxz+7INTGrqj7Qn1+U6XIlwyuZPpWMKYmXajVmolnom5hV
VdfClEQvC2amuV3KsDweYUlhJlV1dtmmRwW+iZmVdU1s6gggRvZH/MlxZlcuhsOx09/W6XYuJAZ
H83GDNG3X0DVJ9qY2ke3pQ12NKiLFRhr92kZSaBhb6JSdWnWun3KUsjuD8/aEmBv8YqC30zCSz0TU
wK9pEvLYz80M9ITWRe7lSg6uxkrpl4FvomJlXvt5CfOYUZ6ZF5U9ZgS4uyqKprQdVu0jITy0LfxKT
ddS1R0bUTtLQokwsXe6lv6nK7FBPjLPRNzDnbls3Jlq6oOIkbLTQX6tdr28mmoW+iTmX+vOj6Eh/
/sxppCZ57Xp9M+Es9E3MqapvJsnr4arZ6W6X41iC18Oigx22clcm8Es9E3Mqapr4crZ6SqnRNZMW
SNZUpjFwdNtNpOWmVAW+iam9A/42NfQGhWXag62pDCTvgGl+1TbyI2NGSMLfRNTjzproKtvgMWFkT
MJulNLAj+obPpEM5Es9E1M2R0IzCWF0XPlTlBeRgp56SmX9sGYieAo9EVktYgcEpEaEXl4iNe/ICI
HRGSviPxeROaEvDYgIrsDX5sHr2tMOO2uayEzNZHi6alulzImSwozLftNhBox9EXECzwB3AmUafeK
SNmgZlVAuaouAp4Gvh7yWpeqLgl8rcGYCbS7voXFBZmIiNuljMmSokzqmjq50NHjdikmRjk5018J1
KjqMVXtxT/x+drQBqr6sqp2Bp5uwz8BujGTqqOnn8Pn2i/1jUejYO17G1pdrstEKiehnw/UhzxvCC
wbzv3Ab0Oep4hIhYhsE5EPjqFGYxzZ19CKqv9oOVpdk5+BR2zETTNxnMyRO9TvyUOOCiUiHwPKgZt
CFhep6ikRmQu8JCL7VPXooPXWA+sBioqKHBVuzGDBvVDFBdB+lOTE5g/M8369c2EcXKk3wAUhjwv
AE4NbiQitwFfAtao6qUOSVU9Ffj7GPAKsHTwuqq6QVXLVbU8Nzd3VDtgTNDu+mbmTE8le2qS26WMy
5LCTPbU24ibZmI4Cf2dQKmIlIhIErAOeNtVOCKyFPge/sA/F7I8S0SSA49zgOuBA+Eq3phQe+pbo7
o/P2hJYSatXX2cuNA5cmNjRmnE0FfvfuBBYctwENikqtUi8piIBK/G+QYwDfjFoEszrwQqRGQP8DL
wVvW10Ddh6alnz3bER+oFzErttxE0zAZz06aOqW4Atg5Y9GvL4tmHWewO4ZjwFGuNEMCAx0Do
185IizXJy+66Fj601C6EM+Fld+SamFBV30KiVyibFT0jaw7H6xGuyc+wk7lmQljom5iwp76Fslnpp
CRG18iaw11S1MmB02309NuImya8LPRN1BvWkfsaYuMkbtDSwIibB2zETRNmFvom6h05187F3oGovi
lrsOCacdbFY8LNQt9Evd110Tuy5nBsxE0zUSz0TdTbXd9CxpToHVLzODbippkIFvom6u2ub2FxyfS
OrDmcxYWZ1F7opO1ir9ulmBhioW+i2sWefg6fje6RNYdzaSatBjvaN+FjoW+i2r6TrfiUqJwTdySL
CvwbgbgPWRgTDhb6JqpdGlkzBkPfRtW0E8FC30S13XUtMTGy5nCWFGayp8FG3DThY6FvolpwesRYt
aQwk5ZOG3HThI+Fvolaplu7ONPWzbIYuilsKVF/nsPqupsxE0THhb6JmpVBU5wBoMxFl0xYxrTkh
Mu7asx42Whb6JWVv0zSQkeroyBkTWH4/UIiwszqLKx9U2YWOibqLW7voWrZ6eTlBDbH+MlhZm8dbq
drl4bcdOMn6PvFhFZLSKHKRGRB4e4vUviMgBEDkIr8XkTkh90nIkcCX/eFs3gTv/oGfOxtaI3p
rp2gpYVZ9PuU/ada3S7FxiARQ19EvMATwJ1AGXCviJQNaLYfLKVqIuBp4OuBdbOBwCrgJXAV0Qk9
r9LzYR763Q7Pf0+lsbwSdyg4OihdjLXhIOTI/2VQI2qH1PVXmAjsDa0gaq+rKrBa8q2AcE53u4AX1
DVJlVtB14AVoendBPPgn3c8XCknzMtmaLSVDuZa8LCSejNa/UhzxsCy4ZzP/DbMa5rjCNVdS3MSEt
mdkaK26VMiqVFmRb6JiyhP5QQxcOeXugiHwMKAE+MZp1RWS9iFSISEVjY60Dkky8q6prZkkMjqw5
nCWfMzxp6+Z0a5fbpZgo5yT0G4DCkOcfWKnBjUTkNuBLwBpV7RnNuqq6QVXLVbU8NzfXae0mTjVd7
OXEHc646NoJ+tNNWna0b8bHSEjvBEpFpEREkoBlwObQBikYFPge/sA/F/LSVuC9IpIVOIH73sAYY8
Zsd6A/PxaHUx502Sz/pam7aulkrhmfhJEaqGq/iDyIP6y9wJOqWi0ijwEVqroZf3fONOAXGv+361R
1jao2icg/4f/BAfCYqjZNYJ6YuLgrtuXSTUvXiInBwzX5GVTZiJtmnEYmfQBV3QJsGbTs0ZDht11m
3SeBJ8daoDGDVdY2UzYrndQkRx/fmLF8Thb/9ccT9PQPkJzgdbscE6Vi+1ZGE3P6B3zsrn9h+Zz46
c8PWlaURe+Aj/0n29wuxUQxC30TVd46005X3wDL4jh05/jPYVi/vhkPC30TVSoDgRePR/oz01Ioyk
699G9gzFhY6JuoUlnbTF56StzclDXY8j1ZVNY120xaZsws9E1UqaxtZvmcrLi5KWuwZXOyaGzvoaH

ZbtIyY2Ohb6LG2bZuTrZ0xWV/ftDyWEla1sVjxspC30SNXXHcnx+0IC+NqUledtmIm2aMLPRN1Kis
bSY5wUNZDM+UNRKvR1hS1G1H+mbMLPRN1Kisa2ZRQUBmZ5Q1kuVFWRw83UZHT7/bpZgoFN/fPSZqd
PUOsK+hleVzst0uxXXLi7PxqU2qYsbGQt9Ehar6Zvp9ysqS+O3PD1pWl1lHYOdxG8bKjJ6FvokKO4
83I4Id6QNpKYmUzU5nxwkLfTN6FvomKuw80cSCmWlkTE10u5SISKI4m6q6Fnr7fW6XYqKMhb6JeP0
DPnbVNB0yxI7yg1YWZ9PT72PfyVa3SzFRxkLfRLzqU2109g6wothCP6g88G+x07p4zChZ6JuIFww2
O9L/k9y0ZObmTLWTuWbUHIW+iKwWkUMiUiMiDw/x+o0isktE+kXk7kGvDYjI7sDX5sHrGjOSHcebK
MpOZWZ6fA6yNpwVxd1U1Dbj89nga8a5EUNfRLzAE8CdQB1wr4iUDWpWB3wceGqITXSp6pLA15px1m
vijKpSUdtsXTtDWFGSTWtXH4fPtbtDiokiTo70VwI1qnpMVXuBjcDa0AaqekJV9wJ2KYEJq6ONHTR
d7LXr84ewMtivb108ZhSchH4+UB/yvCGwzKkUEakQkW0i8sFRVWfi3o7j/rtO7Uj/nQqzp5CXnsJ2
C30zCk5mlh5q4PLRdCIWqeopEZkLvCQi+1T16NveQGQ9sB6gqKhoFJs2se7NYxeYkZMZSc5Ut0uJO
CLCqrnZ/LHmAqoat3MMmNFxcqTfABSGPC8ATj19A1U9Ffj7GPAKsHSINhtUtVxVy3Nzc51u2sQ4Ve
XNoxe4dt50C7RhXdt3Ouc7eqg51+F2KSZKOAn9nUCpiJSISBKwDnB0FY6IZ1lIcuBxDnA9cGCsxZr
4UnOug/MdPVw3b7rbpUSS6+b1AP7fiIxxYstQV9V+4EFgK3AQ2KSq1SLymIisARCRFSLSAHwE+J6I
VAdWvxKoEJE9wMvAV1XVQt84Egyya+fmuFxFJ5CrMnkJ+5hTePGqhb5xx0qePqm4Btgxa9mjI4534u
30Gr/cGcM04azRx6o2aC+RnTqEwe4rbpUQsEeHaedN58eBZfD7F47FuMHN5dkeuiUg+n7LtuPXnO3
Ht3Om0dPbx1hm7Xt+MzELfRKSDZ9po6eyz/nwHrg38G71x9LzL1ZhoYKFvIlKwj/paC/0Rzc6cQvH
0VLbZyVzjgIW+iUbjl2geHoqszKsP9+Ja+dNZ/uxJvoH7KZ4c3kW+ibi9A/42H6syY7yR+HaeTm0
9/Sz/1Sb26WYCGehbyLO7voW2nv6efcVdqOeU8FzH68dbnS5EhPpLPRNXPnD4UY8Au++wq7PdyypN
jJX56fzhyMW+ubyLPRNHNlyHkWF2aSkWrz4Y7GjaW57Kproa27z+1STASz0DcRpfliL3sbWrix1L
p2RuvG+bkM+JQ3auwqHjM8C30TUV6vOY+qP8DM6CwrymJqkte6eMxlWeiBiPKHw42kpySwuCDD7VK
iTlKCh2vn5fChw42o2hSKZmgW+iZiqCp/ONLIu0tzSPDaR3MsbpqqQ0NzF8fPX3S7FBOh7DvLRizD
Zzs429bDTda1M2Y3zZ8B+H9jMmYoFvomYgSDyvrzx65oeirF01N51ULfDMNC30SMFw+eZcHMNBt6Y
ZxuXjCDN45eoLO33+1STASy0DcRofliLxWlzdxeNtPtUqLe7WUz6en38foRG3XTvJOj0Ber1SJySE
RqROThIV6/UUR2iUi/iNw96LX7RORI4Ou+cBVuYssrh88x4FNus9Aft5U12aSlJPDiwbNul2Ii0Ii
hLyJe4AngTqAMuFdEygY1qWm+DjwlaN1s4CvAKmAl8BURyRp/2SbWvHDgLDPSklmUb5dqjlei18PN
C2bw+4P+H6TGHHJypL8SqFHVY6raC2wE1oY2UNUTqroXGDYU6x3AC6rapKrNwAvA6jDUBWJIT/8Ar
x5q5NYrZ9p0f2Fye9lMLlzsZXds9s9ulmAjJPTZgfgQ5w2BZU44WldE1otIhYhUNDBaVQfxZtuxJi
72DnB72Qy3S4kZn83PJcEjvHDgnNulmAjJJPShOvRy+jujo3VVdYOqlqtqeW6uXa4Xb148cJYpiV6
um2ejaoZLxpREV53Ntn598w5OQR8BKAX5XgCccrj98axr4oCq8uLbs9xQmKNKotftcmLKbVfOpOzc
h92da97GSejvBEFpFERekoB1wGaH298KvFdEsgIncN8bWGYMAHsaWjnd2m2Xak6A4L/p8/vPuFyJi
Sjhr6q9gMP4g/rg8AmVa0WkcdEZA2AiKwQkQbgI8D3RKQ6sG4T8E/4f3DsBB4LLDMGgof2nCLRK7
z3qjy3S4k5BVmpLCnM5Lm99sul+ZMEJ41UdQuwZdCyR0Me78TfdTPUuk8CT46jRhOjfd7lub2nuW1
+LhlTbMKUiXDXoln8828Ocqyxg7m509wux0QAuyPXuKayrpkzbd18YPFst0uJWXctmo0IPLf3tNul
mAhhoW9c8+yeUyQneLj1SuvPnyh5GSmsmJPNS3usi8f4WegbV/QP+Niy7zS3XjmDacmOehnNGH1g8
SyOnOvg0Jl2t0sxEcBC37hi+/Emznf0ctci69qZaKuvnoVHsKN9A1joG5c8u+cUU508vGeB3YU70X
LTkrLUxg7P7j110ygaC30z+Tp7+3lu72lWXz2LKUL2Q9Zk+ODSfGovdLLjuF0xHe8s9M2k+83e03T
09HPPisKRG5uweN81eUxLTuDNffUjNzYxzULfTLpNFfXmZnKimIbZXuypCYlsGbJbLbs001bd5/b
5RgXWeibSVVzroOdJ5r56IpCRGwY5cl0T3kh3X0+Nu+2E7rxzELfTKpfvNTj9QgfXuZ0dG4TLosKM
liY18Ym6+KJaxb6ZtL0Dfh4ZlCdtyycwYy0FLfLiTsiwKfLC9nb0MrB021ul2NcYqFvJs3vqs9yvq
OXdxYc1zUfWppPUoKHn2yrdbsU4xILfTNpvv/6MeZMT+VmuzbfNV1Tk/jgktk8s6uB5ou9bpdjXGC
hbyZFZW0zVXUtfPL6Erw2D66r7n/3XLr7fDy1o87tUowLLPTNpHjy9eOkpyRw9/IhR+A2k2hBXho3
lObwozd00Nvvc7scM8kchb6IrBaRQyJSiYIPD/F6soj8PPD6dhEpDiwvFpEuEdkd+PrP8JZvokF9U
ye/3X+aelcVMdUGV4sI97+7hHPTPtBBSHwaMfRFXAs8AdwJlAH3ikjZoGb3A82qegXwLeBrIa8dVd
Ulga8Hwls3iSL/9cyJPCJ8/Lpit0sxATfNz6V0xjR+8PpxG48nzjg5018J1KjqMVXtBTYCaWe1WQv
8KPD4aeBWsTtvDHCuvZufbq/lrkWzmJUxxelyTICI8Nc3zKX6VBsvvXXO7XLMJHIS+v1a6N0cDYfL
Q7YJzKnBckwPvFYi1lUi8qqI3DDOek2U+e4rR+kbUB66tdTtUswgH1qWz5zpqXzZd4fx+exoP144C
f2hjtGhf0KGa3MaKFLVpcAXgKdEJP0dbyCyXkQqRKSisbHRQUkmGpxq6eKn2+r4s2X5Nj9rBER0ev
j8baUCON3G89Vn3C7HTBINod8AhN5NUwAMPvtzqY2IJAazQJOq9qjqBQBVrQSOAvMHv4GqblDVclU
tz83NHflemIj0Hy/XoCifvcWO8iPVmsX5XDFjGo+/cJgBO9qPC05CfydQKi1lIpIERAM2D2qzGbgv
8Phu4CVVVRHJDZwIRkTmAqXAsfCUbiJZ3YVONu2sZ92KIgqzU90uxwzD6xG+cPt8as518Kuqk26XY
ybBiKEf6KN/ENgKHAQ2qWql1DwmImSczX4ATBeRGvzDOMHLOm8E9orIHvwnB9QVZvFIQ489twBEr
0ePvOeK9wuxYxg9VV5XJOfwde3vkVHT7/b5ZgJJPf2uVZ5eblWVFS4XYYZhxcPnOWv/ruCh+9cyAM
3zXO7HONAVV0zH/7uG9x/fQlFvmvWfDkmGohIpaqWj9TO7sg1YdXVO8A/PltN6YxpfPL6ErFLMQ4t
Lcpi3YoifvjGcd46YyNwxjILfRNW33mlhobmLh5bezVJCfbxiiz/d8cC0lMS+Idf7bdLOGOYfVeas
N1T38J3XznKh5bmc+09a40vAAAKpEleQVS86SOvYCJk1tQkHnnflew80cwP3zjhdjlmgljom7Do60
nnoY1VzExP4R8/cJXb5Zgx+sJyAm4vm8nXfvsW+0+2ul2OmQAW+iYsHv31fuqbOvm3dUvISE10uxw
zRiLC1/5sEV1TE3l0YxWdvXY1T6yx0Dfj9ouKen656ySfvaWUFcXZbpdjxi17ahLf+ugSjp+/yJd/

td8GZIsxFvpmXN6oOc8jv9zH9VdM57032DX5seK6K3L4/K3z+eWuk3z7pRq3yzFhZIObmzE7fLadT
/2kkrm5U/nux5aT4LVjiFjy0K1XUNt0kcdfoExB1hQ+vMwmWIkFFvpmTE6cv8jHn9zB1EQvP/zESt
JTrB8/1ogIX/3wIs60dvPFZ/aSnpLibWUz3S7LjJMdmp1RO3SmnY987026+gb44SdWkJ9p4+THqqQ
ED9/92HLKZqXzwE8q+fVuG58n2lnom1GprG3mng1v4hHY9K1ruWp2htslmQmWMSWRn/zVKpbPyeLz
P9/Nj9884XZJZhws9I0jqsoP/3icdRveJD0lkacfuI7SmWlul2UmSVpKIj/65Eres2AG//Drav72F
3vo6h1wuywzBhb6ZkQXOnp48Kkq/u+zB7hpf17PPvhuGy45DqUkevl//6uch24t5ZldDXzo03/k4G
kbpYfaWOibYf18ys921HHLN19la/UZvrh6IRv+stxuvopjwfH3/+sTK2ls7+Gub7/OPz93wIZkjiI
2tLJ5B59P+d2BM3z7pRqqT7WxsiSbf/ngldadY96mpbOXrz1/iJ/tqCM3LZ1P3TiXP19VRGqSXRT
BqdDK1vom0uaL/by690n+en2Oo6c66AkZyqfu7WUtUtmIzLUNMjGwK66Zv516yHeOHqB7K1JrFtRy
N3LC2xe5EkW1tAXkdXAvwNe4Puq+tvBrycD/w0sBy4A96jqicBrjwD3AwPAQ6q69XLvZaE/uc61df
PyoX08ePAcrx5qpHfAx9X56fz1DX05a9FsvB4Le+NMZW0T3331KC+9dQ6fwrKiTG4vy+PWK2dQOmO
aHThMsLCFfmCO28PA7fgnQN8J3KuqB0La/G9gkao+ICLrgA+p6j0iUgb8DFgJzAZeBOar6rCn/S30
J05LZy+Hz3Zw6Gw7e+pb2FXbzLHzFwGYNZCHCHVfn8ZHLhZTNTne5UhPNzrZ188tdJ312zykOBE705
qYls6wok6VFWsZMS2NBXhp56Sn2gyCMnIa+k8631UCNqh4LbHgjsBY4ENJmLFCPgcdPA/8h/v/Ntc
BGVe0Bjgfm0F0Jv0l0R8zbqSr9PqW330d33wBdfQN09Q7Q0dPPxZ4BWrV6aOnqpfliL+faezjX1sP
Jli7qmzt6ey7tJ3sqUksK8rioysKuX1BLgtmptk3oAmLmekpfPrmeXz65nmcae3mlUPn2HG8icq6
ZrZwN73Ubqk18LsKRRkpTIzPZncacnkpCWTMSWR9CmJpCUnMDU5gWnJCSQnekhJ9JKc4CHJ67HP6
jg4Cf18oD7keQOwarg2qtovIq3A9MDyBYPWzR9ztZfR0tnLR/4z/D9LRnvGY7jfnDTwhw5q61NQFF
UCX/51P1V8gYaf8Cn9A/6/+3w+nJ6GSU9JIDctmfysVBYVZFCUncr8mWmUzpxGfuYU+8YxEy4vI4V
1K4tYt7II8J83Ony2ncNn2zl+vpP65k4amrvY29DKhYs9jj/biV7B6xESPR48HiHBI3g8glcEj/iH
kBDB/0XgMVz6zMulPy799TZj+d4Ix3fTwlnpfPvewWHY0vCchP5Q+zL4v2a4Nk7WRUTWA+sBioqKH
JT0Th6PUDpzYk4cyWj/04dpHvzQhb78tg9o4MPpDTz3evwfbI9I4EPuIckrJHo9JCX4j3ymJHpJSf
IyLdnL1KQEMLOTyJiSSGZqIimJ3rHusjETImtqEqvmTmfV3HfOrNY/4KO5s4/Wrj5au3rp6BngYk8
/F3v66e730d07QO+Aj95+H70DPv9B0IAPn08ZUP9Bkc8HAXo4iLp0MKUoXPqB4n+slx6/wxiubdGx
rDSEwqyJH9LESeg3AIUhzwuAU800aRCRBCADaHK4Lqq6AdgA/j59p8WHSk9J5Dt/sXwsqxpjIkCC1
0NuWjK5aclulxLTnNyctRMOfZESEUkC1gGbB7XZDNwXeHw38JL6f5RuBtaJSLKilAClwI7wlG6MMW
a0RjzSD/TRPwhsX/J5pOqWi0ijwEVqroZ+AHw48CJ2ib8PxgItNuE/6RvP/CZy125Y4wxZmLzZvN
GGBMDnF6yaWPvGGNMHLHQ8aYOGKhb4wxccRC3xhj4oiFvjHGxJGIu3pHRBqB2nFsIgc4H6Zy3Gb7
EplsXyJTL00LjH5/5qhq7kiNiI70x0tEKpxcthQNbf8ik+1LZlqlfYgJ2x/r3jHGMDhioW+MMXEKf
kN/g9sFhJHtS2SyfYlMsbQvMEH7E3N9+sYY4YXi0f6xhhjhGToS8inxWRQyJLSLJfd7uecBCrvx
URFZEct2sZKxH5hoi8JSJ7ReR/RCTT7ZpGS0RWBz5bNSLysNv1jJWIFIrIyyJyMPB98jm3axovEfG
KSJWIPOd2LeMhIpki8nTge+WgiFwbzu3HXOliLyHvwz827SFWvAv7V5ZLGTUQK8U9MX+d2LeP0AnC1
qi4CDgOPuFzPqIiIF3gCuBMoA+4VktJ3qxqzfuBvVPVK4F3AZ6J4X4I+Bxx0u4gw+HfgeVVdCCwmz
PsUc6EPfBr4amAydlT1nMv1hMO3gL9jTBO5RQ5V/Z2q9geebSM/k1o0WQnUqOoxVe0FNuI/wIg6qn
paVXcFhrfjD5YJmb96MohIAfB+4PtulzIeIpIO3Ih/jhJUtdVdVW8L5HrEY+vOBG0Rku4i8KiIr3C5
oPERkdXBSVfe4XUuYfRL4rdtFjFI+UB/yvIEoDsogESkGlgLb3a1kXP4N/4GRz+1Cxmku0Aj8MNBV
9X0RmRrON3AyR27EEZEXgbwhXvoS/n3Kwv8r6wpgk4jM1Qi+TGMe/fl74L2TW9HYXW5fVPPXgTZfw
t+98NPJrC0MhpryPmI/V06IyDTgGeDzqtrmdj1jISJ3AedUtVJEbna7nnFKAJYBn1XV7SLy78DDwD
+E8w2ijqreNtxrIvJp4JeBkN8hIj78Y1g0TlZ9ozXc/ojINUAJsEdEwN8dsktEVqrqmUks0bHL/d8
AiMh9wF3ArZH8g3gYDUBhyPMC4JRLtYybiCTid/yfquov3a5nHK4H1ojI+4AUIF1EfQKqH3O5rrFo
ABpUnfhhb19P4Qz9sYrF751fALQAIh9IiKoHYVLVfao6Q1WLVBuY/wdiWaQG/khEZDXwRWCNqna6X
c8Y7ARKRaRERJLwzwW92eWaxkT8RxE/AA6q6uNu1zMeqvqIqhYEvkfWAS9FaeAT+N6uF5EFgUW34p
9jPGyi8kh/BE8CT4rIfqAXuC8Kjyhj1X8AycALgd9ctqnqA+6W5Jyq9ovIg8BWwAs8qarVLpclVtc
DfwnsE5HdgWV/r6pbXKzJ+H0W+GngwOIY8IlwbztzuyDXGmDgSi907xhhjhmGhb4wxccRC3xhj4oiF
vjHGxBELfWOMiSMW+sYYE0cs9I0xJo5Y6BtjTBz5//k9Tn9uZMHnAAAAElfTksuQmCC\\n\\",\\n"

```
,
"      \"text/plain\": [\\n\",
"      \"<matplotlib.figure.Figure at 0x7f4b1883bb00>\"\\n\",
"      ]\\n\",
"      },\\n\",
"      \"metadata\": {},\\n\",
"      \"output_type\": \"display_data\"\\n\",
"    }\\n\",
"  ],\\n\",
"  \"source\": [\\n\",
"    \"# Define a Gaussian distribution function (see Statistics 1
notebook)\\n\\n\",\\n\",
```

```

"    \"def Gaussian(xvalues, mean, standard_deviation):\n\n",
"    \"    sdsquared = standard_deviation**2\n",
"    \"    exponent = -(xvalues - mean)**2/(2 * sdsquared)\n",
"    \"    preexponential = 1 / np.sqrt(2 * np.pi * sdsquared)\n",
"    \"    return preexponential * np.exp(exponent)\n",
"    \"\n",
"    \"# Generate a list of x values\n",
"    \"x = np.arange(-6, 6, 0.1)\n",
"    \"\n",
"    \"# Generate a Gaussian for a given mean and standard
deviation\n",
"    \"mean = 0\n",
"    \"std_dev = 1\n",
"    \"y = Gaussian(x, mean, std_dev)\n",
"    \"\n",
"    \"# Plot the result\n",
"    \"\n",
"    \"plt.plot(x, y)\n",
"    \"plt.show()\n",
"    ]\n",
" },\n",
" {\n",
"     \"cell_type\": \"markdown\",\n",
"     \"metadata\": {},\n",
"     \"source\": [\n",
"         \"Before we write our function, we need to introduce the idea of a\n",
"         \"for\" loop. For loops keep repeating a specific operation for a\n",
"         specific number of times. In Python, for loops are defined by a variable\n",
"         cycling through an array or list. In the example below, the variable is $i$\n",
"         and the array is $indexes$. On the first iteration of the loop, $i=0$ (the\n",
"         first element in the $indexes$ array). It will execute the block of code that\n",
"         is indented and then go on to the next element in the $indexes$ array, $i=1$.\n",
"         The variables inside the for loop can get overwritten and change. To exit the\n",
"         for loop, remove the indent. The code below loops through $indexes$ and\n",
"         prints the square of each index as it goes.\n",
"         ]\n",
"     },\n",
"     {\n",
"         \"cell_type\": \"code\",\n",
"         \"execution_count\": 6,\n",
"         \"metadata\": {},\n",
"         \"outputs\": [\n",
"             {\n",
"                 \"name\": \"stdout\",\n",
"                 \"output_type\": \"stream\",\n",
"                 \"text\": [\n",
"                     \"Starting to loop through elements in indexes: [0 1 2 3\n",
"4]\n",
"                     \"The value of k in this loop is: 0\n",
"                     \"Running Some Code...k squared is: 0\n",
"                     \"The value of k in this loop is: 1\n",
"                     \"Running Some Code...k squared is: 1\n",
"                     \"The value of k in this loop is: 2\n",
"                     \"Running Some Code...k squared is: 4\n",
"                     \"The value of k in this loop is: 3\n",
"                     \"Running Some Code...k squared is: 9\n",
"                     \"The value of k in this loop is: 4\n",

```

```

"    \"Running Some Code...k squared is: 16\\n\\n\",\\n\",
"    \"For loop done\\n\\n\\n\",
"    ],\\n\",
"    },\\n\",
" ],\\n\",
" \"source\\\": [\\n\",
"     \"indexes = np.arange(0, 5)\\n\\n\\n\",\\n\",
"     \"print(\\\"\\\"\\\"Starting to loop through elements in indexes:\\\"\\\"\\\",
indexes)\\n\\n\\n\",\\n\",
"     \"for k in indexes:\\n\\n\\n\",\\n\",
"         print(\\\"\\\"\\\"The value of k in this loop is:\\\"\\\", k)\\n\\n\\n\",\\n\",
"         \"s = k**2\\n\\n\\n\\n\",\\n\",
"         \"print(\\\"\\\"\\\"Running Some Code...k squared is:\\\"\\\", s)\\n\\n\\n\\n\",\\n\",
"         \"print(\\\"\\\"\\\"For loop done\\\"\\\")\\n\\n\\n\\n\",
"     ],\\n\",
" },\\n\",
" {\\n\",
"     \"cell_type\\\": \"markdown\\\",\\n\",
"     \"metadata\\\": {},\\n\",
"     \"source\\\": [\\n\",
"         \"Now, let's write a function to calculate the integral. Our data
point area function comes directly from the trapezoid rule above. Our
integral function will just apply this to each data point in the $x$ and $y$
arrays using a for loop. Note, we need to start at the second element of the
list rather than the first element because we need to compare two data
points, $k$ and $k-1$. \\n\\n\\n\",
"     ],\\n\",
"     },\\n\",
"     {\\n\",
"         \"cell_type\\\": \"code\\\",\\n\",
"         \"execution_count\\\": 7,\\n\",
"         \"metadata\\\": {},\\n\",
"         \"outputs\\\": [\\n\",
"             {\\n\",
"                 \"name\\\": \"stdout\\\",\\n\",
"                 \"output_type\\\": \"stream\\\",\\n\",
"                 \"text\\\": [\\n\",
"                     \"0.999999997112\\n\\n\\n\\n\\n\",
"                 ],\\n\",
"             },\\n\",
"         ],\\n\",
"         \"source\\\": [\\n\",
"             \"# A function to calculate the trapezoid area for two points. This
comes directly from the code above.\\n\\n\\n\\n\",\\n\",
"             \"def data_point_area(x1, x2, y1, y2):\\n\\n\\n\\n\\n\",
"             \"    average_y = (y1 + y2)/2\\n\\n\\n\\n\\n\",
"             \"    delta_x = (x2-x1)\\n\\n\\n\\n\\n\",
"             \"    area = average_y * delta_x\\n\\n\\n\\n\\n\",
"             \"    return area\\n\\n\\n\\n\\n\",
"             \"\\\"\\\"\\n\\n\\n\\n\\n\",
"             \"# A function to calculate the integral for an array\\n\\n\\n\\n\\n\",
"             \"def integral(x, y):\\n\\n\\n\\n\\n\",
"             \"    length = len(x) # Calculate the length of the x
array\\n\\n\\n\\n\\n\",
"             \"    indexes = np.arange(1,length) # Create an array of indexes.
Don't start at 0 but 1.\\n\\n\\n\\n\\n\",
"             \"    sum_value = 0 # Set the sum value to 0 initially\\n\\n\\n\\n\\n\",

```

```

    "    \"        for k in indexes:\\n\\\",\\n\",
    "    \"        # Get the correct x and y values from the list based on
their indexes\\n\\\",\\n\",
    "    \"        x1 = x[k-1]\\n\\\",\\n\",
    "    \"        y1 = y[k-1]\\n\\\",\\n\",
    "    \"        x2 = x[k]\\n\\\",\\n\",
    "    \"        y2 = y[k]\\n\\\",\\n\",
    "    \"        # Calculate the area for those set of two points from the
function above\\n\\\",\\n\",
    "    \"        area = data_point_area(x1, x2, y1, y2)\\n\\\",\\n\",
    "    \"        # Add the area from the two points to the total
sum\\n\\\",\\n\",
    "    \"        sum_value = sum_value + area\\n\\\",\\n\",
    "    \"        return sum_value\\n\\\",\\n\",
    "    \"\\n\\\",\\n\",
    "    \"print(integral(x, y))    \"\\n\",
    "    ]\\n\",
    " },\\n\",
    " {\\n\",
    "     \"cell_type\\\": \"markdown\\\",\\n\",
    "     \"metadata\\\": {},\\n\",
    "     \"source\\\": [\\n\",
    "         \"The integral should be almost exactly one. That is because the

```

area of a Gaussian distribution is normalized to 1. Numpy also has a trapezoidal integral function, trapz. You can see below that our homemade function should match this. Note, this function takes x as the second argument because x is optional in this function. If you don't have x , it will assume Δx is 1 for each data point.

```

    "     ]\\n\",
    "     },\\n\",
    "     {\\n\",
    "         \"cell_type\\\": \"code\\\",\\n\",
    "         \"execution_count\\\": 8,\\n\",
    "         \"metadata\\\": {},\\n\",
    "         \"outputs\\\": [\\n\",
    "             {\\n\",
    "                 \"name\\\": \"stdout\\\",\\n\",
    "                 \"output_type\\\": \"stream\\\",\\n\",
    "                 \"text\\\": [\\n\",
    "                     \"0.999999997112\\n\\\",\\n\",
    "                 ]\\n\",
    "             }\\n\",
    "         ],\\n\",
    "         \"source\\\": [\\n\",
    "             \"print(np.trapz(y, x=x))\\n\\\",
    "         ]\\n\",
    "     },\\n\",
    "     {\\n\",
    "         \"cell_type\\\": \"markdown\\\",\\n\",
    "         \"metadata\\\": {},\\n\",
    "         \"source\\\": [\\n\",
    "             \"The final thing we need to learn to put it all together is how to

```

chop out specific parts of the data. I've included a function to do this using Boolean indexing. Basically, it grabs all the x values that are both greater than x_{min} and less than x_{max} and chops the array down to include only the values within that range.

```

    "     \"\\n\\\",\\n\",

```

```
"    \"After doing that, we can calculate the area under that part of the
curve only. \\\"\\n\",
"    ]\\n\",
"    },\\n\",
"    {\\n\",
"    \"cell_type\\\": \"code\\\",\\n\",
"    \"execution_count\\\": 10,\\n\",
"    \"metadata\\\": {},\\n\",
"    \"outputs\\\": [\\n\",
"    {\\n\",
"    \"name\\\": \"stdout\\\",\\n\",
"    \"output_type\\\": \"stream\\\",\\n\",
"    \"text\\\": [\\n\",
"    \"Integral from -1 to 1 is: 0.656883274351\\\"\\n\\\"\\n\",
"    ]\\n\",
"    },\\n\",
"    {\\n\",
"    \"data\\\": {\\n\",
"    \"image/png\\\":
\\\"iVBORw0KGgoAAAANSUHeUgAAAX0AAAEKCAyAAAD+XoUoAAAABHNCSVQICAgIfAhkiAAAAAlwSFlz
AAALEgAACxIB0t1+/AAAIABJREFUeJzt3Xt41PwD9/H3N4FwEDAoCBJOQQ4rgqKGcBITaAVcBavu
Ag2ubbGsq/Rg3avVWmmXXbbaPqtbFayVFp+1bRRFbcVWpCKgYCAQBNGAKCBIAOW4CoRTku/zx0xwD
ImZJJPcc/i8rouLmXvuO/MZTT7cuQ+/n7k7IiKSGtKCDiAiIk1HpS8ikkJU+iIiKUSlLYKSQlT6Ii
IpRKUvIpJCVpoiIilEpS8ikkJU+iIiKaRZ0AGq6tChg/fs2TPoGCIiCWxNmjX73L1jbevFXen37Nm
ToqKioGOiICQUM9sezXo6vCMikkJU+iIiKUSlLYKSQlT6IiIpJKrSN70xZrbJzDab2dlfst5NZuZm
lhOx7J7wdpvMbEwsQouISP3UWvpmlg48CowD+gOTzax/Neu1Bb4LFEYs6w9MAi4AxpK/Dn89kcSTn
w89e0JaWujv/PygE4nUWTR7+rnAZnff6u4ngLnAhGrW+3fgF8CxiGUTgLnufztzdPwQ2h7+eSOLiz4
cOHWDKFNi+HxdDf998M5jpHwBJKNGUfhawI+J5SXjZKWZ2MdDN3f9S121F4lp+PkybBvv3n/5a5VS
j27eH1lHxSwKIpvStmmWnJtYlSzTgIeCuum4b8TWmmVmRmRXt3bs3ikgiTeTee6G0tPb1SktD64rE
uWhKvwToFvG8K7Ar4nlbYACw1My2AUOB+eGTubVtC4C7z3b3HHfP6dix1ruIRRpF+Pi9b4/qJseQ7
du1ty9xL5phGFYDfcsG9hJ6MTs1ytfDpDpgQ6Vz81sKfCv715kZkeBp8zsQaAL0AdYFbv4Io2g8p
BOaWmlv6p+qWnTQn/n5cU6lUhM1Lqn7+5lwHRgIbAReNbdi81sppmNr2XbYuBZYAPwCnCHU5c3PLZ
II4r2ke51dJhH4py5n3aIPVA50TmuAdckUGlpn5+krpHD5g1K/R4ypTqlzGDiorGySZSAzNb4+45
ta0Xd6NsigSue/fQ8fmqevSAbds+f37vvdWv1717o0UTaSGNwyBSKeLk7Wn7+a1bf76HX2nWrNDyC
A6hfwH07b7EKZW+CHx+8nb7dozQtcanir9HD5g9+/STs3l5oeU9euDh9U+d+NW1+xKndExfBEJ75t
Ec0mms7UUAkNpj+trTFwH46KO6LY/19iJNRKUvAjWffi32pGxDtxdpIip9EYBZsyhvVuVitupO3n7
J9lVP6tZpe5EmotIXyc+He+8lrazs1AnZGk/elqSak7peeaOWTuZKHNF1+pLaqhtyoXIPva5DKYTX
9299i7QTJ0LLKq/iiXhdJEi6ekdSW6yvutFVPBIQxb0jEo1YX3Wj3qgkzqn0JbXF+qobXcUjCU6lL
6ltlizKrcOayg256kZX8UicU+1LSvPJkzkIHE9LC42OWderdqqqchUPwP677tJJXIKbKn1Jae8/9x
wd3Fk1ZUpoOORt2xpe0Hl5sG0bHy1aBMC777zT8KAiMaLS19QUH1Gz78SJOHBR//4xf4seV17Jrmb
NGPbii6Ex+jXypsSBqErfzMAa2SYz22xmd1fz+m1m9o6ZrTOz5WbWP7y8p5kdDS9fZ2a/ifUHEKmz
akbUbDdzZuwLOT+fThUVZLiHJmXRYJsSB2q9Tt/M0oH3ga8Smuh8NTDZ3TdeRNPO3T8LPx4P3O7uY
82sJ/AXdx8QbSBdpy+Nrqmupdc1+9KEYnmdfi6w2d23uvsJYC4wIXKFysIPOwNon4NCJG401bX0um
Zf4lA0pZ8F7Ih4XhJe9gVmdoeZbQF+AXw34qVsM1trZq+b2cjq3sDMpp1ZkZkV7d27tw7xReqhqa6
1lzX7EoeiKX2rZtlpe/Lu/qi7nwf8CPhJePFuoLu7Xwz8AHjKzNpVs+1sd89x95yOHTtGn16kPmbN
wlul+uKyxriWXtfsSxyKpvRLgG4Rz7sCu75k/bnA9QDuftzd94cfrwG2AH3rF1UkRvLy+PC664B6j
qhZh/dh9mwqsrJw4FhGRuO8j0gdRFP6q4E+ZpZtZhnAJGB+5Apm1ifi6d8DH4SXdwfCMbMegF9gK
2xCC7SEB9t3cpxoPSTT2JzbX5N8vJIKylhfbt2bG3WTIUVgau19N29DJgOLAQ2As+6e7GZzQxfqQM
w3cyKzWbdocM4t4SXXw6sN7O3geeA29z9QMw/hUgdnfvuuxS3b88Z55zTJO93cPBg/q60lH0bNzbJ
+4nUJKrx9N39ZeDlKstmRDz+Xg3bPQ8835CAIrH2ybp19Dt2jKVf+UqTvWfHvDzSXnuN9x97jA4PP
9xk7ytSle7IldSSn8+ZI0MXkY0oKmqyG6XOT0ujHBj2yCO6M1cCpZmzJHWE78RtWVoKQPN9+5pmVq
v8fNJuV/3z55pNSwKkmbMkdQR1h6zuzJUmoJmzRKO6g5Z3ZkrcUS1L6kjQdtkdWeuxBGVvqSOWM+
SVYf3rXpnruxOXAmISl9Shk+ezGF3jqWnx2aWrGhVM5vWnttv10lcCYRKX1LGtldf5UxglarJsZs1
K1rh2bS2/+1vAGza9WUjmYg0HpW+pIwd4Wvjuwa4h93jyiv5OC2N9DffDCyDpDaVvqSM9GXL2JOWR
vaYMYFlsLQ0tnbtSq8dO/CKisBySOpS6Uvyy8/He/Rg+LZttDPDnn460Djts7I4t6ICmjXT3bnS5H
RHriS38F24Fr4Lt2V5ebB3w+bn0++ttwCwyHlZg8ojKUD35Epyi7e7YeMtjyQN3ZErAvF3N2y85ZG
Uo9KX5BZvd8PGWx5JOSp9SW6zZuEtW35xWZB3w2reXAlYVKVvZmPNbJOZbTazu6t5/TYze8fM1pnZ
```

cjPrH/HaPeHtNplZcNfKSWrKy2PL1VcDjTwfbh3yMHs23q1baN7c5s01b640qVpP5IbnuH0f+CqhS
dJXA5PdfUPEOu3c/bPw4/HA7e4+Nlz+TwO5QBdgEdDX3ctrej+dyJVYWzp0KMMLCynfv59WZ50VdJ
xT1mVm0qysjAGHDwcdRZJALE/k5gKb3X2ru58A5gITiLeoLPyWM+DUECMTgLnufztzdpWq2h7+eSJN
pv3Ejm9q0iavCB/jf/v3pd+QIRw9o2mhpOtGUfhawI+J5SXjZF5jZHWa2BfgF8N06bjvNzIrMrGjv
3r3RZhep1fHPPqPfZ5+Xv1+/oKOcpvVVV9EceF83Z0kTiqb0rZplpx0TcvdH3f084EfAT+q47Wx3z
3H3nI4d00YRSSQ6Hz7LC2BlqNHBx3lNL1vvhmAg3/5S8BJJJVEU/olQLeI512BLxsicC5wft23FY
md/Hx6fzf0S+fgP/4x7oY7OKtPHz5JT2fYokWQlqYhGaRJRFp6q4E+ZpZtZhnAJGB+5Aplifi6d8
DH4QfzwcmmVklM8sG+gCrGh5bpBaVk6AfPQpA+u7doeEO4qlU8/PpUFFBi4oKiBySIZ4yStKJahgG
M7sG+G8gHXjC3WeZ2UygyN3nm9mvgKuAk8BBYLq7F4e3vRf4FlAGfN/dF3zZe+nqHYmJRBjuIBeYS
sKI9uodjb0jySktLbT3XJVZaAKVeJAIGSVhaOwdSW2JMNxBImSupKPS1+QU1CTodaEhGSQAKn1JTn
157GrenBNmTTsJel2Eh2Q41qIFDlRkZcVfRkk6Kn1JSkcPHqTTiRMU5OY2/STodZGXx7v33YcBb99
5Z3xmlKSi0pek9MG8eWQArUaNCjpKrXp//esAfLpwYcBJJBWo9CUphfjrXwHoOXFiwElql5mdzZaM
DFqtXx90FEkBKn1JShlr11KSnk6nQYOCjhKVXd2702vPHlyXakoJ+1LUUq+axcfdekSdIyoeW4uH
d3Z8cYbQUeRJKfS1+SSn0/ZuefStbycs/btS5ghDXqHh33uNmQUxuCRRTUs6AAiMRMeb6dZaSlAaN
ydadNcR8XzVTH5+Zz7xBNAefjayjF4IL5zS0LSMAySPBJ1LJtEzS1xRcMwSOr56K06LY8XiZpbEpJ
KX5JHoo5lk6i5JSGp9CV5zJpFRcuWX1yWCGPZaAweaUIqfUkeeXkszcTjG+DxOt5OdcJj8NCjBw4c
S09PjNySkKIqfTMba2abzGyzmdldzes/MLMNZrbezF4zxs4Rr5Wb2brwn/lvtXWJpaeAS886C8rL4
3e8nerk5cG2bRR26cInZomTWxJOraVvZunAo8A4oD8w2cz6V1ltLZDj7hcCzwG/iHjtqLsPCv8ZH6
PcItUqLCwkNzcXqzqscoI4dtFF9CgrY9/GjUFHkSQVzZ5+LrDZ3be6+w1CE59PiFzB3Ze4e2n46Up
CE6CLNK1Dhw5RXFzMKCFDgo5Sb5ljxgDw4TPPBjXeklU0pZ8F7Ih4XhJeVpOpQOQ8uC3NrMjMVprZ
9fXIKBKVNWvW4O4JXfq9J06kHDIyeHHQUSRJRNXHbnW/J1d7R5eZTQFygCsiFnd3911mlgtYbGbvU
PuWktNA6YBdNdla1JPhYWFAAwePDjgJPXXpnNn3m/ZkYbNgQdRZJUNHv6JUC3iOddgV1VVzKzq4
B7gfHufxyubvvCv+9FVGKXfx1W3ef7e457p7TsWPHOn0AkUqFhYWcd955d0jQIegoDVKemcnF+/f
jaWkah0diLprSXw30MbNsM8sAJgGffuArHzC4GHidU+Hsilrc3sxbhxx2AEYB2YaRRrFq1KqEP7QCQ
n0/fvXtJB8z983F4VPwSi7WWvruXAdOBhCBG4F13LzazmWZWeTXOL4E2wLwql2aeDxSZ2dvAEuB+d
1fpS8zt3LmTnTt3Jn7p33sv6eXlX1xWWgr33htMHkk6UY2y6e4vAy9XWTYj4vFVNWxXAAXsSECRaF
Qez8/NzQ04SQNpHB5pZLojV5JCYWEhzZs3Z1CCzJRVI43DI41MpS9JYdWqVQwaNIiWVcfeSTQah0c
amUpfEl55eTlFRUWJfzWfTo3Dc6RDBYqA4507axweiSmVviS8DRs2cPjw4eQofYc8PP533TrSgcFv
uUeFLzG10peEV3kSN21KH8jKyIrk+uVzx0JFZW+JLzCwkLat29P7969g44SU0OGDFHps8yp9CXhJ
frImjXJzc1ly5Yt7Nu3L+gokkRU+pLQDh8+nPAja9ak8jOtXr064CSSTFT6ktDWrFlDRUVFUpZ+Tk
4OaWlpOsQjMaXSL4SWNHfiVqNNmzZccMEfKn2JKZW+JLRkGVmzJkOGDGHVqlW4VzuauUidqfQloVW
exE1WQ4YM4cCBA2zevDnoKJIKvpqSsEpKsti5cyfDhg0LokqjGtp0KAARv64MOIkKc5W+JKzKIqws
xmR0/vnn07ZtW5W+ixKXxLWypUradGiBRdddFHQURpNeno6ubm5Kn2JGZW+JKzCwkIuueQSMjIyg
o7SqIYMGcL69espLS0NOookgahK38zGmtkmM9tsZndX8/oPzGyDma03s9fMrEfEa7eY2QfhP7fEMr
ykrpMnT1JUVJTUh3YqDR06lLKyMt56662go0gSqLX0zSwdeBQYB/QHJptZ/yqrrQVy3P1C4DngF+F
tzwJ+CgwBcoGfmln72MWXVLV+/XqOHTuWEqVfeeOZDvFILESzp58LbHb3re5+ApGLTIhcw2XuHv1
754rga7hx2OAV939gLSfBF4FxsYmuqSyVDiJW+mcc86hV69eKn2JiWhKPwvYefG8JLysJ1OBBfXcV
iQqK1eu5Nxxx6Vbt25BR2kSQ4cOvelLTERT+tUNXvj27YFmNgXIAX5Z123NbJqZFZ1Z0d69e60IJK
lu5cqVDBkyJO1G1qzJkCFD2L1zJyU1JUFHkQQTEmXAJG7U12BXVVXMrOrgHuB8e5+vC7buvtsd89
x95yOHTtGm11S1L59+9i8eXNKHnqppJu0JFaiKf3VQB8zyzazDGASMD9yBTO7GHicUOHviXhpIXC1
mbUPn8C9OrxMpN6Scaas2gwaNIgWLVqWysWKOknIgmtW2wruXmZm0wmVdTrwhLsXm9lMoMjd5xM6n
NMGMbf+dfsJdx/v7gfM7N8J/cMBMNPdDzTKJ5GUsWLFctLT0xk8eHDQUZpMRkYGl156qfb0pcFqLX
0Ad38ZeLnKshkrj6/6km2fAJ6ob0CRqgoKChg0aBBnnHFG0FGa1PDhw3n44Yc5fvw4LVq0CDQOJCj
dkSsJpaysjMLCQoYPHx501CY3fPhwTpW4oZu0pEFU+pJQKocjSMXSrxxNtKCgIOakKshU+pJQKgsV
FUu/c+fO9OrVS6UvDaLSl4RSUFBAV1ZWYtyUVdXw4cMpKCjQTFpSbyp9SSgFBQUHmZ48ZW7Kqmr48
OF8/PHHbNu2LegokqBU+pIwdu3axfbt21Py0E6lys+uQzxSXyp9SRiVNyalcukPGDCANm3a6CYtqt
eVviSMgoICWrZsyaBBg4KOEpj09HSGDBmiPX2pn5W+JIYcggJycnKSfqas2gwfPpy3336bQ4cOBR1
FEpBKXxJCAWkpRUVFjBgxIugogRsxYgQVFRUakkHqRAUvCaGwsJCysjJGjhwZdTADRs2jLS0NJYt
WxZ0FE1AKn1JCMuWLCpMtKcPtGvXjkGDBqn0pV5Up+IQ1li1bxsCBA8nMzAw6S1wYOXIk1leu5MSJE
0FHkQSj0pe4V1ZWxooVK3RoJ8LlIkSM5duwYa9asCTqKJBiVvsS9tWvXcuTIEZV+hMsuwxh3ikz1
T6Evcqi021/71OnTrRt29flb7UWVSlb2ZjzWyTmW02s7uref1yM3vLzMrM7KYqr5Wb2brwn/lvtXW
pzbJly+jVqxddunQJokpcGTlyJG+++SYVFRVBR5EEUmvpm1k68CgwDugPTDaz/lVW+wj4BvBUNV/i
qLsPCv8Z38C8kmLcneXLl2svvxojR47k4MGDFBcXBx1FEkg0e/q5wGZ33+ruJ4C5wITIFdx9m7uvB
7TLITH13nvvsW/fPpV+NSr/m+gQj9RFNKWfBeyIef4SXhatlmZWZGYrrez6OqWtlKfj+TXLzs4mKy
uLN954I+gokkCimRi9uoHL6zKDQ3d332VmvYDFZvaOu2/5whuYTQOmAXTv3r0OX1qS3ZiLSzj33HP
p06dP0FHijplxxRVX8Npr+HuKTvHgNRNNHv6JUDkNEVDgV3RvoG77wr/vRVYClxczTqz3T3H3XM6
duwY7ZeWJOfuLFmyhFGjRqnQajBq1Cg++eQTNm7cGHQUSRDR1P5qoI+ZZZtZBjAJiOoqHDNRb2Ytw

o87ACOADfUNK6ll48aNfPLJJ4wePTroKHGr8r/NkiVLak4iiaLW0nf3MmA6sBDYCDzr7sVmNtPMxg
OY2WazKwH+AXjczCovJzgfKDKzt4ElwP3urtKXqFQW2ahRowJOEr+ys7Pp3r27S1+iFs0xfdz9ZeD
lKstmRDxeTeiwt9XtCoCBDcwoKWrX4sX06NGD7OzsoKPELTNj9OjRzJ8/n4qKcTLsDL+lfd19h0hc
qqioYOnSpTqeH4VRo0Zx4MAB3nnnnnCjSAJQ6UtcWr9+PQcOHNDx/ChUHv5avHhxwEkkEaj0JS7pe
H70unXrRu/evXVcX6Ki0pe4tGTJEnr37k3XrqedKpJqjBolitdff52ysrKgo0icU+lL3CkrK+P111
/XXn4djB49ms8++4y33nor6CgS51T6EncKCwv57LPP+OpXvxp0lIRRee7jb3/7W8BJJN6p9CXuLFy
4kLS0NK666qqgoySMc845h0suuYSFCxcGHUXinEp4s7ChQvJzc2lffv2QUdJKGPGjGHFihV8+umn
QUeROKbs17iyf/+9+Vq9ezZgxY4KOknDGjBlDeXm5Lt2UL6XS17iyaNEi3F21Xw/Dhg2jTZs2OsQjX
0qlL3F14cKFZGZmMnjw4KCjJJyMjAxGjx7NwoULca/L6OeSS1T6EjfcnyULF3LVVVfRrFlUw0JJFW
PGjGHbtm188MEHQUeROKXS17hRXFzMrl27GDt2bNBRElblfzsd4pGaqPQlblQWlY7n11+vXr3o3bs
3r7zyStBRJE6p9CVuvPTSSwwYMEBDLzTQNdcdw+Lfizly5EjQUSQOqfQlLuzfv5/ly5czYcKEoKMK
vPHjx3Ps2DEWLVoUdBSJQ1GVvpmNNbNNZrbzZ06u5vXLzewtMyszs5uqvHaLmX0Q/nNLRiJLclmwY
AH15eWMHhZ+6CgJ7/LLl+fMM89k/vyoZjWVFFNr6ZtZOvAoMA7oD0w2s/5VVvsI+AbwVJVtzwJ+Cg
wBcoGfmpplus5TTvPjii5x77rnk5OQEHSXhNW/enHHjxvHSSy9RX14edByJM9Hs6ecCm919q7ufAOY
CX/gd3N23uft6oKLKtmOAV939gLSfBF4FdGmGfMHx48d55ZVXu0666zTdx4xMmDCBvXv3UlhYGHQU
iTPR/IRlATsinpeEl0Ujqm3NbJqZFZlZ0d69e6P80pIsli5dyuHDh3VoJ4bGjh1Ls2bNdIhHTNN6
Vc3QWm0t/tFta27z3b3HHfP6dixY5RfWpLF/Pnzad26taZGjKHMzEyuOIKl6cJprSLwG6RTzvCu
yK8us3ZFtJAE70/Pnzufrrq2nVqlXQcZLK+PHj2bhxo+70ls+IprvXA33MLNvMMoBJQLS7DwuBq82
sfffgE7tXhZSIARf69mpKSEl2q2Qgq/5u+8MILASeReFJR6bt7GTCdUFlvBJ5192Izm2lm4wHmbLCZ
lQD/ADxuZsXhbQ8A/07oH47VwMzwMhEAnnmGZO3b871118fdJSk06NHD4YMGcIzzzwTdBsjIXZvo
/H15OR4UVFR0DGKcVRUVNC9e3cuueQSHXtuJA899BA/+MEP2LRpE3379g06jjQiMlvj7rVe86zr4y
QwBQUF7Ny5k0mTJgUdJWn94z/+I2amvX05RaUvgZk7dy4tW7bkuuuuCzpK0srKyuKyyy5j7ty5QUe
ROKHS10CUlZUxb948rr32Wtq2bRt0nKQ2adIkNmzYwLvvhvht0FIkDKn0JxOuvv86ePXuYOHfi0FGS
3o033khaWpr29gVQ6UtA5s6dS5s2bbjmmmuCjpl0OnXqxOjRo5k7d66mURSVvjS9I0e08Mwzz3Djj
TfSunXro00khC1TprBlyxaWLVsWdBQJmEpfmtY8efM4dOgQU6dODTpKyrjpppto27Ytc+bMCTqKBE
ylL0luzpw5903bl8suuyzoKCnjDPO4Otf/zrz5s3j008/DTqOBEilL03qvffeY/ny5UydOhWz6sb
jk8YydepUjh49ytNPPx10FamQs1+a1BNPPEF6ejr/9E//FHSULJOTk8PAGQNliCfFqfSlyZw8eZIn
n3ySa6+9ls6dOwcdJ+WYGVOnTqWoqIj169cHHUcCotKXJvPnP/+ZPXv2cOuttwYdJWVNmTKFFila8
NhjjwUdRQKi0pcm8+CDD3Leeecxby4oKOKrLPPpu8vDyefPJJ9u/fH3QcCYBKX5rEihUrWLLyJd
///vdJT08P0k5Ku/POOzl69CizZ8800ooEQKUvTeKhx4iMzOTb3zjG0FHSXkDBgzg6quv5pFHHuH
EiRNBx5EmFlXpm91YM9tkZpvN7O5qXm9hZs+EXY80s57h5T3N7KiZrQv/+Uls40si2LZtG88//zzT
pk2jTZs2QccRQnv7u3fvlpDLKajW0jezdOBRYBzQH5hsZv2rrDYVO0juvYGHgAciXtvi7oPCf26LU
W5JIA8//DBpaWl85zvfCTqKhI0ZM4b+/fvz0EMPATyefBPNnn4usNndt7r7CWAUUhVC0wnAk+HHzw
FXmu68EeDjjz/mN7/5DRMnTqRr165Bx5EwM+Ouu+5i7dq1/PWvfw06jjShaEo/C9gR8bwkvKzadcJ
z6n4KnB1+LdvMlprZ62Y2soF5JcHcf//9nDhxghkzZgQdRaq4+eabOe+887jvvvuokgIOo40kWhK
v7o99qq/D9a0zm6gu7tfdPwAeMrM2p32BmbTzKzIzIr27t0bRSRJBDt27OCxxx7jlltu0fyscah58
+b87Gc/Y926dbzwwgtBx5EmEk3plwDdIp53BXbVtI6ZNQPOBA64+3F33w/g7muALcBpP/3uPtvdC9
w9p2PHjnX/FBKXZs2ahbtz3333BR1FajB58mTOP/98ZsyYQXl5edBxpAlEU/qrgT5mlm1mGcAkYH6
VdeYDt4Qf3wQsdnc3s47hE8GYWS+gd7A1NtElnm3dupU5c+bw7W9/m549ewYdr2qQnp7OzJkz2bhx
I/n5+UHHkSZQa+mHj9FPBxYCG4Fn3b3YzGaa2fjwanOAs81sM6HDOJWXdv4OrDeztwmd4L3N3Q/E+
kNI/LnzzjvJyMjgxz/+cdBRpBY33HADl156Kffccw+HDh0KOo40Mou3y7VycnK8qKgo6BjSAC+99B
Ljx4/ngQce4Ic//GHQcSQKhYWFDBS2jDvvvJP/+q//Cjq0lIOZrXH3nFrXU+lLLJWWlnLBBRfQunV
rlq5dS0ZGRtCRJEr//M//zJw5cli7di0DBw4MOo7UUbSlr2EYJKZ+/vOfs23bNh599FEVfoL5z//8
TzIz7m99ttlCWcSU+lLzKxevZr777+fKVOm8JWvfcXoOFJHZ599Nr/85S9Zvvn5Dz/8cNBxpJHo8
I7ExKFDh7j44os5efIk69ato3379kFHknppwd772ta+xYMECVq5cycUXXxx0JImSDu9Ik5o+fTofv
gh+fn5KvWEZmb87ne/o0OHDkyePJkjR44EHULiTKUvDfY//M//P73v+e+++7jssuCzqONFCHDh3
4wx/+wPvvv8/tt9+uAdmSjEpFGmTx4sV8+9vf5sorr+QnP/1J0HEkRkaPHs3PfvYzfv/73/Mf//Ef
QceRGQwDABJXMXFdxwww3069eP559/nmbN902UTO677z62bNnCjBkz6NmzJzfffHPQkSQG9FMq9
bJ582bGjRtH69atefnEdz3fAAAKNElEQVTl1znzDODjIqXZmb89re/paSkhKlTp5KZmcl1110XdC
xpIB3ekTp79913GTlyJKWlpbz88st079496EjSSDIyMnj++ecZNGgQN9xwA08//XTQkaSBVPpSJyt
WrOCKK64gLS2NN954g0GDBgUdSRpZzMymixYtYsSIEEt15fHrX/866EjSACp9iYq78/DDD3PFFVeQ
mZnJ8uXL6d+/6qyZkqzatWvHggULUOaaa7jjjjv45je/SWlpadCxpB5U+lKrvXv3MnHiRL73ve8xd
uxYioqKyM7ODjqWNLFWrVrx4osvMmPGDJ588kmGDh3K+vXrg44ldaTSlxpVVFTw29/+ln79+vGnP/
2J+++nz//+c+6+SqFpaen82//9m8sWLCAjz/+mEsuuYS77rpLQzInEJW+nKaiooIXXniBnJwcpk2
bxsCBA3n77bf50Y9+RFqavmUExowZw3vvvcfUqVN58MEH6du3Lw8++KDu4E0A+gmWU/bv388jjzC
wIEDufHGGz18+DB//OMfWbp0qY7fy2nOoussHn/8cVasWEH//v2566676NmzJz/+8Y95//33g44nN
Yiq9M1srJltMrPNZnZ3Na+3MLNnwq8Xm1nPInfuCS/fZGZjYhddYmH37t3MmTOH66+/ni5duvDd73

6Xl1l1b8tRTT7Fx40by8vIwq27ee5GQoUOH8tprrr/Hmm28ybNgwHnjgAfr168fw4cN54IEHKC4u11A
OcaTWUTbDc9y+D3yV0AToq4HJ7r4hYp3bgQvd/TYzmrR8zd0nml1/4GkgF+gCLAL6unuNMzBrlM3G
c+DAAYqLi3n33XdZtWoVBQUFp/bIunXrxg033MA3v/1NLrroooCTSiLbtWsXf/jDH5g7dy7rlq0Do
HPnzgwbNoyhQ4dy4YUXMmDAALKysrRDEUMxmznLzIYBP3P3MeHn9wC4+88j11kYXmeFmTUDPgY6Ep
4rt3LdyPVqej+V/pdzd8rKyjh+/DhHjx6ltLSUI0eOcoJQIQ4dOsTBgwc5cOAA+/btY/fu3ezvZv
t27fz4YcfcuDA59MTd+jQgeHDhzNixAjGjRvHgAED9AMoMbdz504WLFjAG2+8QUFBAVu2bDn1WuvW
rcnOzqZnz5506dKFzp07061TJ9q3b0/79ulp164dbdu2pW3btrRq1YpWrVrRsmVLMjIy9L1ajWhLP
5phGLKAHRHPS4AhNa3j7mVm9ilwdnj5yirbZkXxnnV24MABRo4cGfOvW9dfS2ta391P/YlcVlFRcW
p55eOKigoqKiooLy+nrKyM8vJyTp48SVlZGSdPnow6U2ZmJp07d6ZHjx4MHjyYXrl6ccEFF3DBBRf
QvXt3/eBio8vKyuLWW2/11ltvBULnjSp/2/zggw/48MMP2bZtG0VfRezZsyf7+3mzZvTrFkzmjdv
Tnp6Os2aNSM9PZ309HTS0tIws1N/V/0DnPa4qvr8bMTi5+nCCy9s9Lueoyn96j5J1f8zNa0TzbaY2
TRgGlDvW/rT09Mb7WRjXf9n1rR+1W88oNpv0PT0dMzsC9/Ild/kGRkZZGRk0KJFC1qlakXrlq1p3b
rlqT2is846i/bt23P22WfTsmXLbnlukVg7++yzufzyy7n88stPe62srIz9+/ef+m218rfXw4cPn/q
t9vjx45w4cYLjx4+f2gkqLy8/tYNUubNUdUcqcocer6uOq6nP+IVbnLJri/pdoSr8E6BbxvCuwq4Z1
SsKHd84EDkS5Le4+G5gNocM70YaPdOaZzJv3r3z6bCoicaBZs2Z06tSJTp06BR0lqUVz9c5qoI+ZZ
ZtZBJAJmF9lnfnALeHHNWGLPFRP33xgUvjqnmygD7AqNtFFRKSuat3TDx+jnw4sBNKBj9y92MxmAk
XuPh+YA/zBzDYT2sOfFN622MyeBTYAZcAdX3bljoiINC5NjC4ikgQ0MbqIiJxGpS8ikkJU+iIiKUS
lLyKSQlT6IiIpJO6u3jGzvcD2BnyJDsC+GMUJmJ5LfnJniU/J9Fmg7p+nh7t3rG2luCv9hjKzomgu
W0oE+izxSZ8lPiXTZ4HG+zw6vCMikkJU+iIiKSQZS3920AFiSJ8lPumzxKdk+izQSJ8n6Y7pi4hIz
ZJxTl9ERGqQlKVvZt8JT8RebGa/CDpPLJjZv5qZmlmHoLPUL5n90szeM7P1ZvYnM8sM0lNdmdnY8P
fWZjO7O+g89WVm3cxsiZltdP+cfc/oTA1lZulmttbM/hJ0loYws0wzey78s7IxPGVtzCRd6ZvZKGA
CoYnaLwD+T8CRGsZMuhGamP6joLM00KvAAHe/EHgfucfghPHViZunAo8A4oD8w2cwaZ7q2xlcG3OXu
5wNDgTsS+LNU+h6wMegQMfAr4BV3/zvgImL8mZKu9IF/Ae539+MA7r4n4Dyx8BDwQ6qZajKRuPvf3
L0s/HQloZnUEkkusNndt7r7CWAuoR2MhOPuu939rfdJQ4SKpVHmr24KZtYV+Hvgd0FnaQgzawdcTm
iOEtz9hLv/byzfIx1LvY8w0swKzex1MxscdKCGMLPxwE53fzvoLDH2LWBB0CHqKAVYefG8hAQuypk
mlhO4GCgMNMkd/DehHaOKoIM0UC9gL/D/woeqfmdmZ8TyDaKZIzfumNkioHML1L91L6DO1J/Qr62Dg
WTPR5XF8mVitn+fHwNVNm6j+vuyzuPuL4XXuJXR4Ib8ps8VAdTPex+33VTtMrA3wPPB9d/8s6Dz1Y
WbXANvcfY2ZfSXoPA3UDLgE+I67F5rZr4C7gfti+QYJx92vqulM/sX4IVWya8yswpCY1jsbap8dV
XT5zGzgUA28LaZQehwyFtmLuvuHzdhxKh92f8bADO7BbgWuDKe/yGuQQnQLeJ5V2BXQFkazMyaEyr
8fHd/Ieg8DTACGG9mlwAtgXZm9kd3nxJwrvooAUrcvfk3rucIlX7MJOPhnt8DowHMrC+QQYIOwuTu
77j7Oe7e0917EvqGuCreC782ZjYW+BEW3t1Lg85TD6uBPmaWbWYzhOaCnh9wpnqx0F7EHGCjuz8Yd
J6GcPd73L1r+GdkErA4QQuf8M/2DjPrF150JaE5xmMmIff0a/EE8ISZvQucAG5JwD3KZPV/gRbAq+
HfXFA6+23BRoqeu5eZ2XRgIZAOPOHuxQHHqq8RwM3A02a2Lrzsx+7+coCZJOQ7QH54x2Ir8M1YfnH
dkSsikkKS8fCOiIjUQKUvIpJCVpoiIilEpS8ikkJU+iIiKUSlLyKSQlT6IiIpRKUvEgUzu83M1oX/
fGhms4LOJFIfuJlLpA7C49UsBn7h7i8FnUekrrSnLlI3vyI0tosKXxJSMo69I9IozOwbQA9gesBRR
OpNh3dEomBmlwJPAiPd/WDQeUTqS4d3RKIZHTgLWBI+mZvQ0/JJ6tKevohICtGevohIClHpi4ikeJ
W+iEgKUemLiKQqlb6ISApr6YuIpBCVvohIClHpi4ikkP8P0SzbXhCk2LIAAAAASUVORK5CYII=\n
\\",\n",

```
"      \"text/plain\": [\n",  
"      \"<matplotlib.figure.Figure at 0x7f4b1873c588>\"\n",  
"      ]\n",  
"      },\n",  
"      \"metadata\": {},\n",  
"      \"output_type\": \"display_data\"\n",  
"    },\n",  
"  ],\n",  
"  \"source\": [\n",  
"    \"# Chop an array to include only x values between minx and  
maxx\n",  
"    \"def datachop(x, y, minx, maxx):\n",  
"    \"    bool = np.logical_and(x <= maxx, x >= minx)\n",  
"    \"    return x[bool], y[bool]\n",  
"    \"\n",  
"    \"# Set the min and max\n",  
"    \"minx = -1\n",  
"    \"maxx = 1\n",  
"    \"
```

```

"    \n# Get the new cut x and y arrays\n\n",\n",\n",
"    \n\"cutx, cuty = datachop(x, y, minx, maxx)\n\n",\n",\n",
"    \n\n\n",\n",\n",
"    \n# Calculate the area under the cut arrays\n\n",\n",\n",
"    \n\"area = integral(cutx, cuty)\n\n",\n",\n",
"    \n\"print(\\\"\\\"Integral from\\\"\\\", minx, \\\"\\\"to\\\"\\\", maxx, \\\"\\\"is:\\\"\\\",
area)\n\n",\n",\n",
"    \n\n\n",\n",\n",
"    \n# Plot the cut arrays in red against the original in
black\n\n",\n",\n",
"    \n\"plt.plot(x, y, color=\\\"\\\"black\\\"\\\")\n\n",\n",\n",
"    \n\"plt.plot(cutx, cuty, marker=\\\"\\\"o\\\"\\\",
color=\\\"\\\"red\\\"\\\")\n\n",\n",\n",
"    \n\"plt.xlabel(\\\"\\\"z\\\"\\\")\n\n",\n",\n",
"    \n\"plt.show()\n\n",\n",
"    ]\n",\n",
"    },\n",\n",
"    {\n",\n",
"        \n\"cell_type\\\": \n\"markdown\\\", \n",\n",
"        \n\"metadata\\\": {}, \n",\n",
"        \n\"source\\\": [\n",\n",
"            \n\"Now, we should put everything together to make it easier and in
one function.\n\n",\n",
"        ],\n",\n",
"        },\n",\n",
"        {\n",\n",
"            \n\"cell_type\\\": \n\"code\\\", \n",\n",
"            \n\"execution_count\\\": 11, \n",\n",
"            \n\"metadata\\\": {}, \n",\n",
"            \n\"outputs\\\": [\n",\n",
"                {\n",\n",
"                    \n\"name\\\": \n\"stdout\\\", \n",\n",
"                    \n\"output_type\\\": \n\"stream\\\", \n",\n",
"                    \n\"text\\\": [\n",\n",
"                        \n\"0.656883274351\n\n",\n",\n",
"                    ],\n",
"                },\n",
"            ],\n",
"            \n\"source\\\": [\n",\n",
"                \n\"def integral_range(x, y, minx, maxx):\n\n",\n",
"                \n\"    cutx, cuty = datachop(x, y, minx, maxx)\n\n",\n",
"                \n\"    area = np.trapz(cuty, x = cutx)\n\n",\n",
"                \n\"    return area\n\n",\n",
"                \n\n",\n",
"                \n# Test it\n\n",\n",
"                \n\"print(integral_range(x, y, minx,maxx))\n\n",\n",
"            ],\n",
"        },\n",
"        {\n",\n",
"            \n\"cell_type\\\": \n\"markdown\\\", \n",\n",
"            \n\"metadata\\\": {}, \n",\n",
"            \n\"source\\\": [\n",\n",
"                \n\"Because the mean is 0 and the standard deviation is 1, the $x$
axis is actually $z$ values from page 67 in the Harris textbook. Take a look
at Figure 4-3 and compare it with the plot above. The values you calculate
here should be very close to twice the value of $z=1$ from table 4-1. The

```

reason it is twice is that the \$z\$ value in the table is from \$z=0\$ to \$z=1\$ and we are going from \$z=-1\$ to \$z=1\$.

```
"    \"\\n\",\\n\",
"    \"**Problem 1:** Using the code above, calculate the area under our
Gaussian curve from the range of -2 to +2 standard deviations. This should be
near 95%. \"\\n\",
```

```
"    ]\\n\",
"    },\\n\",
"    {\\n\",
"        \"cell_type\": \"code\",\\n\",
"        \"execution_count\": 75,\\n\",
"        \"metadata\": {},\\n\",
"        \"outputs\": [],\\n\",
"        \"source\": [\\n\",
"            \"# Your code here!\"\\n\",
"        ]\\n\",
"    },\\n\",
"    {\\n\",
"        \"cell_type\": \"markdown\",\\n\",
"        \"metadata\": {},\\n\",
"        \"source\": [\\n\",
"            \"Now, let's apply what you've learned to a simulated chromatogram.
```

This code will generate two peaks with different retention times and widths.

```
"    ]\\n\",
"    },\\n\",
"    {\\n\",
"        \"cell_type\": \"code\",\\n\",
"        \"execution_count\": 12,\\n\",
"        \"metadata\": {},\\n\",
"        \"outputs\": [\\n\",
"            {\\n\",
"                \"data\": {\\n\",
"                    \"image/png\":
```

```
\\\"iVBORw0KGgoAAAANSUheUgAAAYUAAAEKCAyAAAD9xUlFAAAABHNCSVQICAgIfAhkiAAAAAwSF1
zAAALEgAACxIB0t1+/AAAIABJREFUeJzt3Xd4mX18PHvsbwdO8vOIMuBDagjQMyGEii7JZTSULIK
LYXSDaUL6ALat4PSwq80tKzSUErDKiPQ0DACpAECCVlkx9lOHI848ZYtS+f943kkhOMhWZJlxedzX
b6icet5TrS07i2qijHGGAOQluwAjDHG9B2WFIwxoRYUjDGBBNiScEYY0yIJQVjjDEhlhSMMcaEWF
IwxhgTYknBGGNMIcUFY4wxIenJDiBahYWFwLxcnOwwjDEmpXzwwQfVqlrUXbmUSwrFxcUsXbo02WE
YY0xKEZHtkZS5iNjjDEhlhSMMcaEWF IwxhgTkrCkICKPiEiliKzuosx0EVkhImtE5K1ExWKMSYy
iawpzAYu6OxOERke/AWYoapHAp9PYCzGGGMikLCKoKoLgZouinwReFZVd7jlKxMVizHGmMgks09hE
jBYRN4UkQ9E5OokxmKMMYbkJoV0YBrwKeB840ciMqmjgiJyvYgsFZG1VVVVvRmjMRGr8/p4bnlZss
MwJibJTAPLwH9VtVFVq4GFwNSOCqrqg6paoqolRUXdTsgzJileWLGbm55cyfa9jckOxZgeS2ZSeAE
4Q0TSRSQXOAlYl8R4jIlJ+f5mAHbv9yY5EmN6LmHLXIjIHGA6UCgiZcBtQAaAqt6vqutE5L/AKiAA
PKyqnQ5fNaavq6hrcf+lpGBSV8KSgqrOjKDMXcBdiYrBmN5UWe8kgz2WFEwKsxnNxsRJsIawp9aSg
klldhSMiZNgMrDmI5PKLCkYEWfNrX7qvG2ANR+Z1GZJwZg4CPYnZHiESrfd2ZhUZEnBmDgINh0dMb
KAijovgYAmOSJjesaGjFxFUHV1A6OGT2QtoCyt7E1yREZ0zOWFIyJg0q3H+GY0YMA62w2qcuSgjF
xUFHnJTsjjUnd8wEblmpSlyUFY+JgT10LwwuyGVGQ7V63pGBSkyUFY+Kgos7L8IJsCgdkkibWfGRS
lyUFY+Kg0k0K6Z40ivKzrPnIpCxCsBEQU1jK0NyMwAoyM6g3p3I11PPPfccIsL69evjEZ4xEbOkY
EwceNsC5GQ660vmZqXT5PPHdLw5c+Zw+umn88QTT8QjPGMiZknBmBj5A0prW4CcDA8AuRkemlt7X1
NoaGjg7bff5m9/+5slBdPrLCkYeyOvWyIznA+TrmZHppael5TeP7557nggguyNGkSQ4YMYdmyZXG
J05hIJCwpiMgjIlIpIl1unCmiJ4iIX0Q+16hYjEmkZjcp5GR6Qv82x5AU5syZwVXXAHAFVdcwZw5
c2IP0pgIJWyTHWA2MAv4R2cFRMQD3AnMT2AcxiTURzUft/kohprC3r17WbBgAatXr0ZE8Pv9iAi//
/3vEZG4xWxMZxJWU1DVhUBNN8W+C/wbqExUHMYkWjAphPoUMtNp7GGfwjPPMPVV1/N9u3b2bZtGz
t37mT8+PEsWrQobvEa05Wk9SmIyCjgUuD+ZMVgTDw0twaAj9cUetp8NGfOHC699NKP3XbZZZfxr3/
9K7YgjYlQIpuPuvN/wM2q6u+uWiwi1wPXA4wd07YXQjMmct629jUFD23uiKTM9Oh+d7355psH3HbD
DTfEHKMxkUpmUigBnnATQiFwkYi0qerz7Quq6oPAgwAlJSW2UL3pU4K1gpzMNPff9NDt0SYFY5Ita
```

UlBVccHL4vIbOC1jhKCMX1dcPRRVvpHNQWAJl8bA8lIWlZg9ETCkoKIzAGmA4UiUgbcBs4nRFWtH8
EcNLzthqQGk0JjS2yzmolJhoQlBVWdGUXZryQqDmMSraPRR0BMcxWMSRZr8DQmRqE+hYx2zUcxLHV
hTLJYUjAmRs2+jw9JzQn1KVhNwaQeSwrGxOijjuaPlj4Caz4yqcmSgjExavH5yc5Ily3NmW+T5/Yp
xLIonjHJYknBmBg1+/yh/gQIaz6yPgWTgiwpGBOj5lZ/qd8BwjuaraZgUo8lBWNi5A3bYAcgO92Sg
kldlhSMiVH7mkJampAT4+5rxiSLJQVjYuRlO5rD5WXFtvuaMcliScGYGDX7/KHO5aCcGLfkNCZZLC
kYEyNvu9FHALkZ6Tb6yKQkSwrGxKjZ9/E+BbCagkldlhSMiZG39cCkEMvua8YkkyUFY2LUfvIaOCu
lWk3BpCJLCsbEyOsLHNDRnJvpCa2JZEwqsaRgTAxUlelTaLftZm6mh8YW62g2qSdhSUFEHhGRShFZ
3cn9V4rIKvfVHRGZmqhYjEmUljZ32ewOhqRan4JJRYmsKcwGLuji/q3Amap6DPar4MEExmJMQRtFY
CcoN9NDk8+PqiYjLGN6LGFJQVUXAjVd3P+Oqu5zry4GRicqFmMSxdvWWVJIxx9QWv2BZIRlTI/1lT
6Fa4GXO7tTRK4XkaUisrSqqqoXwzKma8GaQkdDUsPvNyZVJD0piMhZOEnh5s7KqOqDqlqiqiVFRUW
9F5wx3QiOMoosKTRaUjApJj2ZJxeRY4CHgQtVdW8yYzGmJ7zu/swHrn3kflRspVSTapJWUxCRscCz
wFWqujFZcRgTC2+wptBuSGqwjyGYNIxJFQmrKYjIHGA6UCgiZcBtQAaAqt4P/AIYCvxFRADaVLuKU
fEYkwih0UftagpZbpIIDlk1JlUkLCmo6sxu7r8OuC5R5zemNwT7FNqPPvooKlVfgkktSe9oNiaVeT
vpaM5yr1tNwaQaSwrGxKCZpJDpcWsKl1qdgUowlBWNi8NGQlI9/1LIyrPnIpCZLCSbEIDQktDM+Bas
pmNRiScGYGHh9fjxpQrqnxU0h3foUTGqypGBMDLy+wAFzFCCs+cj2VDApxpKCMTHwth24FSdY85FJ
XZYUjImB19dxUgiNPrKkYFKMJQVjYtDSFgg1FYUTEbLS02z0kUk5lhSMiUGLz092+oE1BXCakGyeg
kk1lhSMiYHXFzhgjkJQVobHmo9MyrGkYEwMOuTAKz5yKQkSwrGxKCz0UcQTApWUzCpxZKCMTHosv
ko3WN9CiblwFIwJgZenz80e7m9rAxpPjKpJ2FJQUQeEZFKEVndyf0iIveKSKmIrBKR4xMVizGJ0nV
NIY1Waz4yKSARNYXZwAVd3H8hMNH9ux74awJjMSYhWrqqKaTb6COTehKWFFR1IVDTRZFLgH+oYzEw
SERGJioeYxKhpslgHc3moJLMPoVRwM6w62XubcakBH9AafV33nyUaUNSTQqKKCMiYDgROce9nCMi+
XE4t3Rwm3Zy/utFZKmiLK2qqorDqY2JXfALv/Oago0+Mqmn26QgIl8DngEecG8aDTwfh3OXAWPCro
8GdndUUFUFVNUSVS0pKiQKw6mNiVlw520ls6G4OgjSwomtURSU/g2cBpQB6Cqm4BhcTj3XOBqdxT
SyUCTqpbH4bjG9IrO9mcOshnNjHwLr1CmRVVbRzZWHhFJp5NmnnAiMgeYDhSKSBlwG5ABoKr3A/OA
i4BSOAm4pgfxG5M03ScFG31kUk8kSeEtEfKjKCMi5wLfAl7s7kGqOrOb+xWnFmJMSgo2H2V1lnzkz
lNQVYI/qozp6yJpProFqAI+BL6O8wv/Z4kMyphU4O2uoZnDNtoXqSeSmkIO8IiqPgQgIh73tqZEBm
ZMXxdsPupokx0gNKmtq7kMxvQ1kdQUXsdJAKe5wGuJCceY1BEcbtpVRzNgnc0mpUSSFLJVtSF4xb2
cm7iQjEkNoXkKXey8BthcBZNSIkkKjEGL1YnINKA5cSEZkxpC8xs62HkNrE/BpJZI+hS+BzwtIsGJ
ZSOBLyQuJGNSQyTzFMCaj0xq6TYpqOoSETkcmIyzNMV6VfULPDJj+rjIk4LVFEZqiKSmAHACUOyWP
05EUNV/JCwqYlKAt62b5qPg6CPrUzAppNukICKPAYcBK4BgPVgBSwqmXwsNSeli5zWw5iOTWiKpKZ
QAU9wZyMYyl9cXIMMjeNI6nq1szUcmFUUy+mg1MCLRgRiTarf+6fDUCGSgklNkdQUCoG1IvI+0BK
8UVVnJCwqYlJAS5s/NOy0Ix/1KVjzkUkdkSSF2xMdhDGpyOvrFNclSjQCSU2RDEl9qzcCMSbVtLT5
ulzTKHztI2NSRSQ7r50sIktEpEFEWkXELyJlVRGcMX1ZtzUFG31kUlAkHc2zgJnAJpzF8K5zb+uWi
FwgIhtEpFREbung/rEi8oaILBeRVsJyUTTBG5NM3XU0Z3ps7SOTeiJJCqhQKeBRVb+q/hlnR7UuuU
ts3wdcCEwBZorIlHbFfgY8parHAVcAf4kidmOSyuvruvkoLU3I9KTR6rekYFJHJB3NTSKSCawQkd8
D5UBeBi87EShV1S0AIvIEcAmwNqyMAGXu5YHAbOXJEV5fgCF5Xf+uykpPs5qCSSmR1BSucst9B2gE
xgCXRFc4UcDOsOt17m3hbge+507hPA/4bgTHNaZP8HYzJBWcfGxRuzCpPNukoKrbgQDO2kfPare4z
Und6WiaZ/tZ0TOB2ao6GrgIeExEDohJRK4XkaUisrSqqiqCUxuTeC2+QKf7MwdlpXts9JFJKZGMPv
oUsBm4F6eDuVRELozg2GU4tYqg0RzYPHQ2t8BSAqr4LZONMlvsYVX1QVUtUtaSoqCiCUxuTeN31KYD
bfGRJwaSQSjQp/gicparTVfVM4CznggetwSYKCLj3T6JK4C57crsAD4JICJH4CQFqwgYlNDD6COA
zPQ0m9FsUkokSaGyXXPRFqCyuwepahtOP8R8YB3OKKM1IvJLEQkukfED4GsishKYA3zFFt4zqcLb1
vU8BxB2X2u2pGBSSKejj0Tks+7FNSIyD6eZR4HP49QCuwQq83A6kMNv+0XY5bXAaVHGbeZs+fwB/A
Elp5vmo5wMG31kUktXQ1IvDrTCAZzpXq4CBicsImNSQPDXf05md0nBQ3VDA2+ElNI8Hg9HH300qor
H42HWRfMceuqpyQ6rX+o0KajqNe4EtBtUNZI+BGP6jdAGO93UFLIzPKGypnM5OTmsWLECGPnz53Pr
rbfy1lu27FoydNkgqqp+wJbINqYdb6vTJNR985H1KUSrrq6OwYotMSJZIpnr/I6IzAKexJm8BoCqL
ktYVMb0cV53Qlp3Hc3ZmVZTiERzcZPHHnssXq+X8vJyFixYkOyQ+q1IkKwYe+XYbcpChb8wzEmNT
S3un0KEdQUvNbR3K3w5qN3332Xq6++mtWrVyPS8VanJmEi2U/hrN4IxJhUEupo7rZPIYl1mnx9VtS+
4CJlyyilUV1dTVVXFfSGHDKh1OvxPJjOaBInJ3cJkJEfmjiAzsjeCM6asi7WjOyfdgDyg+v02/idT6
9evx+/0MHT02aH0S5E0Hz0CrAYud69fBfwd+GynjzDmIOeNuKbg3O9t85PZzTpJ/VmwTFAVXn00
UfXeLp+bk1iRJIUDlPV8FVR7xCRFYkKyJhUEOwn6LajOZgUWv0UZGckPK5U5fdbz3xfEclPl2YROT
14RUOA5oTF5Ixfv80k9fCyxvt10VSU/gm8KjbJyBADfCVRAZlTF8X8egjN2nYCCSTKiIZfbQCmCo
iBe71uoRHZUwf99E8he5HH4HVFEZqiGT00Y1luQqgH7haRZSJyXuJDM6bv8ro1he422QkmjWDNwpi+
LpI+ha+6tYPzgGHANcdvEhqVMX1ccNns7uYe5ISNPjImFUSSFILv+ouAv6vqSjreatOYfq051d9tf
wKE9SlyTcGkiEiSwgci8gpOUpgvIvk4ezYb0281+yJLCSgd2axPwaSKSJLctcAtwAmq2gRk4jQhdU
tELhCRDSJSKiK3dFLmchFZKyJrRORfEUduTBJfsj8z2Ogjk3oiGX0UEJFi4EsiosAiVX2uu8e5ezH
cB5wLlAFLRGsuu9tasMxE4FbgNFXdJyK20ilJCZEmhWybp2BSTCSjj/4CfAP4EGe5i6+LyH0RHPtE
oFRVt6hqK/AECem7Ml8D71PVfQCq2u3ez8b0BV5f9/szw0dDum35bJMqIpm8diZwlKoqgIg8ipMgu
jMK2B12vQw4qV2ZSe4x3WY8wO2q+t8IjmlMUjX7/N3OZgbI9KSRJpYUTOqIpE9hAZa27PoYFFUEj+

tohFL7pSLTgYnAdGAm8LCIDDRgQCLXBldpraqqiuDUxiRWpKOPRMTZfc1GH5kU0WlSEJEXRWQuMBR
YJyJvisibwFqgKIj1l+EkkKDRwO4Oyryggj5V3YqTgCa2P5CqPqiqJapaUlQUyamNSSxvm7/bzbOD
sm1LTpNCumo++kMhtwlvOs6v+u4sASaKyHhgF3AF8MV2ZZ53jzVbRAPxmpO2RHBsY5LKG2FNAZykY
KOPTKroNCmo6lvByyJyLM4X+uXAVuD+7g6sqm0i8h1gPk5/wSOqkZEFgksVdW57n3nichawA/8SF
X3xvIfmQY3BGc0RyLH9mk2KaTTpCAik3B+3c8E9gJPAhLN9pyqOg+Y1+62X4RdVuD77p8xKSPSPgV
wlrqw5iOTKrpqPloP/A+4WFLAUTkpl6Jypg+TFUjntEMzrBUqymYVNFV/fcyYA/whog8JCKfxNY8
MoaWNqd/wDqazcGo06Sgqs+p6heAw4E3gZuA4SLyV1s62/Rnke7PHGRDUk0q6banTFUbVfVxVf00z
rDSFThrIRnTL320P3M0o48sKbTX0ubnukeXcP9bm5MdigkTyYzmEFwTAR5w/4zplz7anznC0Uc2JL
VDv523ntfWVfLaukoOGZTDjKmHJDskQ2Qzmo0xYSLdnzkoJ9P6FNpbtKma2e9s4+pTxnFC8WB+9PR
Kqubpbk2WwZKCMVEL7qIwaUdzVkaaJYV2n1lWxuDCDH76qSO4Y8ZrTLQFWLC+ItlhGSwpGBM1b7Q1
hQwPrW0BAoH2S3/1T4GAsnBTNWdMLCIR3cMRI/MZNSiHV9faIsl9gSUFY6LU3IPRR2D7NAetLa+ju
qGFMyc565iJCJ88YhiLSqusQ74PsKRgtJSiHX0UXGLbhqU63trorHR8xqTC0G3nHDEcry/A26XVyQ
rLuCwpGB0laGsKwX2avW02Ag1g4cYqjjykGh52aHbTjp0CAOy0lmw3pqQks2SgjFRCjZxRLogXrb
VFEJa2wIs27GP0ycUfuz2rHQPR48ayJrddUmKzARZUjAmSqGkEMHOaxDWp2Dt5Wzf24jPrxwxsuCA
+yaPyGdTRb1lyCeZJQVjohT1PAW3nA1LhY0VDQBMGDbggPsmDc+nsdXPrv3NvR2WCWNJwZgoNfn8p
KcJGZ7IPj55WU5SaGhps2RYKWFTZT0iHSeFySoc2zZW1Pd2WCZMQpOCiFwgIhtEpFRE0l0vSUQ+Jy
IqIiWJjMeYeGhsaWNAdUQrxOS7ZRu8lhQ2VTYwdkhuhyO3Jg7PB2CDJYWkSlhSEBEPcB9wITAFmCk
iUzoolw/cALyXqFiMiacGbxSdsiJPCnlwWaspQG1fAXm7qCUAFGRncMjAbDbusaSQTImSKZwIlKrQ
FlVtBZ4ALumg3K+A3wPeBMziTNzUt0SXFIJlG/t5UmjzB9hS3cCEYfmdlpk0Ip8Nbr+DSY5EJoVRw
M6w62XubSEichwWrlVfSmAcxsRVg7ct1CQUibxMp2x9P28+217ThM+vndYUACYPz2dzZQNTfpvTkS
yJTAod7dIWGmsmImnAPcAPuj2QyPUis1REllZVVcUxRGOi1xB1TSEtTcjL9PT75qNNbg1g4vDOK8L
E4fm0+gNs29vUW2GZdhKZFMqAMWHXRwO7w67nA0cBb4rINuBkYG5Hnc2q+qCqlqhQSVFRUQJDNqZ7
DS1tDMjOiOoxA7LT+33zUWml0ldwWFHnSeHQojzAmc9gkiORSWEJMFFExotIJnAFMDd4p6rWqmghq
harajGwGJihqksTGJMxMauPsmZnH6F+n6eFDZVNjBqUE6o470jxU0dpGA1heRJWFJQ1TbgO8B8YB
3wlKquEZFFisiMRJ3XmERraPFF1acATlLo70NSN1U0dn10BDA4N4P87HSrKSRRdO/sKKnqPGBeu9t
+0UnZ6YmMxZh48PkDeH2B6GsK2en9uk/BH1A2VzVw2oShXZYTEYqH5rG12pJCstiMZMoieOwX6Enz
UX/uUyjb10RLW4CJXQXHDRO3NJft1nyUNJYUjIlCcFhpNDOawZnAlp+HpAZHHk3opvkInH6Fsn1Nt
NpS40lhScGYKASbgPKjrCnkZ/Xv5qNNle5w1C7mKAQVF+YRUGxhvCSxpGBMFijf7NHWFIJDULX757
LQmyrrGTkwM/wIhviWD80FYJt1NieFJQVjohAcQRRtn0JeVjptAaWlnzaJlFY2dLgyakfGucNstlt
nc1JYUjAmCg097GgOnjflx36FQEAprWyIqJMzoHBAJnmZHPurkCSWFIyJQizNR9A/F8WrqPfs1OoP
zVbujoGwbmiezVVEKsKxkShx81Hmf13+ezg8NLgbOVIFBfasNRksaRgTBSCS1UEv+QjFawp9Mfmo
+Av/nFuB3Ikxg3NY+e+JlStNQsKRgTheAG02lpHS0C3Ln8LgfUTX+sKWzb20SGRzhkUE7Ejykemo
vPr5TX2jYrvc2SgjFRaGjxRd10BP27T2H73kbGDMnFE0UiHRdaGM/6FXqbJQVjotAQ5f7MQXlZzp7
E/XG1lG3VTYwbEnnTEcd4Q1stNVksKRgThZ4smw1hzUf9rE9BVdlR0xt65R+pYflZZGek2VyFJLcK
YEwUGlqi24ozKDsJDU+a9Lvmo72NrTS0tIVmKUCquFqqNR/1PksKxkShoYc1BZH+uSVnaORRYXQ1B
XBGK1nzUe+zpGBMFKLdnzlcfnZGvxuSuq3a+VKPtK8BnHkNO/Y24Q/0z/WikiWhSUFELhCRDSJSKi
K3dHD/90VkrYisEphXRWRcIuMxJlYN3p51NEP/3FNh+95G0grGD44+KYwbmkerP8CeOhuW2psSlhR
ExAPcBlwITAFmisiUdsWWAyWqegzWDPD7RMVjTKwCAaWhT53qZbOD8rL6X/PR2vi6xhfmkZke/VdN
caGTSLSLZUNCQ7LNOFRNYUTgRKVXWLqRYCTwCXhBdQ1TdUndhouBgYncB4jIlJbbMPVSjI6X75544My
s2kprElz1H1XarKip21TB0zqEePnzZcWUBvw576eIZlupHIpDAK2B12vcy9rTPXAi93dIeIXC8iS0
VkaVVVVRxDNCZyO/c5v1/G9KB9HGB4QTaV9f2nKaS81kt1QwtTR/cSKQwdkEVRfhbrLSn0qkQmhY6
mL3bYYyQiXwJKgLS6ul9VH1TVElUtKSoqimOIxkRuZ42ze9jYHieFLKobWvH1k/V8VpXtB+hxTQHg
8BH5VlPoZYlMCMxAMLDro4Hd7QuJyDnAT4EZqtqSwHiMicmOmthrCgCV9f3jbb5iZy0ZHuGIkZHto
9CRycPz2VhRbyOQelEik8ISYKKIjBeRTOAKYG54ARE5DngAJyFUJjAWY2K2o6aJIXmZPR6S0sJNCh
X9ZDTNqrL9HDGygKx0T4+PcfjIAlraAjaJrRclLCmoahvHWA+sA54SlXXiMgvRWsgw+wuYADwtIi
seJG5nRzOmKQR29fEmMGRr/TZ3rCCLAAq+sHKn4GA8mFZLceMHhjTcQ4fYZ3NvalnP3kipKrzgHnt
bvtF2OVzEnl+Y+JpR00TR4/q+Zdcf6opbKlupL6lrcedZEETHg0gTWB9eR0XHT0yTtGZrtiMzMmi4
A8ou/Y197itGWBwbiYZHqGiH/QprNwZeyczQHaGh/GFeayzmKvsaRgTATKa5tpC2iPO5kB0tKEYf
nZ/aL5aFXZfnIzPRxWNCdmY00dM4gPteu8jYJ3NvcKSgJERiHU4atDwgiwq+sFchRVltRw9amBUG+t
05oyJhdQ0trK2vC4OkZnuWFIwJgI7g8NRe7CGT7jhBdnsOchrCq1tAdbtrou56SjotAmFAPxvU3Vc
jme6ZknBmAjs3NeEJ00YOSg7puMML8imsu7g71lNYv6eOVn8g5k7moGH52Rw+Ip//bbLVDHqDJQVjI
rBtbxOjBuWQ4YntIzO8IJv6lraDerXU1WW1ADEPRw33iU1FLN22j+ZWf9yOaTpmScGYCYrbqS4Bx
vFtDc8OFFhIB6WumLHfobmZTI6hjdk7Z0+oZBWf4D3tu6N2zFNxywpGNMNVWvbdSpjo9xSsiPBuQo
H8x4BS7bVUFI8GJHYO5mDSooH40ktlm7bF7djm05ZUjCmG3sbW6lvaYtLTSF4jE0VB+ceAeW1zeyo
aeLE8UPjetzczHSOoQSAJdtq4npccyBLCsZ0Y1uls+50PJLCyIHZFA7IZJXb7n6weX+r86V90vgHC
T92SfEQVuzcT2tb/1h1NlksKRjTjaluUhG/NPakICiCpWogH+7aH/Ox+qL3t9YwICudI0YwXp3YJe
MG09IWYPXugzOh9hWWFIzpxtbqRtLTJG4dp0ePhkRpZcNBOQLp/a01TBs3OC6T1tqbVjwYgKXWhJR

QlhSM6ca2vY2MGZJLeozDUYOmjh5IQDnoZuhWlHnZVNnAiQloOgJnvkLx0FxeWlXO26XVtuxFglhS
MKYbW6ubKI7DyKOG4EqrBlu/wv+9tpH0NEnoagaXHjealbtqufLh9/jF3NWoWmKIN0sKxnRBVdm+N
z5zFIKGFQWQzoia7tF3lWWDN7lqeWLKTL59azPg4Plft3XjORFbdfj7XnT6efy7ewV/e3BzR47ZWN1
Lb7EtYXAeThCYFEblARDaISKmI3NLB/Vki8qR7/3siUpzIeCJRWlKf9a+Peq+P55aXUdv00ZtuW3U
j5bXN8Q7PxIGq8r0nlnPnf9d3W/Zf7++gqdXPpOE931KyIyeOH8JLq8r59X/W0tIW3lm6Pn+AtrB9
oFWVdeVlCWluWbO7lp8+9yGX3/8ug3IyuOHsiXE/R3sDstL5yUVHcMmxh/DHVzawYmfXyXV/UysX/
3kRNz6xvMtytc2+UOKUEp7VAAAXqElEQVROafP3uM9nS1UDv5m3joYU7TNKWFIEQ9wH3AhMAWYKS
JT2hW7FtinghOAe4A7ExVPV4JT519cuZtz7l7IX9/6+K8PVeXh/22htLLjseW/emktNz25kpN/+zq
PvbuN2iYfl/31Ha6dvTSUYOq8Pu5+ZQO3Pvshzy4rA6Cy3hvaUaq1LUBpZUPUe9F+sL2G1bu6b4ZQ
VVbs3I/X17MvoNW7anl9XQU1ja09enx7myrqP5ZEw+3Y28TMBxfzfwfZ9qCoLN1Zx8Z8XcfkD73a66
X00iXzB+kqeX7Gbv7+9lTpvxzGoKrPf3spPnlvNWZOL+OzxoyI+fiR+dclRXF4yhof+tt5U/v17aZd
mGlraPfc1359pHl3LpX94Jvdb/eHc7F/7pf9y7YFOozKJN1dz9yoYOH7+vsZVrZy/hn4u3d3mem59
ZxafuXcQzH5Rx4dEjefLrpzAwNyPiOGORlib8v88cxbD8bG5+ZhWPLNrKw//b0uH7YPY722hoaePN
DVWhIbMd+fbjy5h+1xs8u6yM8+9ZyF1/eJMtVdHNJ2lu9fPlxz7gwYVb+MZjH6Tk8FlJVJuciJwC3
K6q57vXbwVQld+GlZnvlnlXRNKBPUCRdhFUSUmJLl26NC4xqiqzFpRyz2sbsubxkDK+sdb70CrLT+d
+Pzw69wUsrGzjn7rcYPTiHud85nSF5maFjrn5VY8WzFnHpsaOoamjh7dJqTj50K09sdqbjP3btiRw
7ZhBXP/I+K3fuZ0BWOOnXeNm6/eAoPLtxCdUMrf7nyeO5dsIlVZbXkZ6Uzekgu44bkcxnozjnioF4
0oSmljay0z2khY3qaGxp49TfLSAzPY0FPziT/OyPfyCrGlq48+X15GdnsLa8lsVbajh8RD6zvng8E
4Z1vM59Zb2X+9/cwgc79vHQ1dMYlp/N5qoGZvx5EY2tfjI8wl+vnMY5U4YDzq+ify8r462NVdx64R
GhFS278sKKXXz/qZWMHZLLE9efHNRQPui389bxwMit5GV6OGruQN7bWkNRfhZV9S386PzJfPusCQD
sqfXy+/+u59V1FTS1+vnNpUfxhRPG8u7mvSwqrSKg8PVPPhMqgXOf1Wldex+aqBu5+dSO1TT72Nrby
/z5zFF86eRzvbdnLzf9exc8/PYUzJhZx29zVzH1/J+dOGc6sLx4X0z7DXblhznLmr9nDn644lntfL
+W6M8bz2eNHA85EsHtFL+XppTs5ZFAON507kRlTRx0wskdVaWhpY0BWOst270Oyv74LwDWNFXPM6I
H88OlVpKcJaSK88cPpDM7L4Ow/vMWu/c0s/NFZjA3rL6ms9/Klh99jY0UDmelp/PfGMzi0gz0R3lh
fyTWz13D1KeP4wbmTey0ZtPfa2gqu+8dH3wc3X3A435x+GABzV+5GgJ+/sJqjRw1kw556BuZkMHJQ
DlNHD+TGT06kuqGVnAwPaWlw7C9fRVUJKBQOyETVST6Hj8gnw5PGJyYW8pnjRoXeT+2pKj9+ZhXPL
CvjqpPH8Y93t3POEcP40xXHkZeVzrbqRm58YjmnTijkyppGMjgKFXf3N7Wya38zBdkZPd7TQ0Q+UN
WSbsslMC18DrhAVa9zrl8FnKSq3wkrs9otU+Ze3+yW6XSN3HglhabWNn709Cr+82E5U8cMYuXO/WR
60vjj5VO54YnlXHDkCE4aP4RLjxvN8yt2cdvcNWR4hBOKh/CPr57Iwk1V3P3qRirqWggElDd+NB2P
CDNmLWJzVSMzTzxLgvUVDC/IDtUCzn3xeM46vIgvPLCYFTv3k5+VTmf+VmjI403nTmJPrZfyWi+ry
vZTWd/C9MlFfPPMw7j+sQ84b8pw7vr8VDZW1FOQncF/V5dz+4trAbj29PH8/NMfVcS8Pj8zh1rMml
1lpKVBToaHL508jsff24E/oLz03dMpys/i9XWVvL05mvGFeeRmpvObeeto9vkr4FPHjOTXlx7N5/7
6DpX1Lfzh88dwz6ub2FrdyHfPnsB/1+xh+Y79pAkMzMkgoPD8t09jQFY6/1ldzu5aL6ceNpSaxlbK
9jWT6Ulj+c59vLx6D8eMHkRpRT15WemMGZLLieOH8K3ph5Gbmc5pvlvAmCE51DS2sq/Jx3fPnsAXT
xrL959ayStr9nBY0QAq61uoaWwlmZ2NS48dxZLtnQQCygnXlXdhnxasJoLifLj/dMVxdM3L5NN/Xk
SL+8vtbl18u4a75G8jwpHH7jClc8/cl1HnbyMnwMGN4AFaW1fLtsw7jB+dO/lgiJreyfU2c/ce3aG0
LhL7sf3jeZHbtb+KppWWoKpcdP5pVZbWsLa9j8vB8Pnv8KI4ZPYiTDx3Ck0t2cseLa2n2+TnykAIK
sjNYW17HuVOG88wHTo308BH53H35sXzmvr5c3IRU0cP5A+vbATgJxcdzvWfcl5E2/wBvvjQe3y4q
5bfXXY0P3t+NaMH51I4IBovz8+QvEyG5GUyODEtuSt3k5Wexss3foLM9OR2Tb63ZS8jBmbzh1c28t
Kq3dz52WNA4MfPrAqV+fc3T6W0sp6b//0h44bmsn1vE2OH5LJzXxMnFA/ha2ccytf+sZS/XHk8Gyv
q+XzJGBpb2vjxM6tQoK7Zx9bqRgqy0znp0KES2VZD4YAszphYyPfOmCTAnAzumr+e+97YzAlnT+D7
503msXe3cdvcNRw+ooAnvn4yff5i/gcfff24Gqkp6WxoxjD2FTZQNjBucw64vHo6qhRBTkDyg/f2E1/
3pvBwDfOPMwbrnw8B49T30hKXweOL9dUjhRVb8bVmaNWYy8KZyoqnvbHet64HqAsWPHTtu+vetqbW
fqvT7e3FDFhj31vLaugo0V9dx64RFcd8Z4Fm+pwecP8IlJRfz4mZU8tdT5QF13+ni21zSxfk8dN35
yEj98eiWXHjeK19ZWMGRAJuOG5nHLSWM5/8gRAGyuamD229v44fmTmfP+Dn738nqG5Wfx+88dw/TJ
wwDnF+4vXlJn1z5xKIcMyuEHT63g6lOKPzZqo80f4J+Lt3PHS2tRhUxPGq3+AHfMOJLfzFtHdoaHr
PQ0xgzJzdLwfJ5csoOjRglkzOBcBudl8E7pXrbubXR+1R/hnDfdk8a26kYunrWIUYNy8PkDbK5qJC
fDQ7Pb1Hdi+CHcedkxPL98F396fROFA7KoaWzhka+cwPTJw9i9v5kZsxZR3dKpOEDuOz40XzmuFF
4fx5mxHr7Y515aQLtW8OK8rP41NEjueXCwlmzu5ZC0ppbPhz/janNVCVU4u5a/4G/jzom47cjiaq
zpaMAHsbWvjB0ytJTxOGF2QzcmA2lxw7ijFDcnlhxS5ufGIF44bmUl3fwsIfn0V5rZcbn1jOjpomR
gzMprHFz0NXTyM9LY2pYwbx97e3coebVIvys3jwqml8d85yqhtauOtzU7146ie9ep9F62+LtvJOaT
W3zziS78xZ7vxASU9jxtRDuPGTEExkzJJdAQJm3upw/vbaJTW4z5qGFeWypbuSUQ4dy8qFDefh/W6h
vaeOGsyfwzekTeOTtrRx5SAGnHlZIZnoa971Ryl3znSajUw4dSn2Lj/S0NG67eAovrNjN3sZWxly5
m7svn8pnjx/Nk0t2cOuzHzJ5RAEDc9LZ1+ijpqmVfy2tKPD4dSdx8qHxXc4iFl6fn2sfXcLbpXtJE
zjlSEK+d85E6rw+zj7cqdkGa1RPL93JY4u3U5CdwaLSak4cP4QPy2pZcdu5ndYK1+yu5e5XNrkuvI
6TDxvK/iYfb22somhAFkMHZLJmdx0zTxzLby49KrTu04L1FVz76FK+UDKG/6wq59wpw/nB+Z0597V
NPLosjOH5Weyu9fLU10/h2WV1LN6yl39edxKjB+eyv6mVnz6/mv+sKufKk8Zy2oRcjhhZ0000/L6Q

```

"      \"text/plain\": [\\n",
"      \"      \\\"<matplotlib.figure.Figure at 0x7f4b1872c048>\\\"\\n\",

```

```

"    ]\n",
"    },\n",
"    \"metadata\": {},\n",
"    \"output_type\": \"display_data\"\n",
"  }\n",
" ],\n",
" \"source\": [\n",
"   \"# Set the retention times\\n\\n\",
"   \"t1 = 10\\n\\n\",
"   \"t2 = 15\\n\\n\",
"   \"\\n\\n\",
"   \"# Set the peak widths\\n\\n\",
"   \"w1 = 1\\n\\n\",
"   \"w2 = 1.5\\n\\n\",
"   \"\\n\\n\",
"   \"# Define the x axis\\n\\n\",
"   \"x = np.arange(0,20, 0.1)\\n\\n\",
"   \"\\n\\n\",
"   \"# Sum together two gaussians and some noise. Note, the second one
is set to have half the area of the first\\n\\n\",
"   \"y = Gaussian(x, t1, w1/4) + 0.5*Gaussian(x, t2, w2/4) +
np.random.normal(0,0.01, len(x))\\n\\n\",
"   \"\\n\\n\",
"   \"# Plot the results\\n\\n\",
"   \"plt.plot(x, y)\\n\\n\",
"   \"plt.xlabel(\\\"Retention Time\\\")\\n\\n\",
"   \"plt.ylabel(\\\"Absorbance\\\")\\n\\n\",
"   \"plt.text(t1+1, 1.4, \\\"A\\\")\\n\\n\",
"   \"plt.text(t2, 0.7, \\\"B\\\")\\n\\n\",
"   \"plt.show()\\n\\n\",
" ]\n",
" },\n",
" {\n",
"   \"cell_type\": \"markdown\",\n",
"   \"metadata\": {},\n",
"   \"source\": [\n",
"     \"**Problem 2:** Find the **a)** the area of peak A, **b)** the area
of peak B, and **c)** the ratio of the area of A the area of B. Hint: the
ratio of A/B should be around 2. \"\n\",
"   ]\n",
" },\n",
" {\n",
"   \"cell_type\": \"code\",\n",
"   \"execution_count\": 78,\n",
"   \"metadata\": {},\n",
"   \"outputs\": [],\n",
"   \"source\": [\n",
"     \"# Your code here\\n\\n\",
"     \"# Hint: you should use the integral_range function from
above\\n\\n\",
"     ]\n",
"   },\n",
"   {\n",
"     \"cell_type\": \"markdown\",\n",
"     \"metadata\": {},\n",
"     \"source\": [\n",

```

" \"""Problem 3: Assume peak A is the internal standard with a known molar absorptivity of 4,500 M⁻¹cm⁻¹ and a known concentration of 5 mM. Assume peak B is our analyte with a known molar absorptivity of 3,700 M⁻¹cm⁻¹. **a) What is the detector response factor, F? **b) What is the concentration of our analyte from the chromatogram Problem 2?\""

```
" ]\n",
" },\n",
" {\n",
"   \"cell_type\": \"code\",\n",
"   \"execution_count\": 79,\n",
"   \"metadata\": {},\n",
"   \"outputs\": [],\n",
"   \"source\": [\n",
"     \"# Your code here\"\n",
"   ]\n",
" },\n",
" ],\n",
" \"metadata\": {\n",
"   \"kernel_spec\": {\n",
"     \"display_name\": \"Python 3\",\n",
"     \"language\": \"python\",\n",
"     \"name\": \"python3\"\n",
"   },\n",
"   \"language_info\": {\n",
"     \"codemirror_mode\": {\n",
"       \"name\": \"ipython\",\n",
"       \"version\": \"3\"\n",
"     },\n",
"     \"file_extension\": \".py\",\n",
"     \"mimetype\": \"text/x-python\",\n",
"     \"name\": \"python\",\n",
"     \"nbconvert_exporter\": \"python\",\n",
"     \"pygments_lexer\": \"ipython3\",\n",
"     \"version\": \"3.6.2\"\n",
"   }\n",
" },\n",
" \"nbformat\": 4,\n",
" \"nbformat_minor\": 2\n",
"}"
]
},
{
  \"cell_type\": \"code\",
  \"execution_count\": null,
  \"metadata\": {},
  \"outputs\": [],
  \"source\": []
},
{
  \"cell_type\": \"code\",
  \"execution_count\": null,
  \"metadata\": {},
  \"outputs\": [],
  \"source\": []
},
{
```

```

    "cell_type": "code",
    "execution_count": null,
    "metadata": {},
    "outputs": [],
    "source": []
  },
  {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {},
    "outputs": [],
    "source": []
  },
  {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {},
    "outputs": [],
    "source": []
  }
],
"metadata": {
  "kernelspec": {
    "display_name": "Python 3",
    "language": "python",
    "name": "python3"
  },
  "language_info": {
    "codemirror_mode": {
      "name": "ipython",
      "version": 3
    },
    "file_extension": ".py",
    "mimetype": "text/x-python",
    "name": "python",
    "nbconvert_exporter": "python",
    "pygments_lexer": "ipython3",
    "version": "3.6.2"
  }
},
"nbformat": 4,
"nbformat_minor": 2
}

```