```
#%% md
Import relevant packages here.
#%%
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
#%% md
Load the data and verify it is loaded correctly.
<ul>
    <li>Print it (head, tail, or specific rows, choose a
sensible number of rows).</li>
    <li>Compare it to the source file.</li>
<ul>
#%%
# Path to your CSV file
imp = '/Users/huibvanderveen/Desktop/Pycharm Projects/TU/
assignment 5/cf_data.csv'

# Load the data
data = pd.read_csv(imp)

# Verify the data is loaded correctly
# Print the first 5 rows
print("First 5 rows of the data:")
print(df.head())

# Print the last 5 rows
print("Last 5 rows of the data:")
print(df.tail())
#%% md
In the ensuing, you will use <code>numpy</code>.

Let's create a grid for the values to plot. But first create <
b>two arrays named <code>dv</code> and <code>s</code></b>
using <code>numpy.linspace</code> that hold the grid values at
 the relevant indices in their respective dimension of the
grid.

Create a <b>grid named <code>a</code></b> with zeros using <
code>numpy.zeros</code> in to which calculated acceleration
values can be stored.<br>
<br>
Let the grid span:<br>
<ul>
    <li>Speed difference <code>dv</code> [m/s]
        <ul>
            <li>From -10 till 10</li>
            <li>With 41 evenly spaced values</li>
        </ul>
    </li>
    <li>Headway <code>s</code> [m]
        <ul>
            <li>From 0 till 200</li>
            <li>With 21 evenly spaced values</li>
        </ul>
    </li>
</ul>
#%%
# Create the grid arrays using numpy.linspace
dv = np.linspace(-10, 10, 41)  # Speed difference from -10 to
10 with 41 values
s = np.linspace(0, 200, 21)    # Headway from 0 to 200 with 21
 values

# Create the grid to store calculated acceleration values
using numpy.zeros
a = np.zeros((len(s), len(dv)))  # Grid of zeros with shape
based on dv and s
#%% md
Create from the imported data 3 separate <code>numpy</code>
arrays for each column <code>dv</code>, <code>s</code> and <
code>a</code>. (We do this for speed reasons later.)
<ul>
    <li>Make sure to name them differently from the arrays
that belong to the grid as above.</li>
    <li>You can access the data of each column in a <code>
DataFrame</code> using <code>data.xxx</code> where <code>xxx</
code> is the column name (not as a string).</li>
    <li>Use the method <code>to_numpy()</code> to convert a
column to a <code>numpy</code> array.</li>
</ul>
#%%
# Extract the columns from the dataframe and convert them to
numpy arrays
DV = (data.dv).to_numpy()
S = (data.s).to_numpy()
A = (data.a).to_numpy()

#%% md
Create an algorithm that calculates all the acceleration
values and stores them in the grid. The algorithm is described
 visually in the last part of the lecture. At each grid point
, it calculates a weighted mean of all measurements. The
weights are given by an exponential function, based on the '
distance' between the grid point, and the measurement values
of <code>dv</code> and <code>s</code>. To get you started, how
```

```
 many <code>for</code>-loops do you need?<br>
<br>
For this you will need <code>math</code>.<br>
Use an <i>upsilon</i> of 1.5m/s and a <i>sigma</i> of 30m.<br>
<br>
<b>Warning:</b> This calculation may take some time. So:
<ul>
    <li>Print a line for each iteration of the outer-most <
code>for</code>-loop that shows you the progress.</li>
    <li>Test you code by running it only on the first 50
measurements of the data.</li>
</ul>
#%%
# Constants
upsilon=1.5 #[m/s]
sigma = 30 # [ m ]

for k in range(len(s)):
    for j in range(len(dv)):

        # Initialize the sums
        sum_weight_A = 0
        sum_weight = 0

        for i in range(len(data)):
            weight = np.exp((-abs(DV[i] - dv[j]) / upsilon
) - (abs(S[i] - s[k]) / sigma))
            sum_weight_A += (weight * A[i])
            sum_weight += weight

        A_us = sum_weight_A / sum_weight
        a[k][j] = A_us

print(a)
#%% md
The following code will plot the data for you. Does it make
sense when considering:
<ul>
    <li>Negative (slower than leader) and positive (faster
than leader) speed differences?</li>
    <li>Small and large headways?</li>
</ul>
#%%
X, Y = np.meshgrid(dv, s)
axs = plt.axes()
p = axs.pcolor(X, Y, a, shading='nearest')
axs.set_title('Acceleration [m/s/s]')
axs.set_xlabel('Speed difference [m/s]')
axs.set_ylabel('Headway [m]')
axs.figure.colorbar(p);
axs.figure.set_size_inches(10, 7)
#%%
```