```
#!/usr/bin/env python
# coding: utf-8


# Import Numpy & PyTorch
import
import       as
import                    as
import

import          as
from                  import

# Import nn.functional
import                     as


# class LogFile(object):
#     """File-like object to log text using the `logging` module."""

#     def __init__(self, name=None):
#         self.logger = logging.getLogger(name)

#     def write(self, msg, level=logging.INFO):
#         self.logger.log(level, msg)

#     def flush(self):
#         for handler in self.logger.handlers:
#             handler.flush()

# logging.basicConfig(filename = "Linear Regression Exercise.log")

# # Redirect stdout and stderr
# sys.stdout = LogFile('stdout')
# sys.stderr = LogFile('stderr')

                            'Linear Regression.log'           'utf-8'

# Define the data
              1 1
              2 1
    100

# Create empty file
                                      0  2    "log"
with open          "w"  as    pass


def xprint
    print
        open          "a"
                  "\n"


def reg_compare
    # Define loss function
    def mse
```

```
    return

# Define model 1 (manual)
def model
    return


# Define a utility function to train the model
def fit
    for       in range
        for       in
            # Generate predictions


            # Perform gradient descent



    return

class SimpleNet
    # Initialize the layers
    def __init__ self
        super
        self                    2  2
        self                  # Activation function
        self                    2  1

    # Perform the computation
    def forward self
            self
            self
            self
        return


# Define model 2 (PyTorch)
                  2  1
                                                  001

#loss = loss_fn(model2(inputs), targets)


# Define model 3 (Neural Network)
                                              003



# Define Data
    10                    1


                        1
```

```
# Define PyTorch tensors
#X_tens = torch.tensor(X)
#X1_tens = torch.tensor(X1)
#X1_tens = x

#F1_tens = torch.from_numpy(F1)


#Y1_tens = y


# Define model inputs and targets and initalize weights and bias
#inputs_d = X1_d_tens

#inputs2 = X1_tens


                1 2                 True




# Define data loader
             5
                                                        True
# Closed form




                      1

#Train model for 100 epochs
# Iterate and modify via gradient decent
for   in range 500


    with
                      01


                     1

# Train the model 2 for 100 epochs
               100
                     1

# Train model 3 for 100 epochs
                 500
                       1
```

```
    return

                  0
                0
                0
                  0

# Set number of iterations
    100
for   in range




    if      -1
        del

# Calculate average error




       'Average Matrix inverion solution error = '  str
       'Average Manual model solution error = '  str
       'Average Pytorch model solution error = '  str
       'Average Single layer neural network solution error = '  str


# Plot data points

                     0             1
                  0             2         'k'
                  0             3                        'g'
                  0             4                        'c'
                  0             5                        'm'
                  0             6         'r'
          "X"
          "Y"
         "Linear Regression"
          'Matrix Inversion'  'Manual Model'  'PyTorch Model'  'Single Layer Neural Net'  'True'


           'Comparison plot.pdf'
```