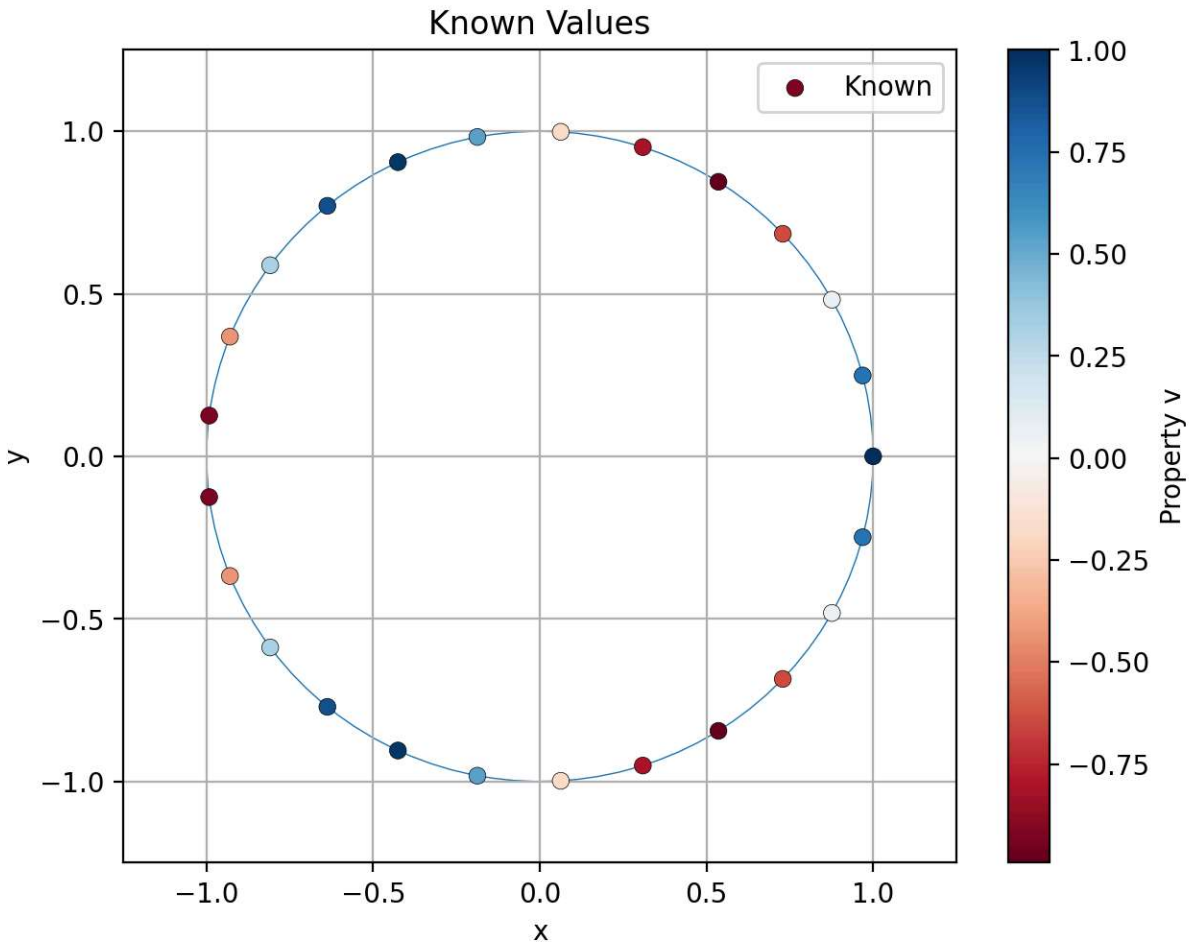
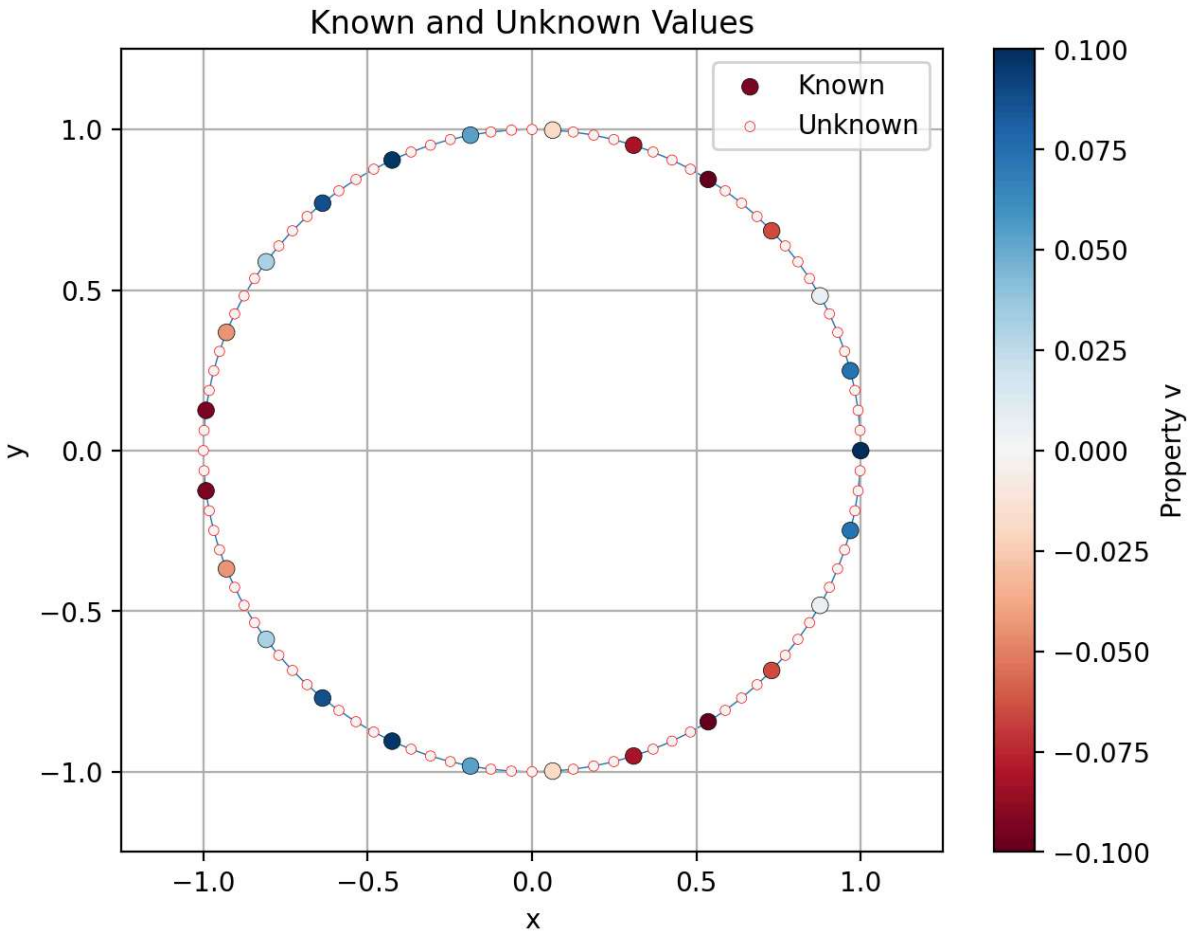
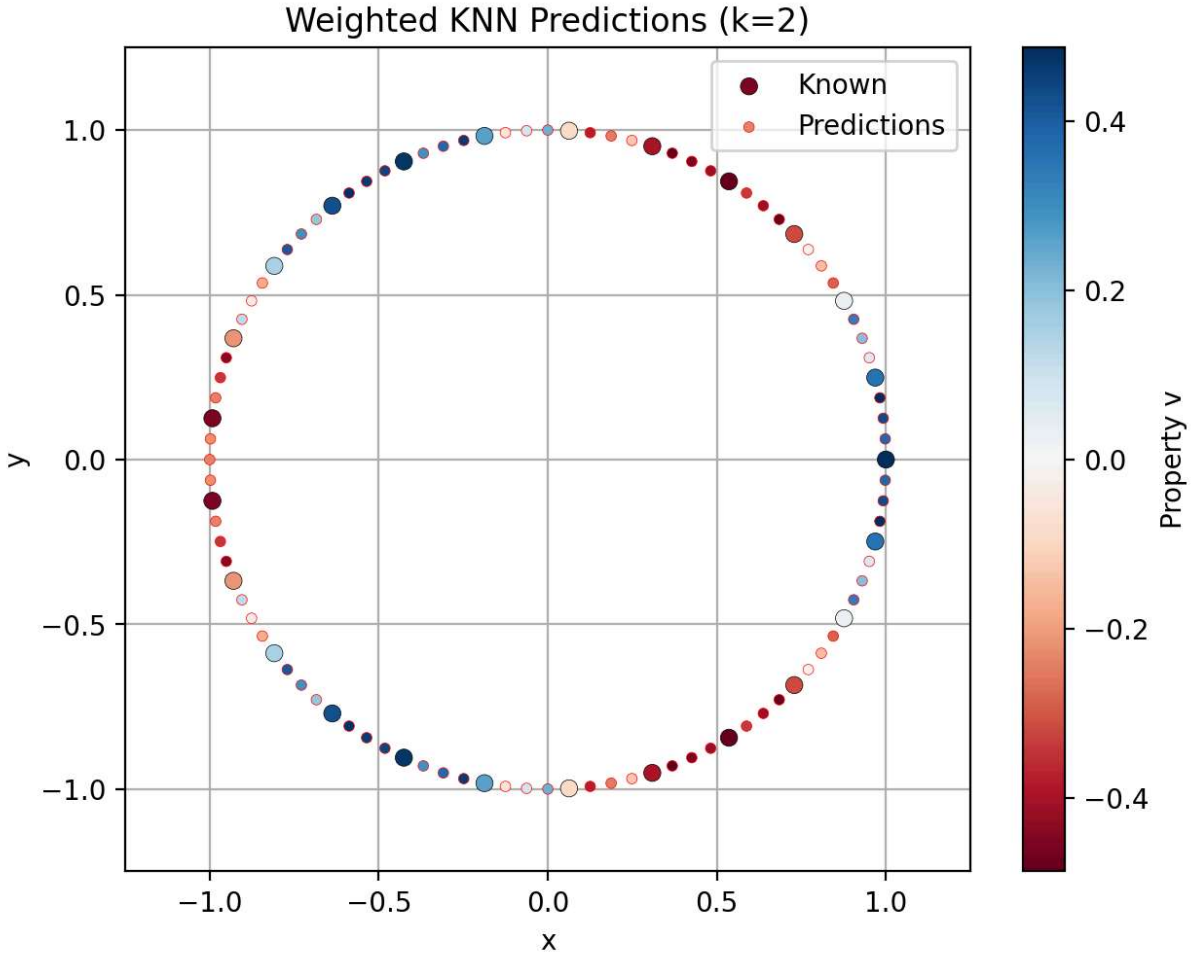
In [38]:

```
# I wrote my own code for weighted knn regression as the sklearn library
# was not very easy to work with for custom distances and weights

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# function to compute distance between two points
def arc_length(x):
    r = 1
    dist = r*(abs(np.arctan2(x[1], x[0]) - np.arctan2(y_data, x_data)))
    return dist

# function to perform regression for a point
def regress(x, k):
    d = arc_length(x)
    indices = np.argpartition(d, k)[:k]

    prox = np.square(d[indices])
    weights = prox/np.sum(prox)  # calculating custom weights

    knn = np.sum(np.dot(v_data[indices], weights))/k  # regressing over two closest
    return knn

# predicting values at unknown points
def predictions(points, k):
    l, b = np.shape(points)
    L = np.zeros(l)
    for i in range(l):
            L[i] = regress(points[i], k)  # predicting the values at each query poi
    return L

# importing data from file
path = ('hw1_bonus.xlsx')
known = pd.read_excel(path, "Known")
unknown = pd.read_excel(path, "Unknown")
x_data = known['x'].values
y_data = known['y'].values
v_data = known['v'].values
x = unknown['x'].values
y = unknown['y'].values
points = np.array([x, y]).T

k = 2
data = predictions(points, k)  # getting output from knn regression

# plotting
# plot 1: known values
plt.figure(figsize = (7, 5.5), dpi = 200)
t = np.linspace(0,np.pi*2,100)
plt.scatter(x_data, y_data, c = v_data, cmap = 'RdBu', s = 40, edgecolors = 'black'
plt.plot(np.cos(t), np.sin(t), linewidth = 0.5, zorder = 0)
plt.colorbar(label='Property v')
plt.title(f'Known Values')
```

```
plt.xlabel('x')
plt.ylabel('y')
plt.xticks(np.arange(-1, 1.5, 0.5))
plt.yticks(np.arange(-1, 1.5, 0.5))
plt.xlim([-1.25, 1.25])
plt.ylim([-1.25, 1.25])
plt.grid(zorder = 0)
plt.legend(["Known"])
plt.show()

# plot 2: known + unknown values
plt.figure(figsize = (7, 5.5), dpi = 200)
t = np.linspace(0,np.pi*2,100)
plt.scatter(x_data, y_data, c = v_data, cmap = 'RdBu', s = 40, edgecolors = 'black'
plt.scatter(points[:,0], points[:,1], c = np.zeros(len(x)), cmap = 'RdBu', facecolo
plt.plot(np.cos(t), np.sin(t), linewidth = 0.5, zorder = 0)
plt.colorbar(label='Property v')
plt.title(f'Known and Unknown Values')
plt.xlabel('x')
plt.ylabel('y')
plt.xticks(np.arange(-1, 1.5, 0.5))
plt.yticks(np.arange(-1, 1.5, 0.5))
plt.xlim([-1.25, 1.25])
plt.ylim([-1.25, 1.25])
plt.grid(zorder = 0)
plt.legend(["Known", "Unknown"])
plt.show()

# plot 3: predicted values
plt.figure(figsize = (7, 5.5), dpi = 200)
plt.scatter(x_data, y_data, c = v_data, cmap = 'RdBu', s = 40, edgecolors = 'black'
plt.scatter(points[:,0], points[:,1], c = data, cmap = 'RdBu', s = 15, edgecolors =
plt.colorbar(label='Property v')
plt.title(f'Weighted KNN Predictions (k={k})')
plt.xlabel('x')
plt.ylabel('y')
plt.xticks(np.arange(-1, 1.5, 0.5))
plt.yticks(np.arange(-1, 1.5, 0.5))
plt.xlim([-1.25, 1.25])
plt.ylim([-1.25, 1.25])
plt.grid(zorder = 0)
plt.legend(["Known", "Predictions"])
plt.show()

# exporting data to file
d_pred = pd.DataFrame(np.c_[x, y, data])
d_pred.columns = ['x', 'y', 'prediction']
d_pred.to_csv("data_export.csv")
```


Known Values
Known
1.0
0.5
0.0
−0.5
−1.0
y
−1.0
−0.5
0.0
0.5
1.0
x
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
Property v


Known and Unknown Values
Known
Unknown
1.0
0.5
0.0
−0.5
−1.0
y
−1.0
−0.5
0.0
0.5
1.0
x
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
Property v


Weighted KNN Predictions (k=2)
Known
Predictions
1.0
0.5
0.0
−0.5
−1.0
y
−1.0
−0.5
0.0
0.5
1.0
x
0.4
0.2
0.0
−0.2
−0.4
Property v