

Here's a beginner-friendly explanation and example code for:

- Perceptron
 - Multilayer Perceptron (MLP)
-

1. Perceptron

✓ What it is:

A perceptron is the simplest type of neural network — a single-layer binary classifier.

✓ How it works:

- Takes input values x_1, x_2, \dots, x_n
- Multiplies them with weights w_1, w_2, \dots, w_n
- Adds bias and applies an activation function (like step function)

✓ Formula:

$$\text{output} = f(w_1x_1 + w_2x_2 + \dots + w_nx_n + b)$$

✓ Example using sklearn:

```
from sklearn.linear_model import Perceptron
```

```
# Example: OR logic gate
```

```
X = [[0, 0], [0, 1], [1, 0], [1, 1]]
```

```
y = [0, 1, 1, 1]
```

```
model = Perceptron(max_iter=1000, tol=1e-3)
```

```
model.fit(X, y)
```

```
print("Predictions:", model.predict(X))
```

2. Multilayer Perceptron (MLP)

✓ What it is:

An MLP is a deep neural network with:

- Input layer
- One or more hidden layers

- Output layer

Each layer uses non-linear activation functions like ReLU or sigmoid to model complex patterns.

✅ Used for:

- Classification
- Regression
- Pattern recognition

✅ Example using sklearn:

```
from sklearn.neural_network import MLPClassifier
```

```
# Example: XOR problem (not solvable by a single perceptron)
```

```
X = [[0, 0], [0, 1], [1, 0], [1, 1]]
```

```
y = [0, 1, 1, 0]
```

```
model = MLPClassifier(hidden_layer_sizes=(4,), activation='relu', max_iter=1000)
```

```
model.fit(X, y)
```

```
print("Predictions:", model.predict(X))
```

🔄 Key Differences

Feature	Perceptron	Multilayer Perceptron (MLP)
Layers	Single-layer	Multi-layer (input, hidden, output)
Capable of XOR	❌ No	✅ Yes
Non-linear learning	❌ No	✅ Yes (via hidden layers)

Would you like a visual explanation of how a neural network works step by step?