```
from PIL import Image

# Mount Google Drive first
from google.colab import drive
drive.mount('/content/drive')

# Open the image
img = Image.open('/content/drive/My Drive/TSU-AI-SUMMERCAMP-2025/ai-camp/sample_image1.png')
img.show() # it might not work on colab
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remoun
```

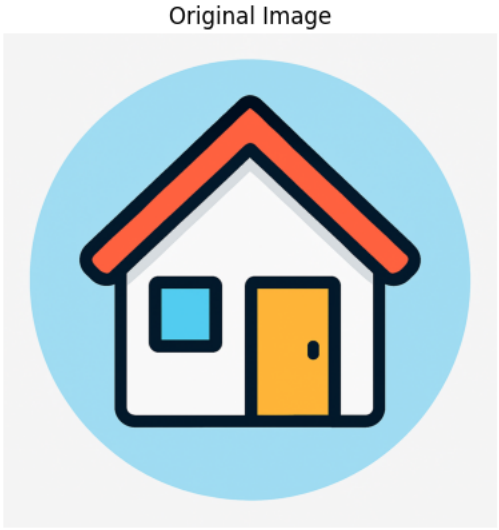
```
from PIL import Image
import matplotlib.pyplot as plt

# Mount Google Drive first
from google.colab import drive
drive.mount('/content/drive')

# Open the image
img = Image.open('/content/drive/My Drive/TSU-AI-SUMMERCAMP-2025/ai-camp/sample_image2.png')

plt.imshow(img)
plt.axis('off')  # Eksenleri kapat (temiz görünüm için)
plt.title("Original Image")  # İsteğe bağlı başlık
plt.show()
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remoun
```



📩 Read and Resize Image (with Pillow)

```
from PIL import Image
import numpy as np

img = Image.open('/content/drive/My Drive/TSU-AI-SUMMERCAMP-2025/ai-camp/sample_image2.png')  # Load image
img = img.resize((28, 28))            # Resize to 28x28
img_array = np.array(img)             # Convert to array

print(img_array.shape)
```

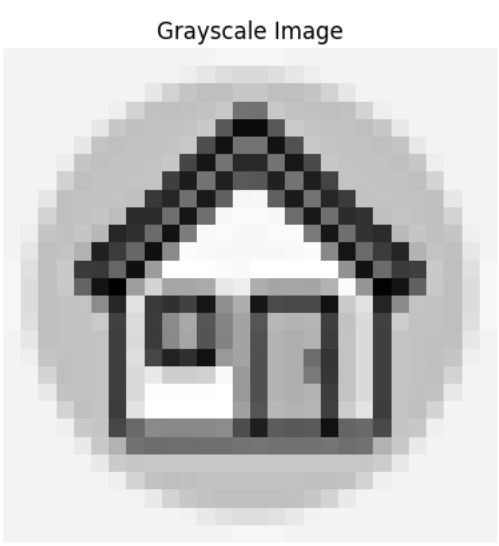
```
(28, 28, 3)
```

🌑 Convert to Grayscale

```
gray_img = img.convert('L')  # 'L' mode is for grayscale / Each pixel now has a single brightness value between 0 (black) ar
gray_array = np.array(gray_img) # This line converts the grayscale image into a numerical matrix (array) using NumPy.
```

```
plt.imshow(gray_array, cmap='gray')
plt.axis('off')
plt.title("Grayscale Image")
```

```
plt.show()
```



⚖️ Normalize Pixel Values

```
normalized = gray_array / 255.0  # Scale pixels to 0–1
print(normalized[:5])           # See first 5 rows
```

Show hidden output

```
plt.imshow(normalized, cmap='gray')
plt.axis('off')
plt.title("Normalized Grayscale Image")
plt.show()
```

Show hidden output

MNIST - Let's Explore the Data!

```
from sklearn.datasets import fetch_openml
import numpy as np

# MNIST dataset'ini yükle
#'mnist_784' → OpenML'deki doğru veri seti adıdır
#version=1 → Hangi versiyonun çekileceğini belirtir (genellikle 1 yeterlidir)
#as_frame=False → NumPy array olarak almanı sağlar, veri çerçevesi değil

X, y = fetch_openml('mnist_784', version=1, return_X_y=True, as_frame=False)

# Rakamların dağılımını hesapla
unique, counts = np.unique(y, return_counts=True)
print(dict(zip(unique, counts)))
```

```
{'0': np.int64(6903), '1': np.int64(7877), '2': np.int64(6990), '3': np.int64(7141), '4': np.int64(6824), '5': np.int64(
```

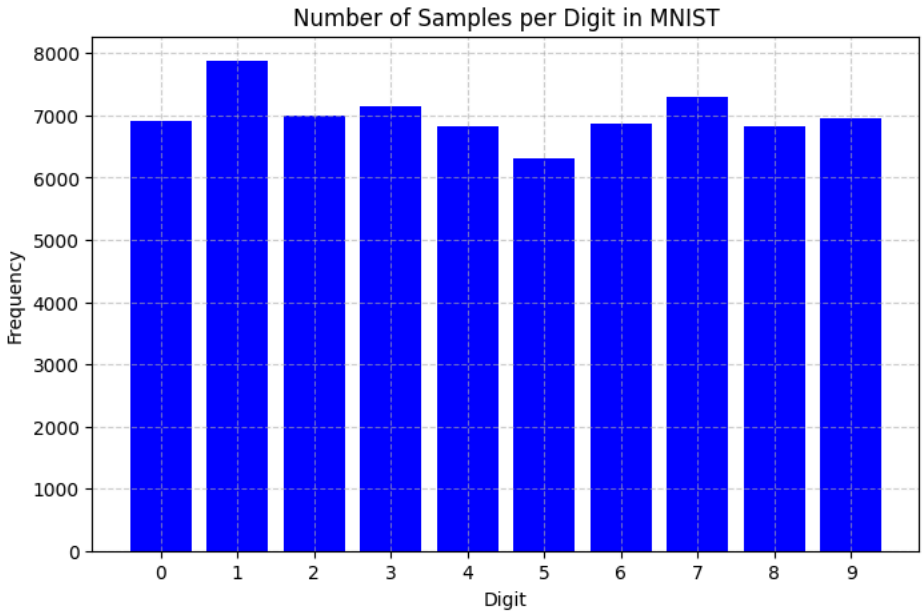
Show the distribution with a bar chart:

```
import matplotlib.pyplot as plt

# String etiketleri integer'a çevir
unique = list(map(int, unique))
counts = list(map(int, counts))

# Görselleştir
plt.figure(figsize=(8, 5))
plt.bar(unique, counts, color='blue')
plt.xlabel("Digit")
plt.ylabel("Frequency")
plt.title("Number of Samples per Digit in MNIST")
plt.grid(True, linestyle='--', alpha=0.6)
plt.xticks(unique)
```

```
plt.show()
```



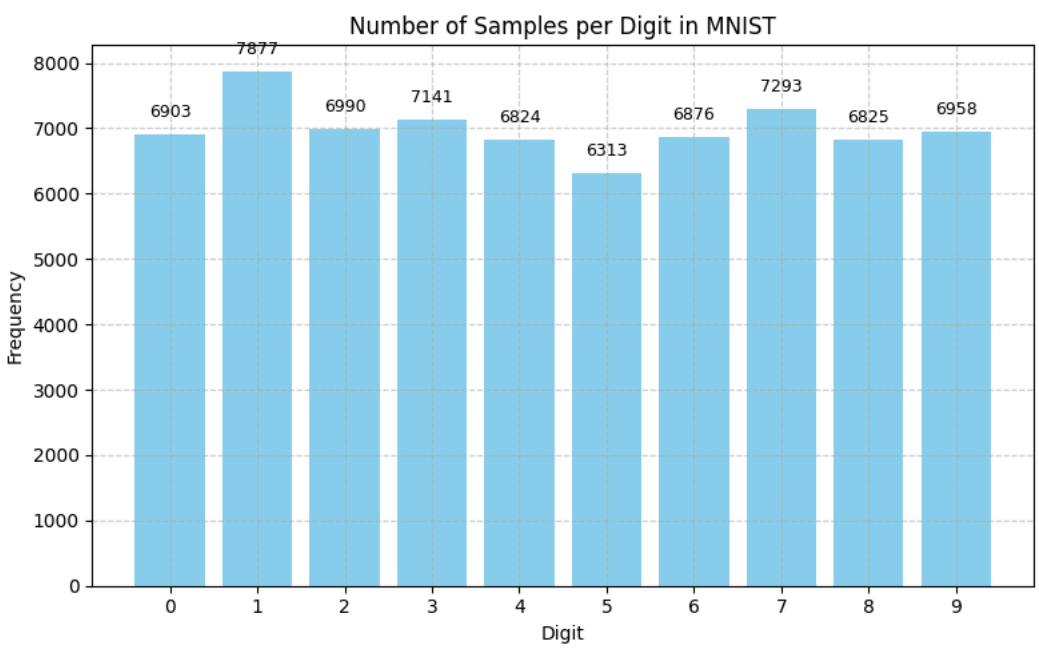

```
# Veri setini yükle
X, y = fetch_openml('mnist_784', version=1, return_X_y=True, as_frame=False)

# Etiketleri say ve integer yap
unique, counts = np.unique(y, return_counts=True)
unique = list(map(int, unique))
counts = list(map(int, counts))

# Grafik
plt.figure(figsize=(8, 5))
bars = plt.bar(unique, counts, color='skyblue')
plt.xlabel("Digit")
plt.ylabel("Frequency")
plt.title("Number of Samples per Digit in MNIST")
plt.grid(True, linestyle='--', alpha=0.6)
plt.xticks(unique)

# Her bar'ın üstüne değeri yaz
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, height + 200, f'{int(height)}',
             ha='center', va='bottom', fontsize=9)

plt.tight_layout()
plt.show()
```



Simple KNN Classifier

```
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load MNIST data
X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
X = X / 255.0  # Normalise pixel values

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a KNN classifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

# Predict and evaluate
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9712857142857143
```