Start coding or generate with AI.

```
# forward_propagation_spam.py

import numpy as np

# Simulated email feature vector
X = np.array([[0.1, 0.3, 0.0, 0.2]])  # Input features

# Random weights and bias for a single-layer neural net
W = np.array([[0.2], [0.4], [0.1], [0.5]])
b = 0.1

# Sigmoid activation
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

# Forward Propagation
Z = np.dot(X, W) + b
output = sigmoid(Z)

print("Spam score (sigmoid output):", output[0][0])
print("Prediction:", "Spam" if output[0][0] > 0.5 else "Not Spam")
```

```
Spam score (sigmoid output): 0.5841905229354074
Prediction: Spam
```

```
# backward_propagation_stock.py

import numpy as np

# Input: [open price, volume]
X = np.array([[1.0, 2.0]])
y_true = np.array([[300.0]])

# Initial weights and bias
W = np.array([[2.0], [1.0]])
b = np.array([[0.5]])

# Forward pass
y_pred = np.dot(X, W) + b
loss = 0.5 * (y_true - y_pred) ** 2

print("Initial Prediction:", y_pred[0][0])
print("Loss:", loss[0][0])

# Backward pass
lr = 0.01
error = y_pred - y_true
dW = np.dot(X.T, error)
db = error

# Gradient Descent Update
W -= lr * dW
b -= lr * db

print("Updated Weights:", W.flatten())
print("Updated Bias:", b.flatten())
```

```
Initial Prediction: 4.5
Loss: 43660.125
Updated Weights: [4.955 6.91 ]
Updated Bias: [3.455]
```

```
# fine_tuning_xray.py

import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense
from tensorflow.keras.models import Model

# Step 1: Load base model
base_model = MobileNetV2(weights=None, include_top=False, input_shape=(224, 224, 3))
```

```
base_model = MobileNetV2(weights=None, include_top=False, input_shape=(224, 224, 3))
base_model.trainable = False  # Freeze all layers

# Step 2: Add custom classification layers
x = GlobalAveragePooling2D()(base_model.output)
x = Dense(128, activation='relu')(x)
output = Dense(1, activation='sigmoid')(x)

model = Model(inputs=base_model.input, outputs=output)

# Step 3: Compile with top layers only
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
print("✅ Model compiled with frozen base and custom head.")

# Step 4: Simulate fine-tuning
base_model.trainable = True  # Unfreeze base for fine-tuning
model.compile(optimizer=tf.keras.optimizers.Adam(1e-5),
              loss='binary_crossentropy', metrics=['accuracy'])
print("✅ Base model unfrozen, fine-tuning ready with lower LR.")
```

```
✅ Model compiled with frozen base and custom head.
✅ Base model unfrozen, fine-tuning ready with lower LR.
```