write a program to implement autoencoder for both binary and real value inputs clearly show the loss after each iteration. consider no on input are 5

```
import numpy as np
import tensorflow as tf
```

Binary values

```
model = tf.keras.Sequential([
        tf.keras.layers.Dense(5, activation="relu", name="firstlayer"),
        tf.keras.layers.Dense(4, activation="sigmoid", name="secondlayer"),
        tf.keras.layers.Dense(5, name="lastlayer"),
])
```

```
input = np.random.randint(2, size=(100, 5)).astype(np.float32)
output = model(input)
for layer in model.layers:
    print(layer.name, layer)
    print('Weights: ',layer.weights)
model.compile( loss='binary_crossentropy')
history = model.fit(input, input,epochs=1,batch_size=32,shuffle=True)
print('Final Loss:', history.history['loss'][-1])
```

```
firstlayer <keras.layers.core.dense.Dense object at 0x7f8fcfe9ecd0>
Weights:  [<tf.Variable 'firstlayer/kernel:0' shape=(5, 5) dtype=float32, numpy=
array([[ 0.53326225, -0.10942608,  0.32509243,  0.11350602,  0.19812894],
       [ 0.41962945, -0.12963545, -0.05914915, -0.54850876,  0.64977527],
       [-0.29979613, -0.02878541, -0.05756992, -0.7387501 , -0.5112642 ],
       [ 0.07005841, -0.5850917 ,  0.36555064,  0.01529211, -0.34184867],
       [ 0.506021  , -0.04194266, -0.22721982, -0.07620341,  0.5805459 ]],
      dtype=float32)>, <tf.Variable 'firstlayer/bias:0' shape=(5,) dtype=float32, nur
secondlayer <keras.layers.core.dense.Dense object at 0x7f8fcfe9e040>
Weights:  [<tf.Variable 'secondlayer/kernel:0' shape=(5, 4) dtype=float32, numpy=
array([[-0.0618571 , -0.5493538 ,  0.73624   ,  0.36165845],
       [ 0.2741413 ,  0.46761012,  0.34199345, -0.06574208],
       [-0.4901198 , -0.52650833,  0.01692754,  0.23970747],
       [ 0.3731326 , -0.19655919, -0.7844214 , -0.26475   ],
       [ 0.5005809 ,  0.13764775, -0.27328658, -0.16576737]],
      dtype=float32)>, <tf.Variable 'secondlayer/bias:0' shape=(4,) dtype=float32, nu
lastlayer <keras.layers.core.dense.Dense object at 0x7f8fcfe9e6d0>
Weights:  [<tf.Variable 'lastlayer/kernel:0' shape=(4, 5) dtype=float32, numpy=
array([[-0.03466958,  0.14620262, -0.48409212,  0.33834338, -0.086564  ],
       [-0.3265882 ,  0.11645347, -0.46598876, -0.0239566 , -0.37109163],
       [-0.5047096 ,  0.8126558 ,  0.1492638 , -0.7454623 , -0.42343727],
       [-0.3005321 ,  0.3333192 , -0.25321418,  0.03942257, -0.09274739]],
      dtype=float32)>, <tf.Variable 'lastlayer/bias:0' shape=(5,) dtype=float32, nump
4/4 [==============================] - 1s 4ms/step - loss: 6.5612
Final Loss: 6.561158657073975
```

```
train_data = np.random.randint(2, size=(100, 5)).astype(np.float32)

encod_dim = 4
input_data = tf.keras.layers.Input(shape=(5,))
encoded = tf.keras.layers.Dense(encod_dim, activation='relu')(input_data)
decoded = tf.keras.layers.Dense(5, activation='sigmoid')(encoded)
autoencoder = tf.keras.models.Model(input_data, decoded)

autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

history = autoencoder.fit(train_data, train_data,epochs=50,batch_size=32,shuffle=True)

print('Loss:', history.history['loss'][-1])
```

```
Epoch 1/50
4/4 [==============================] - 1s 3ms/step - loss: 0.7425
Epoch 2/50
4/4 [==============================] - 0s 3ms/step - loss: 0.7396
Epoch 3/50
4/4 [==============================] - 0s 3ms/step - loss: 0.7368
Epoch 4/50
4/4 [==============================] - 0s 4ms/step - loss: 0.7343
Epoch 5/50
4/4 [==============================] - 0s 5ms/step - loss: 0.7317
Epoch 6/50
4/4 [==============================] - 0s 4ms/step - loss: 0.7292
Epoch 7/50
4/4 [==============================] - 0s 3ms/step - loss: 0.7267
Epoch 8/50
4/4 [==============================] - 0s 3ms/step - loss: 0.7245
Epoch 9/50
4/4 [==============================] - 0s 3ms/step - loss: 0.7223
Epoch 10/50
4/4 [==============================] - 0s 3ms/step - loss: 0.7202
Epoch 11/50
4/4 [==============================] - 0s 3ms/step - loss: 0.7181
Epoch 12/50
4/4 [==============================] - 0s 3ms/step - loss: 0.7162
Epoch 13/50
4/4 [==============================] - 0s 3ms/step - loss: 0.7143
Epoch 14/50
4/4 [==============================] - 0s 3ms/step - loss: 0.7123
Epoch 15/50
4/4 [==============================] - 0s 5ms/step - loss: 0.7104
Epoch 16/50
4/4 [==============================] - 0s 3ms/step - loss: 0.7086
Epoch 17/50
4/4 [==============================] - 0s 3ms/step - loss: 0.7068
Epoch 18/50
4/4 [==============================] - 0s 3ms/step - loss: 0.7051
Epoch 19/50
4/4 [==============================] - 0s 3ms/step - loss: 0.7033
Epoch 20/50
4/4 [==============================] - 0s 3ms/step - loss: 0.7016
Epoch 21/50
```

```
4/4 [==============================] - 0s 3ms/step - loss: 0.7000
Epoch 22/50
4/4 [==============================] - 0s 3ms/step - loss: 0.6984
Epoch 23/50
4/4 [==============================] - 0s 3ms/step - loss: 0.6969
Epoch 24/50
4/4 [==============================] - 0s 3ms/step - loss: 0.6954
Epoch 25/50
4/4 [==============================] - 0s 4ms/step - loss: 0.6939
Epoch 26/50
4/4 [==============================] - 0s 3ms/step - loss: 0.6924
Epoch 27/50
4/4 [==============================] - 0s 3ms/step - loss: 0.6911
Epoch 28/50
4/4 [==============================] - 0s 3ms/step - loss: 0.6896
Epoch 29/50
```

Linear values

```
model = tf.keras.Sequential([
        tf.keras.layers.Dense(5, activation="relu", name="firstlayer"),
        tf.keras.layers.Dense(4, activation="linear", name="secondlayer"),
        tf.keras.layers.Dense(5, name="lastlayer"),
])
```

```
input = tf.random.normal((100,5))
output = model(input)
for layer in model.layers:
    print(layer.name, layer)
    print('Weights: ',layer.weights)
```

```
firstlayer <keras.layers.core.dense.Dense object at 0x7f8fcfd00e20>
Weights:  [<tf.Variable 'firstlayer/kernel:0' shape=(5, 5) dtype=float32, numpy=
array([[-0.24975276,  0.4810909 , -0.5320791 ,  0.7539711 ,  0.24648881],
       [ 0.17211902,  0.39109492,  0.2079612 ,  0.11927444, -0.43806455],
       [-0.2953354 , -0.15944254,  0.45107234,  0.24857521,  0.6365212 ],
       [ 0.6294944 ,  0.6645646 , -0.6526749 , -0.7595022 ,  0.5032488 ],
       [ 0.77332735,  0.6500844 ,  0.04204106, -0.06026644,  0.2080949 ]],
      dtype=float32)>, <tf.Variable 'firstlayer/bias:0' shape=(5,) dtype=float32, nur
secondlayer <keras.layers.core.dense.Dense object at 0x7f8fcfc01ac0>
Weights:  [<tf.Variable 'secondlayer/kernel:0' shape=(5, 4) dtype=float32, numpy=
array([[ 0.66624033,  0.01462227,  0.2573396 ,  0.65799856],
       [ 0.5944427 ,  0.15855569, -0.25065488, -0.511574  ],
       [-0.082744  ,  0.29955816,  0.18981183,  0.43079865],
       [-0.78267306,  0.5456152 ,  0.05659211, -0.5830037 ],
       [-0.5530896 ,  0.48820364, -0.021065  ,  0.6567867 ]],
      dtype=float32)>, <tf.Variable 'secondlayer/bias:0' shape=(4,) dtype=float32, nu
lastlayer <keras.layers.core.dense.Dense object at 0x7f8fcfc01bb0>
Weights:  [<tf.Variable 'lastlayer/kernel:0' shape=(4, 5) dtype=float32, numpy=
array([[-0.31567943,  0.76454306, -0.7117707 , -0.81235975,  0.30468595],
       [ 0.81520844, -0.22948003,  0.64763653, -0.8018759 ,  0.19788098],
       [-0.7388946 , -0.18589348,  0.8027221 , -0.02062935, -0.66245234],
       [ 0.11226249,  0.36891687, -0.70356953, -0.22070205, -0.8002183 ]],
      dtype=float32)>, <tf.Variable 'lastlayer/bias:0' shape=(5,) dtype=float32, nump
```

```
train_data = np.random.rand(1000, 5)

encod_dim = 4
input_data = tf.keras.layers.Input(shape=(5,))
encoded = tf.keras.layers.Dense(encod_dim, activation='relu')(input_data)
decoded = tf.keras.layers.Dense(5, activation='linear')(encoded)
autoencoder = tf.keras.models.Model(input_data, decoded)

autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

history = autoencoder.fit(train_data, train_data,epochs=1,batch_size=32,shuffle=True)

print('Loss:', history.history['loss'][-1])
```

```
32/32 [==============================] - 1s 2ms/step - loss: 3.6370
Loss: 3.637042284011841
```