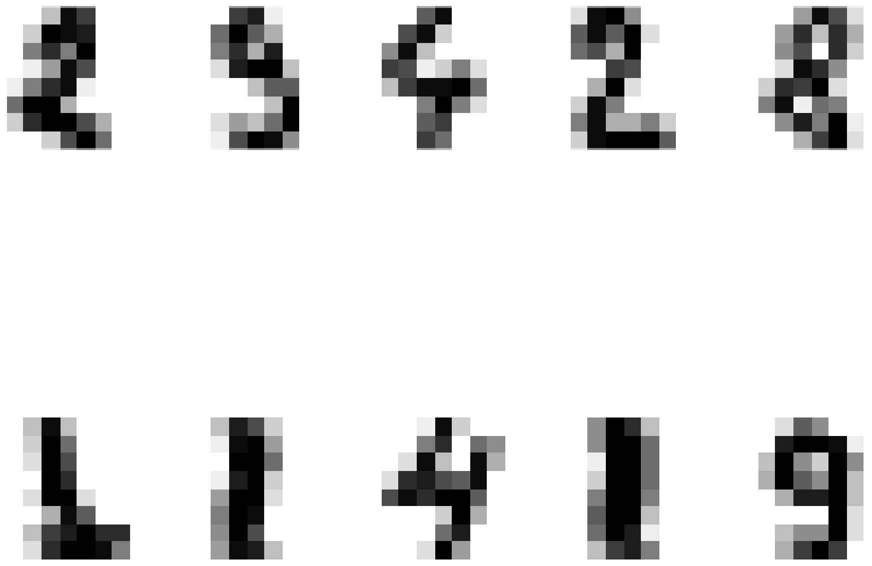
# Computer Lab: K-means Clustering on Digits

The data we're looking at represents an extraction of handwritten digits, where the data was preprocessed to an $8 \times 8$ grid of pixels.

Here's a sample of 10 random points in the set, each visualized in the $8 \times 8$ grid:



When we load the digits (using either `handdigits.mat` file for Matlab/Octave or the built-in digits file in Python), we'll have a matrix that is $1797 \times 64$, and this represents the 1797 data points in $\mathbb{R}^{64}$. The "targets" vector is something we'll use later, but it tells us which digit the image is supposed to be.

Your goal is to cluster the points into 10 clusters using k-means and show the results. Be sure to run k-means several times to get the best clustering.

Here's a partial Matlab file to get you started:

```
load handdigits

%
%  Fill in the missing command, but be sure your output for the cluster centers
%  is the 10 x 64 matrix C for the image commands below to work.
%
```

```
% Show the 10 clusters as 8 x 8 images (You don't need to edit this)
for j=1:10
    subplot(2,5,j)
    temp=reshape(-C(j,:),8,8);
    imagesc(temp');
    colormap(gray); axis equal; axis off
end
```

In Python, you might use:

```
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()  # for plot styling
import numpy as np
from sklearn.cluster import KMeans

from sklearn.datasets import load_digits
digits = load_digits()
digits.data.shape

kmeans =                                       # Instantiation of KMeans
clusters =                                     # Run k-means on digits.data
kmeans.cluster_centers_.shape

# This part you shouldn't need to change- This shows the 10 cluster centers
# as 8x8 blocks

fig, ax = plt.subplots(2, 5, figsize=(8, 3))
centers = kmeans.cluster_centers_.reshape(10, 8, 8)
for axi, center in zip(ax.flat, centers):
    axi.set(xticks=[], yticks=[])
    axi.imshow(center, interpolation='nearest', cmap=plt.cm.binary)
```