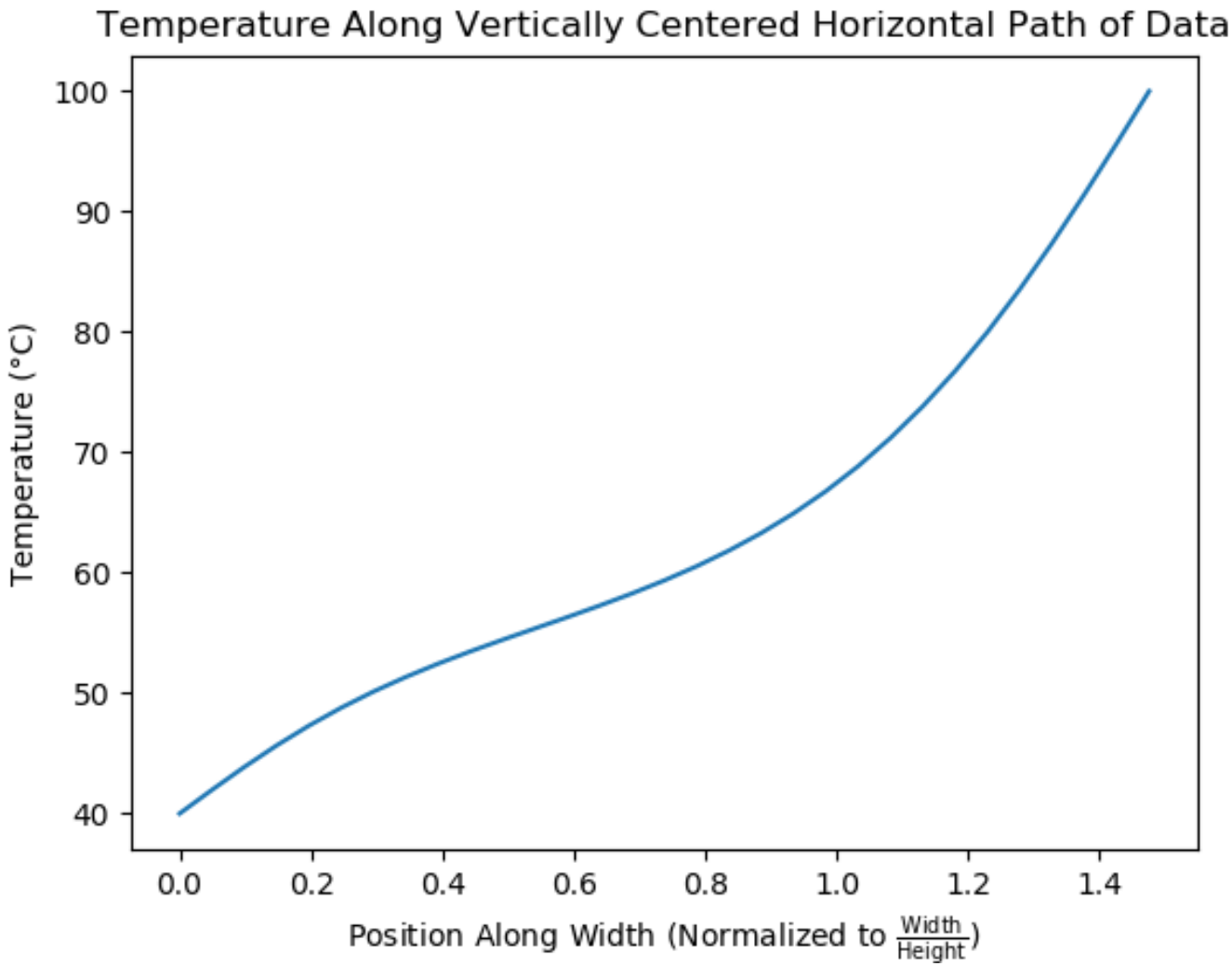
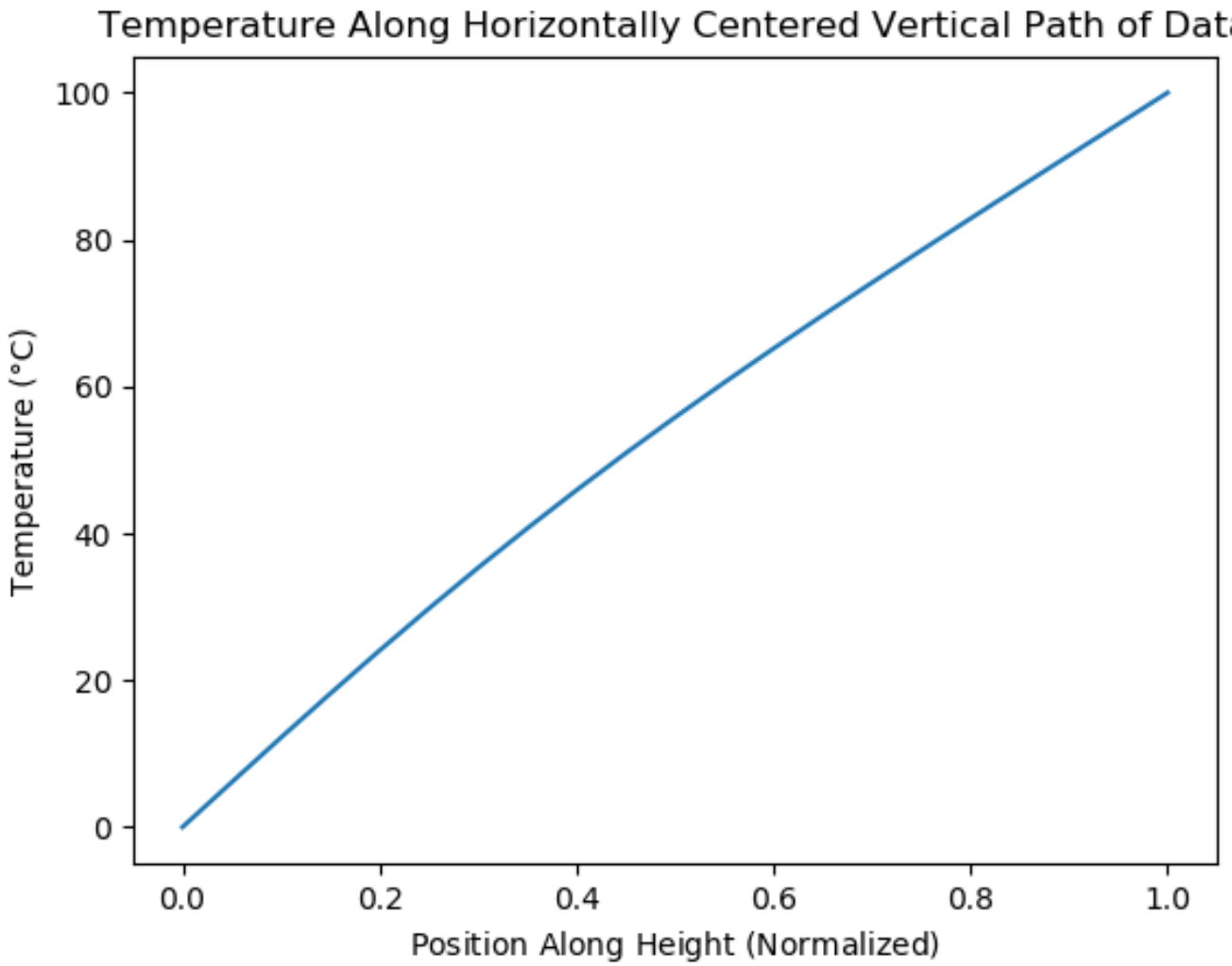
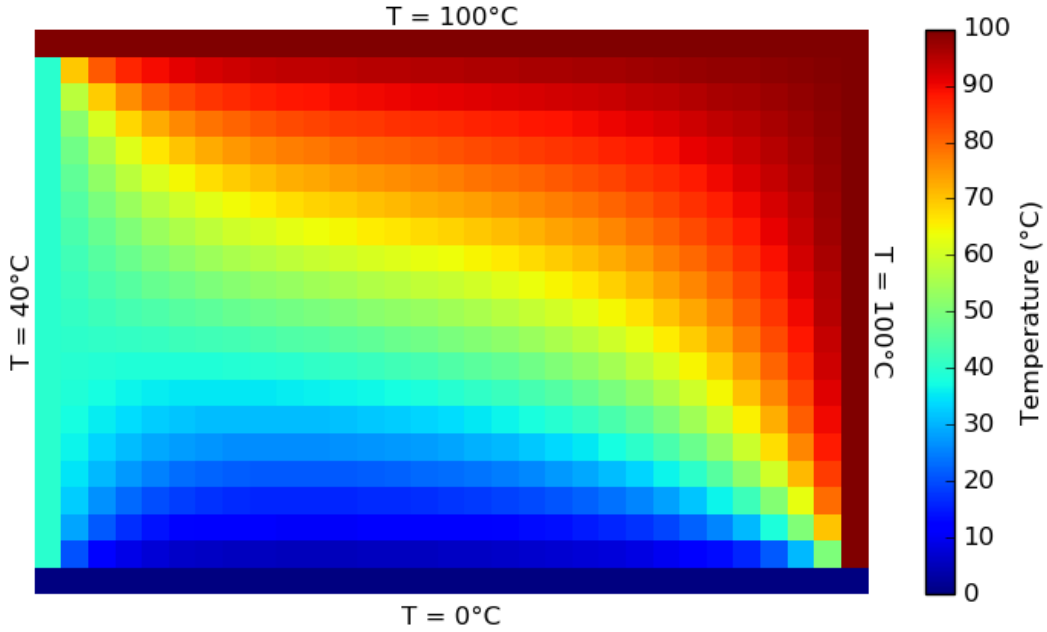

Temperature Along Vertically Centered Horizontal Path of Data
Temperature (°C)
100
90
80
70
60
50
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
Position Along Width (Normalized to Width/Height)


Temperature Along Horizontally Centered Vertical Path of Data
Temperature (°C)
100
80
60
40
20
0
0.0
0.2
0.4
0.6
0.8
1.0
Position Along Height (Normalized)


T = 100°C
T = 40°C
T = 100°C
T = 0°C
100
90
80
70
60
50
40
30
20
10
0
Temperature (°C)

1 PSet-3.py

```
import numpy as np
import matplotlib.pyplot as plt

# plt.axes().set_aspect('equal')
# plt.style.use('classic')

# Grid squares values:
height = 21
width = 31

# Make set temperatures on fixed positions
T_alpha   = 0    # (bottom boundary temperature)
T_bravo   = 40   # (left boundary temperature)
T_charlie = 100  # (top boundary temperature)
T_delta   = 100  # (right boundary temperature)

# Initialize matrix of zeros for that size
# Note: index 0,0 is bottom left
default_temp = (max(T_alpha, T_bravo, T_charlie, T_delta) + min(T_alpha, T_bravo,
    T_charlie, T_delta)) / 2
data = np.zeros((height, width)) + default_temp

# Set boundary conditions
for i in range(width):
    data[0, i] = T_alpha
    data[(height - 1), i] = T_charlie
for j in range(1, (height - 1)):
    data[j, 0] = T_bravo
    data[j, (width - 1)] = T_delta

error_flag = True
error_limit = 1e-4
while error_flag:
```

```
    large_error_term_found = False

    # Gauss-Seidel Iteration
    for n in range(1, (height - 1)):
        for m in range(1, (width - 1)):
            data_old = data[n, m]
            data[n, m] = 0.25 * (data[(n + 1), m] + data[(n - 1), m] + data[n, (m + 1)]
   + data[n, (m - 1)])

            if not large_error_term_found:
                error_term = abs(data[n, m] - data_old) / data_old
                if (error_term <= error_limit):
                    error_flag = False
                else:
                    error_flag = True
                    large_error_term_found = True

#print(data)

fig1 = plt.figure(1)
x = np.linspace(0, (width / height), width)
index = np.ceil(height / 2)
y = data[index.astype(int), :]
plt.plot(x, y)
plt.xlabel(r'$\mathrm{Position\ Along\ Width\ (Normalized\ to\ \frac{Width}{Height})}$')
plt.ylabel("Temperature (\N{DEGREE SIGN}C)")
plt.title("Temperature Along Vertically Centered Horizontal Path of Data")
plt.savefig("Problem-Set-3/images/pset-3-figure-1.png")
plt.show()

fig2 = plt.figure(2)
x = np.linspace(0, 1, height)
index = np.ceil(width / 2)
y = data[:, index.astype(int)]
plt.plot(x, y)
```

```
plt.xlabel(r'$\mathrm{Position\ Along\ Height\ (Normalized)}$')
plt.ylabel("Temperature (\N{DEGREE SIGN}C)")
plt.title("Temperature Along Horizontally Centered Vertical Path of Data")
plt.savefig("Problem-Set-3/images/pset-3-figure-2.png")
plt.show()


fig3 = plt.figure(3)
plt.axes().set_aspect('equal')
plt.style.use('classic')
heatmap = plt.pcolor(data)

plt.text(0.5, -0.02, "T = " + str(T_alpha) + "\N{DEGREE SIGN}C",
         horizontalalignment='center',
         verticalalignment='top',
         rotation=0,
         clip_on=False,
         transform=plt.gca().transAxes)
plt.text(0, 0.5, "T = " + str(T_bravo) + "\N{DEGREE SIGN}C",
         horizontalalignment='right',
         verticalalignment='center',
         rotation=90,
         clip_on=False,
         transform=plt.gca().transAxes)
plt.text(0.5, 1, "T = " + str(T_charlie) + "\N{DEGREE SIGN}C",
         horizontalalignment='center',
         verticalalignment='bottom',
         rotation=0,
         clip_on=False,
         transform=plt.gca().transAxes)
plt.text(1, 0.5, "T = " + str(T_delta) + "\N{DEGREE SIGN}C",
         horizontalalignment='left',
         verticalalignment='center',
         rotation=270,
         clip_on=False,
```

```
          transform=plt.gca().transAxes)

plt.axis("off")

plt.xlim(0, width)
plt.ylim(0, height)

cbar = plt.colorbar(heatmap)
cbar.set_label("Temperature (\N{DEGREE SIGN}C)")
plt.clim(0, 100)

plt.savefig("Problem-Set-3/images/pset-3-figure-3.png")
plt.show()
```