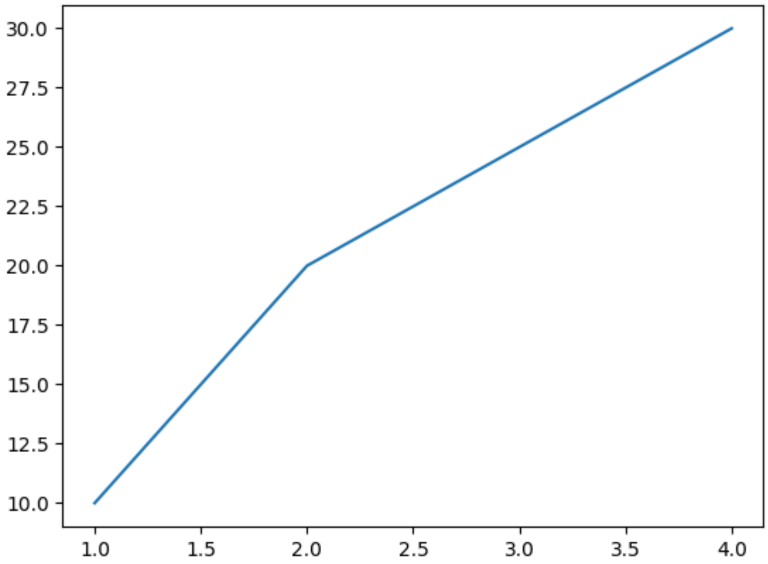
1st program and output:

```
#Sample Python Program
print("Hello, World!")

#Example of a simple calculation
a = 5
b = 10
sum = a+b
print(f"The sum of {a} and {b} is {sum}")
```

```
Hello, World!
The sum of 5 and 10 is 15
```

Plotting program(optional):

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]
plt.plot(x, y)
plt.show()
```



2nd Program and output:

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Load the dataset
# Use np.loadtxt with skiprows to skip the header row
dataset = np.loadtxt('//content/pima-indians-diabetes.csv', delimiter=',', skiprows=1)
# Split into input (X) and output (y)
X = dataset[:, 0:8]
y = dataset[:, 8]

# Define the Keras model
model = Sequential()
model.add(Dense(12, input_shape=(8,), activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Compile the Keras model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Fit the Keras model on the dataset
model.fit(X, y, epochs=150, batch_size=10, verbose=0)

# Make class predictions with the model
```

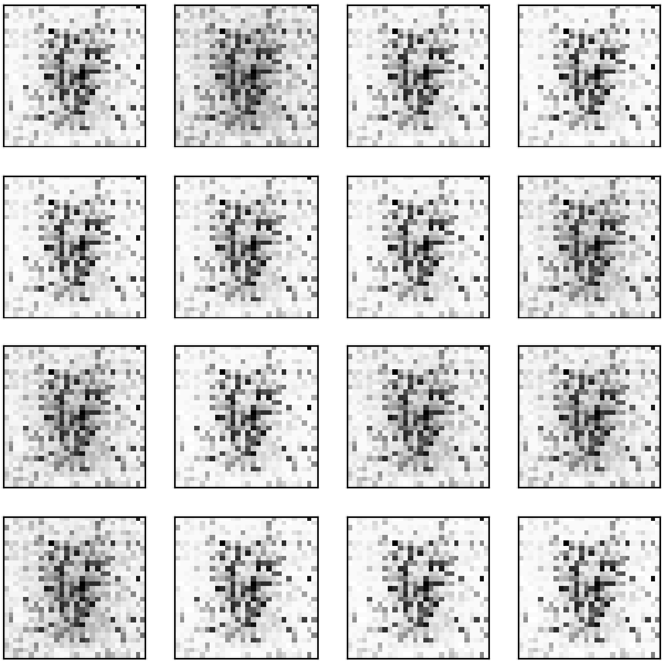
8th program and output:

```
import torch
from torch import nn
import matplotlib.pyplot as plt
import torchvision
import torchvision.transforms as transforms

# Set the manual seed for reproducibility
torch.manual_seed(111)

# Check if CUDA is available and set the device accordingly
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Define transformations for MNIST dataset (normalize and convert to tensor)
transform = transforms.Compose(
    [transforms.ToTensor(), transforms.Normalize((0.5,), (0.5,))]
)

# Download and load the MNIST training dataset
train_set = torchvision.datasets.MNIST(
    root=".", train=True, download=True, transform=transform
)
batch_size = 32
train_loader = torch.utils.data.DataLoader(
    train_set, batch_size=batch_size, shuffle=True
)

# Discriminator class definition
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(784, 1024), nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(1024, 512), nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(512, 256), nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(256, 1), nn.Sigmoid(),
        )

    def forward(self, x):
        x = x.view(x.size(0), 784)
        return self.model(x)

# Generator class definition
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(100, 256), nn.ReLU(),
            nn.Linear(256, 512), nn.ReLU(),
            nn.Linear(512, 1024), nn.ReLU(),
            nn.Linear(1024, 784), nn.Tanh(),
        )

    def forward(self, x):
        return self.model(x).view(x.size(0), 1, 28, 28)

# Initialize models
discriminator = Discriminator().to(device)
generator = Generator().to(device)

# Hyperparameters
lr = 0.0001
num_epochs = 50
loss_function = nn.BCELoss()

# Optimizers
optimizer_discriminator = torch.optim.Adam(discriminator.parameters(), lr=lr)
optimizer_generator = torch.optim.Adam(generator.parameters(), lr=lr)

# Training the GAN
for epoch in range(num_epochs):
    for real_samples, _ in train_loader:
```

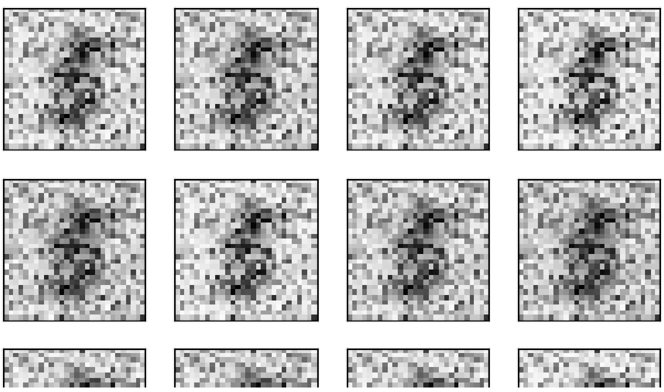
```
    real_samples = real_samples.to(device)
    real_samples_labels = torch.ones((real_samples.size(0), 1)).to(device)

    # Generate random latent space samples
    latent_space_samples = torch.randn((real_samples.size(0), 100)).to(device)
    generated_samples = generator(latent_space_samples)
    generated_samples_labels = torch.zeros((real_samples.size(0), 1)).to(device)

    # Train discriminator
    all_samples = torch.cat((real_samples, generated_samples))
    all_samples_labels = torch.cat((real_samples_labels, generated_samples_labels))

    discriminator.zero_grad()
    output_discriminator = discriminator(all_samples)
    loss_discriminator = loss_function(output_discriminator, all_samples_labels)
    loss_discriminator.backward()
    optimizer_discriminator.step()

    # Train generator
    latent_space_samples = torch.randn((real_samples.size(0), 100)).to(device)
    generator.zero_grad()
    generated_samples = generator(latent_space_samples)
    output_discriminator_generated = discriminator(generated_samples)
    loss_generator = loss_function(output_discriminator_generated, real_samples_labels)
    loss_generator.backward()
    optimizer_generator.step()

# Display generated images after each epoch
latent_space_samples = torch.randn(16, 100).to(device)
generated_samples = generator(latent_space_samples).cpu().detach()
plt.figure(figsize=(6, 6))
for i in range(16):
    ax = plt.subplot(4, 4, i + 1)
    plt.imshow(generated_samples[i].reshape(28, 28), cmap="gray_r")
    plt.xticks([])
    plt.yticks([])
plt.suptitle(f"Epoch {epoch+1}")
plt.show()
```

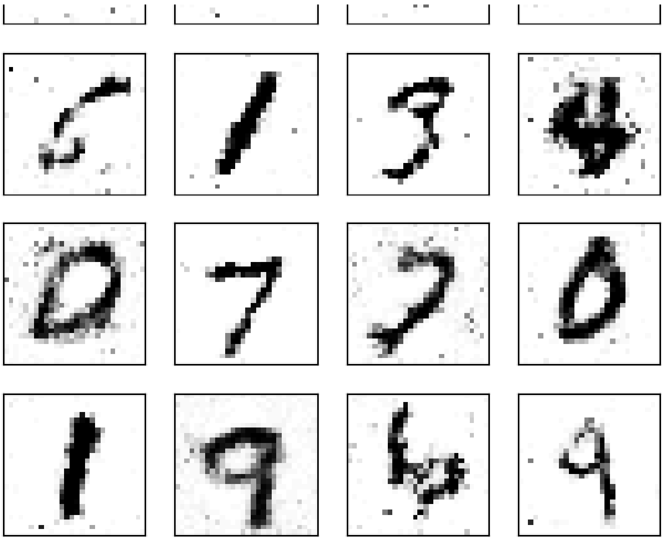
```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-idx3-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-idx3-ubyte.gz to ./MNIST/raw/train-images-idx3
100%|██████████| 9.91M/9.91M [00:00<00:00, 16.3MB/s]
Extracting ./MNIST/raw/train-images-idx3-ubyte.gz to ./MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-labels-idx1-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-labels-idx1-ubyte.gz to ./MNIST/raw/train-labels-idx1
100%|██████████| 28.9k/28.9k [00:00<00:00, 561kB/s]
Extracting ./MNIST/raw/train-labels-idx1-ubyte.gz to ./MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-images-idx3-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-images-idx3-ubyte.gz to ./MNIST/raw/t10k-images-idx3-u
100%|██████████| 1.65M/1.65M [00:00<00:00, 3.94MB/s]
Extracting ./MNIST/raw/t10k-images-idx3-ubyte.gz to ./MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-labels-idx1-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-labels-idx1-ubyte.gz to ./MNIST/raw/t10k-labels-idx1-u
100%|██████████| 4.54k/4.54k [00:00<00:00, 8.66MB/s]
Extracting ./MNIST/raw/t10k-labels-idx1-ubyte.gz to ./MNIST/raw
```

Epoch 1

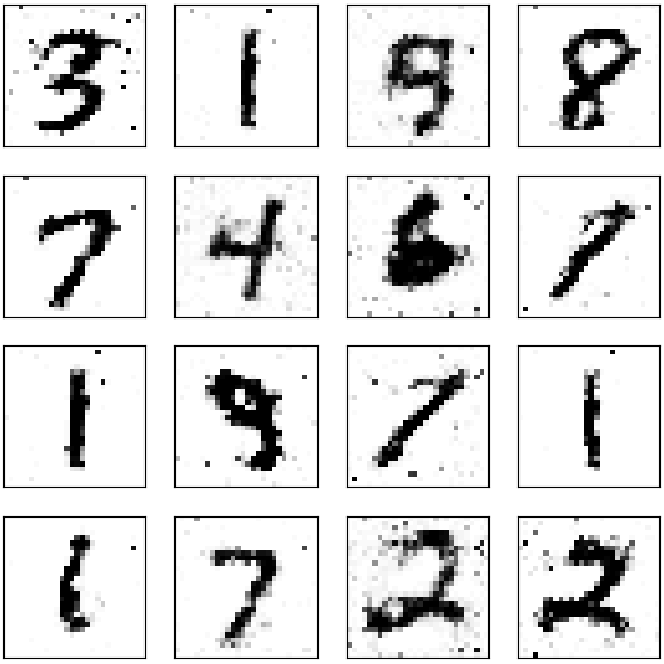


Epoch 2

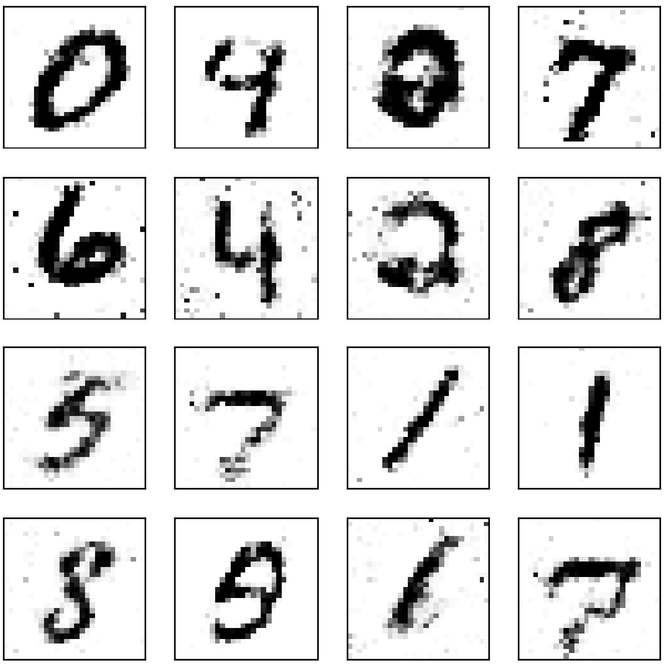




Epoch 48



Epoch 49



Epoch 50