```
# Aim: To Study and installation of following Deep learning Packages :
# i. Tensor Flow
# ii. Keras
# iii. Theno
# iv . PyTorch

# Install Keras and Theano
!pip install keras theano
```

```
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages (3.4.1)
Requirement already satisfied: theano in /usr/local/lib/python3.10/dist-packages (1.0.5)
Requirement already satisfied: absl-py in /usr/local/lib/python3.10/dist-packages (from keras) (1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from keras) (1.26.4)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages (from keras) (13.9.2)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages (from keras) (0.0.8)
Requirement already satisfied: h5py in /usr/local/lib/python3.10/dist-packages (from keras) (3.11.0)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages (from keras) (0.13.0)
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.10/dist-packages (from keras) (0.4.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from keras) (24.1)
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.10/dist-packages (from theano) (1.13.1)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from theano) (1.16.0)
Requirement already satisfied: typing-extensions>=4.5.0 in /usr/local/lib/python3.10/dist-packages (from optree->keras) (4.12.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich->keras) (0.1
```

```
# Import required libraries for version checking
import tensorflow as tf
import keras
# import theano
import torch


# Check versions of deep learning libraries
print(f"TensorFlow version: {tf.__version__}")
print(f"Keras version: {keras.__version__}")
# print(f"Theano version: {theano.__version__}")
print(f"PyTorch version: {torch.__version__}")
```

```
TensorFlow version: 2.17.0
Keras version: 3.4.1
PyTorch version: 2.4.1+cu121
```

```
# 1. TensorFlow
print("\nTensorFlow Example:")
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Define a simple model using TensorFlow
tf_model = Sequential()
tf_model.add(Dense(10, input_shape=(5,), activation='relu'))
tf_model.add(Dense(1, activation='sigmoid'))

# Compile the TensorFlow model
tf_model.compile(optimizer='adam', loss='binary_crossentropy')
tf_model.summary()
```

```
TensorFlow Example:
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` arg
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 10)	60
dense_1 (Dense)	(None, 1)	11

```
Total params: 71 (284.00 B)
Trainable params: 71 (284.00 B)
Non-trainable params: 0 (0.00 B)
```

```
# 2. Keras
print("\nKeras Example (Note: Keras is integrated with TensorFlow):")
# Keras is now integrated with TensorFlow, hence the example above is also valid for Keras.
# Define a simple model using Keras directly
keras_model = Sequential()
keras_model.add(Dense(10, input_shape=(5,), activation='relu'))
keras_model.add(Dense(1, activation='sigmoid'))
```

```
# Compile the Keras model
keras_model.compile(optimizer='adam', loss='binary_crossentropy')
keras_model.summary()
```

```
Keras Example (Note: Keras is integrated with TensorFlow):
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
dense_2 (Dense)	(None, 10)	60
dense_3 (Dense)	(None, 1)	11

```
Total params: 71 (284.00 B)
Trainable params: 71 (284.00 B)
```

```
# # 3. Theano
# print("\nTheano Example:")
# import theano.tensor as T
# from theano import function

# # Define simple variables and function in Theano
# x = T.dscalar('x')
# y = T.dscalar('y')
# z = x + y
# theano_function = function([x, y], z)

# # Use the function
# result = theano_function(1, 2)
# print(f"Theano addition result: {result}")


# 4. PyTorch
print("\nPyTorch Example:")
import torch.nn as nn
import torch.optim as optim

# Define a simple model using PyTorch
class PyTorchModel(nn.Module):
    def __init__(self):
        super(PyTorchModel, self).__init__()
        self.fc1 = nn.Linear(5, 10)
        self.fc2 = nn.Linear(10, 1)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.sigmoid(self.fc2(x))
        return x

# Initialize PyTorch model, optimizer, and loss function
pytorch_model = PyTorchModel()
optimizer = optim.Adam(pytorch_model.parameters(), lr=0.001)
loss_fn = nn.BCELoss()

print(pytorch_model)
```

```
PyTorch Example:
PyTorchModel(
  (fc1): Linear(in_features=5, out_features=10, bias=True)
  (fc2): Linear(in_features=10, out_features=1, bias=True)
)
```