Program:

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D
from keras.models import Model
from keras.datasets import mnist
```

In [2]:
```
(x_train, _), (x_test, _) = mnist.load_data()
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
```

In [3]:
```
noise_factor = 0.5
x_train_noisy = x_train + noise_factor * np.random.normal(size = x_train.shape)
x_test_noisy = x_test + noise_factor * np.random.normal(size = x_test.shape)
x_train_noisy = np.clip(x_train_noisy, 0., 1.)
x_test_noisy = np.clip(x_test_noisy, 0., 1.)
```

In [4]:
```
input_img = Input(shape = (28, 28, 1))
```

In [5]:
```
x = Conv2D(32, (3, 3), activation = 'relu', padding = 'same')(input_img)
x = MaxPooling2D((2, 2), padding = 'same')(x)
x = Conv2D(64, (3, 3), activation = 'relu', padding = 'same')(x)
x = MaxPooling2D((2, 2), padding = 'same')(x)
```

In [6]:
```
encoded = Conv2D(128, (3, 3), activation = 'relu', padding = 'same')(x)
```

In [7]:
```
x = Conv2D(64, (3, 3), activation = 'relu', padding ='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), activation = 'relu', padding = 'same')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(1, (3, 3), activation = 'sigmoid', padding = 'same')(x)
```

In [8]:
```
autoencoder = Model(input_img, decoded)
```

In [9]:
```
autoencoder.compile(optimizer = 'adam', loss = 'binary_crossentropy')
```

In [10]:
```
autoencoder.fit(x_train_noisy, x_train, epochs = 5, batch_size = 28, shuffle = True, validation_data = (x_test_noisy, x_test))
```

```
Epoch 1/5
2143/2143 [==============================] - 140s 65ms/step - loss: 0.1147 - val_loss: 0.0979
Epoch 2/5
2143/2143 [==============================] - 181s 85ms/step - loss: 0.0971 - val_loss: 0.0953
Epoch 3/5
2143/2143 [==============================] - 175s 82ms/step - loss: 0.0947 - val_loss: 0.0938
Epoch 4/5
2143/2143 [==============================] - 168s 78ms/step - loss: 0.0935 - val_loss: 0.0935
Epoch 5/5
2143/2143 [==============================] - 156s 73ms/step - loss: 0.0927 - val_loss: 0.0922
```

Out[10]:
```
<keras.callbacks.History at 0x277ad0eeaf0>
```

In [11]:
```
denoised_images = autoencoder.predict(x_test_noisy)
```

```
313/313 [==============================] - 9s 20ms/step
```

In [12]:
```
n = 10
plt.figure(figsize = (20, 4))

for i in range(n):
    # Original images
    ax = plt.subplot(3, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # Noisy images
    ax = plt.subplot(3, n, i + 1 + n)
    plt.imshow(x_test_noisy[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # Denoised images
    ax = plt.subplot(3, n, i + 1 + 2 * n)
    plt.imshow(denoised_images[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

plt.show()
```