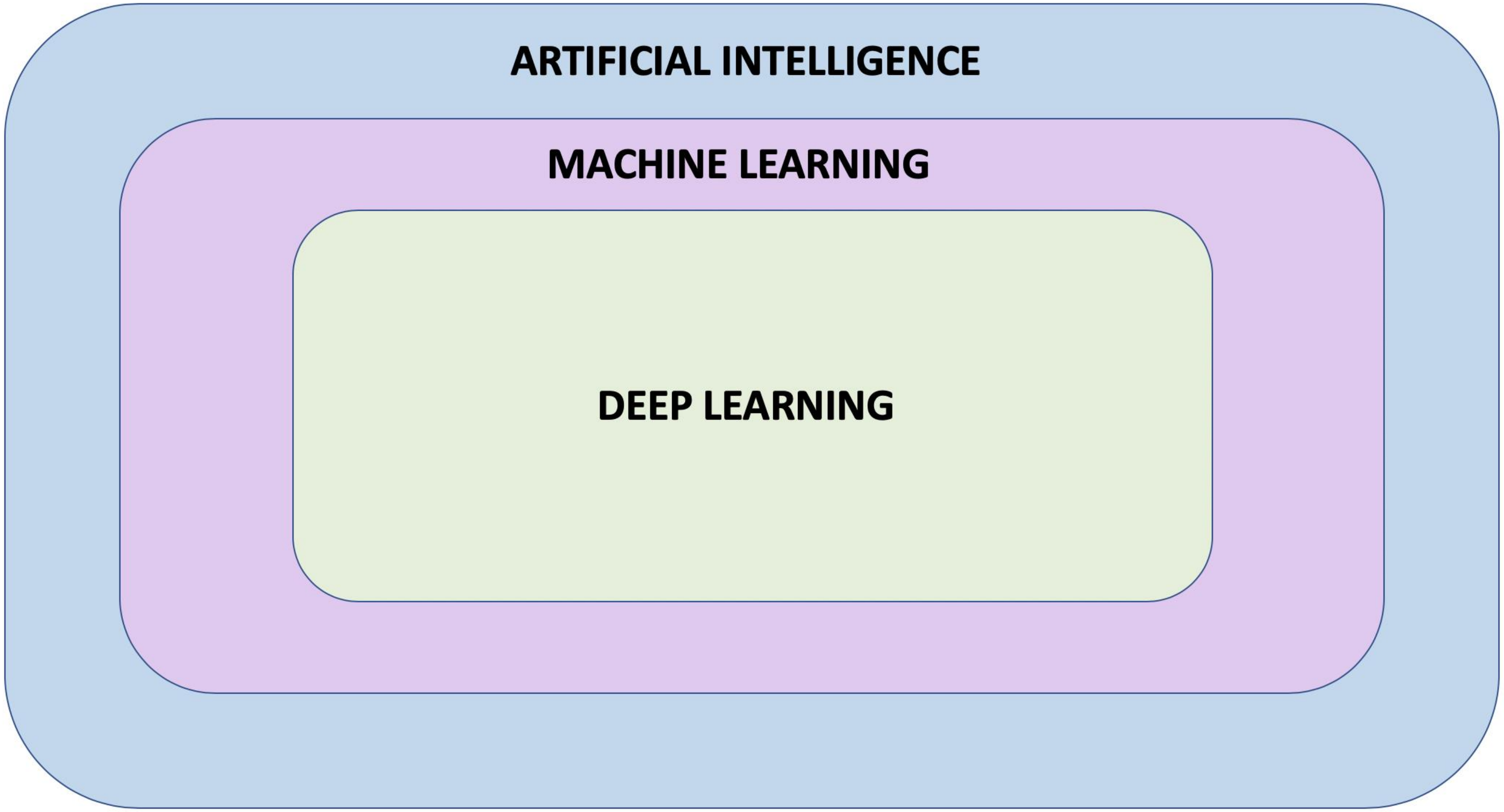
Artificial Neural Network (ANN)

Machine Learning introductory

Samar shaaban

AI – ML – ANN - DL



ARTIFICIAL INTELLIGENCE

MACHINE LEARNING

DEEP LEARNING

AI – ML – ANN - DL

- **Artificial Intelligence:** A field of computer science that aims to make computers achieve human-style intelligence. There are many approaches to reaching this goal, including machine learning and deep learning.

- **Machine Learning:** A set of related techniques in which computers are trained to perform a particular task rather than by explicitly programming them.

- **Neural Network:** A construct in Machine Learning inspired by the network of neurons (nerve cells) in the biological brain. Neural networks are a fundamental part of deep learning, and will be covered in this course.

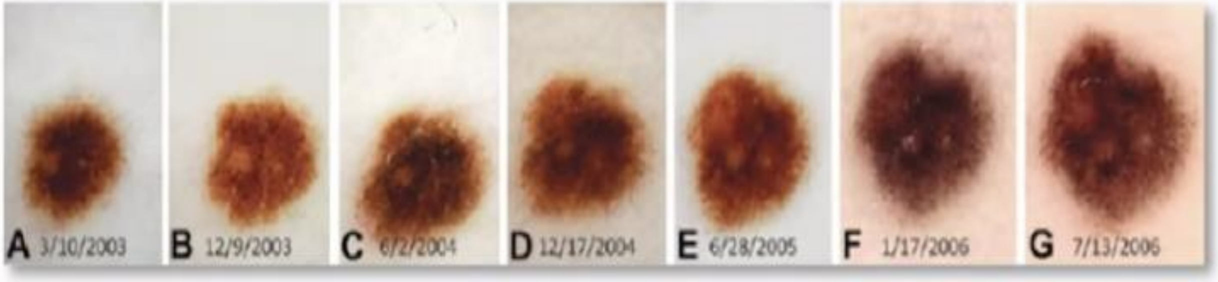
- **Deep Learning:** A subfield of machine learning that uses multi-layered neural networks. Often, “machine learning” and “deep learning” are used interchangeably.

AI – ML – ANN - DL

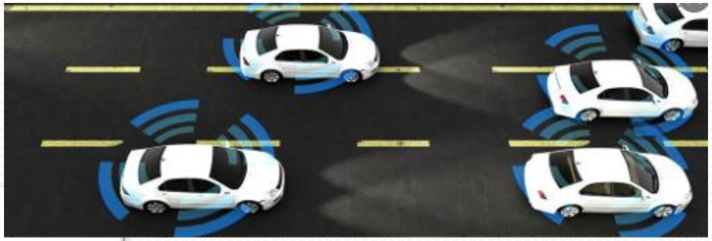
- Machine learning and deep learning also have many subfields, branches, and special techniques.
- A notable example of this diversity is the separation of **Supervised Learning** and **Unsupervised Learning**.
- supervised learning you know what you want to teach the computer.
-
- unsupervised learning is about letting the computer figure out what can be learned.

Applications

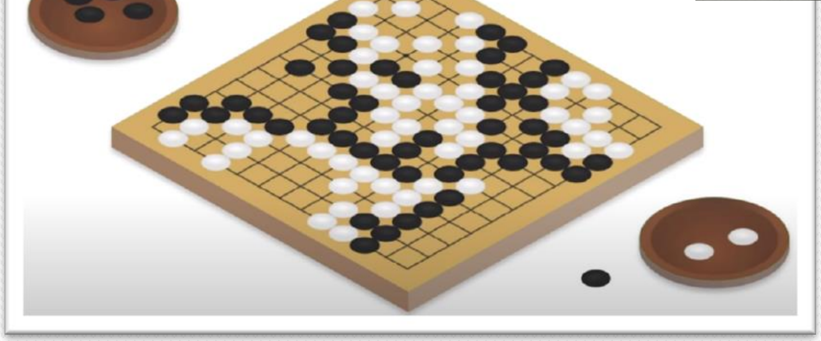
- Skin Cancer.



- Self-driving cars: autonomous vehicles



- Games



- Others.

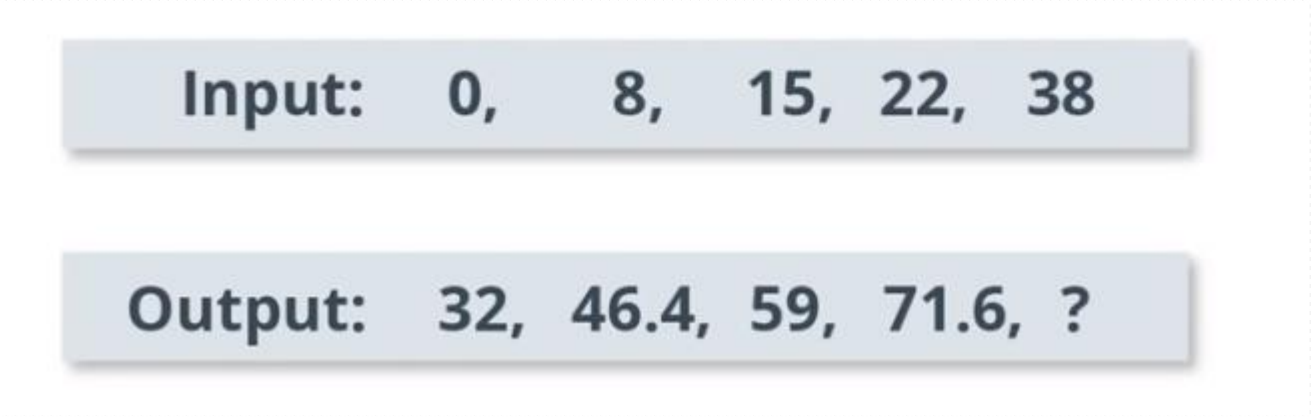
Tools and APIs

- Python programming language.
- Anaconda, jupyter notebook
- Tensorflow
- Keras
- Deployment

- Goggle Colab:

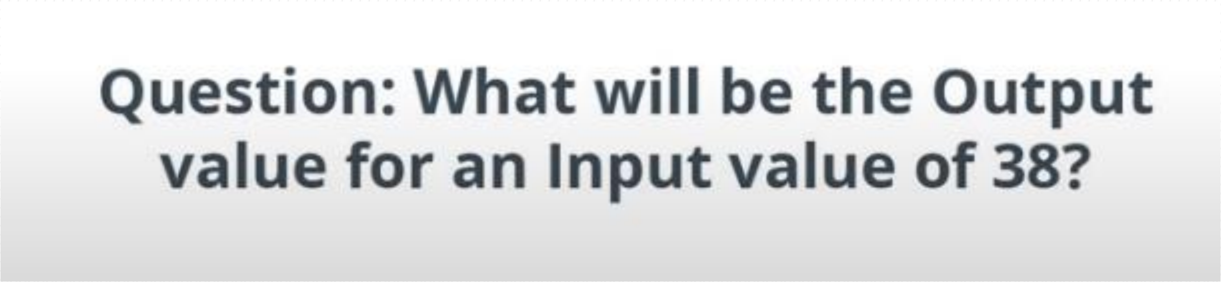
Simple example

Celsius to Fahrenheit conversion problem

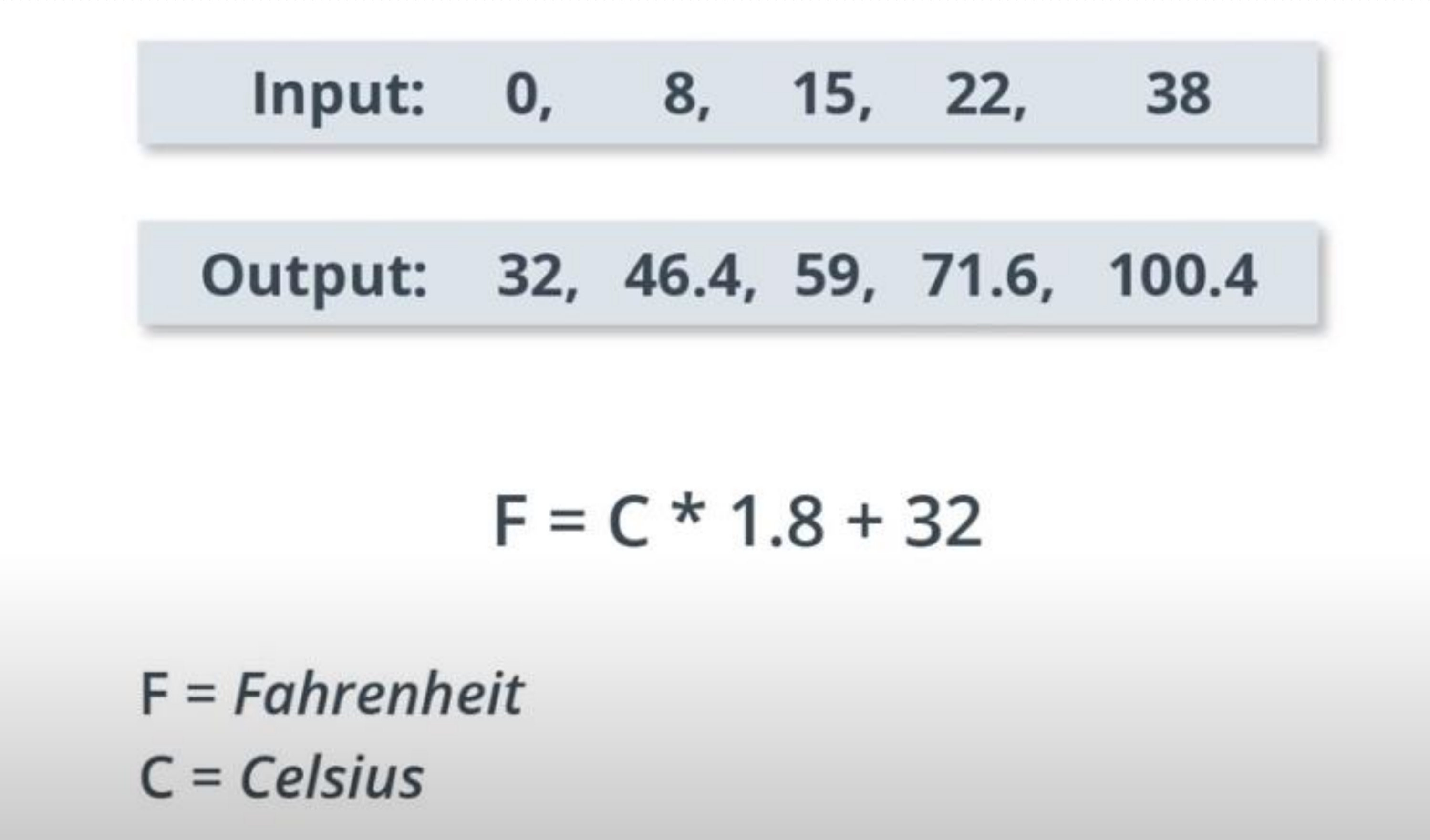


Input: 0, 8, 15, 22, 38

Output: 32, 46.4, 59, 71.6, ?



Question: What will be the Output value for an Input value of 38?

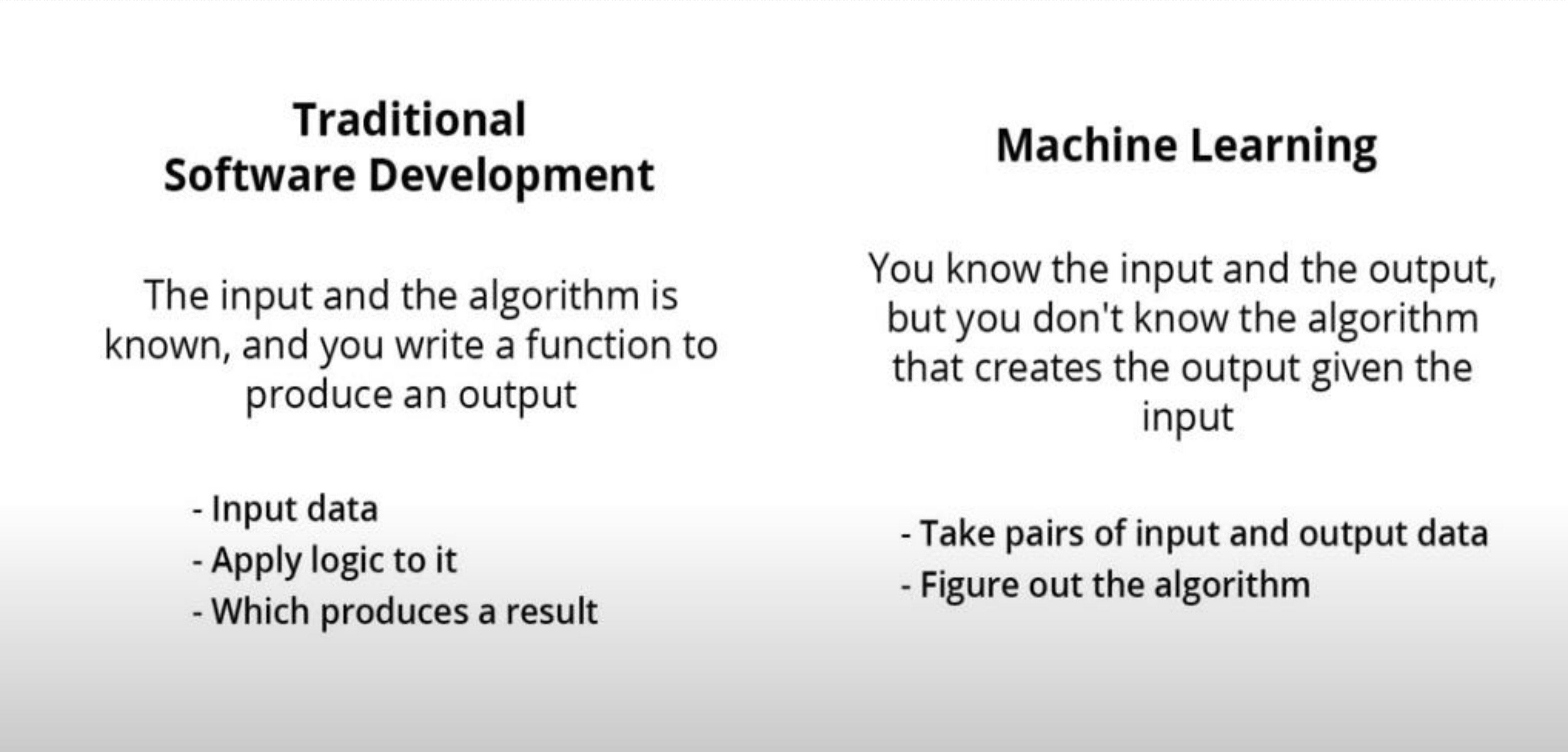


Input:	**0,**	**8,**	**15,**	**22,**	**38**
Output:	**32,**	**46.4,**	**59,**	**71.6,**	**100.4**

$$F = C * 1.8 + 32$$

F = *Fahrenheit*

C = *Celsius*



Traditional Software Development

The input and the algorithm is known, and you write a function to produce an output

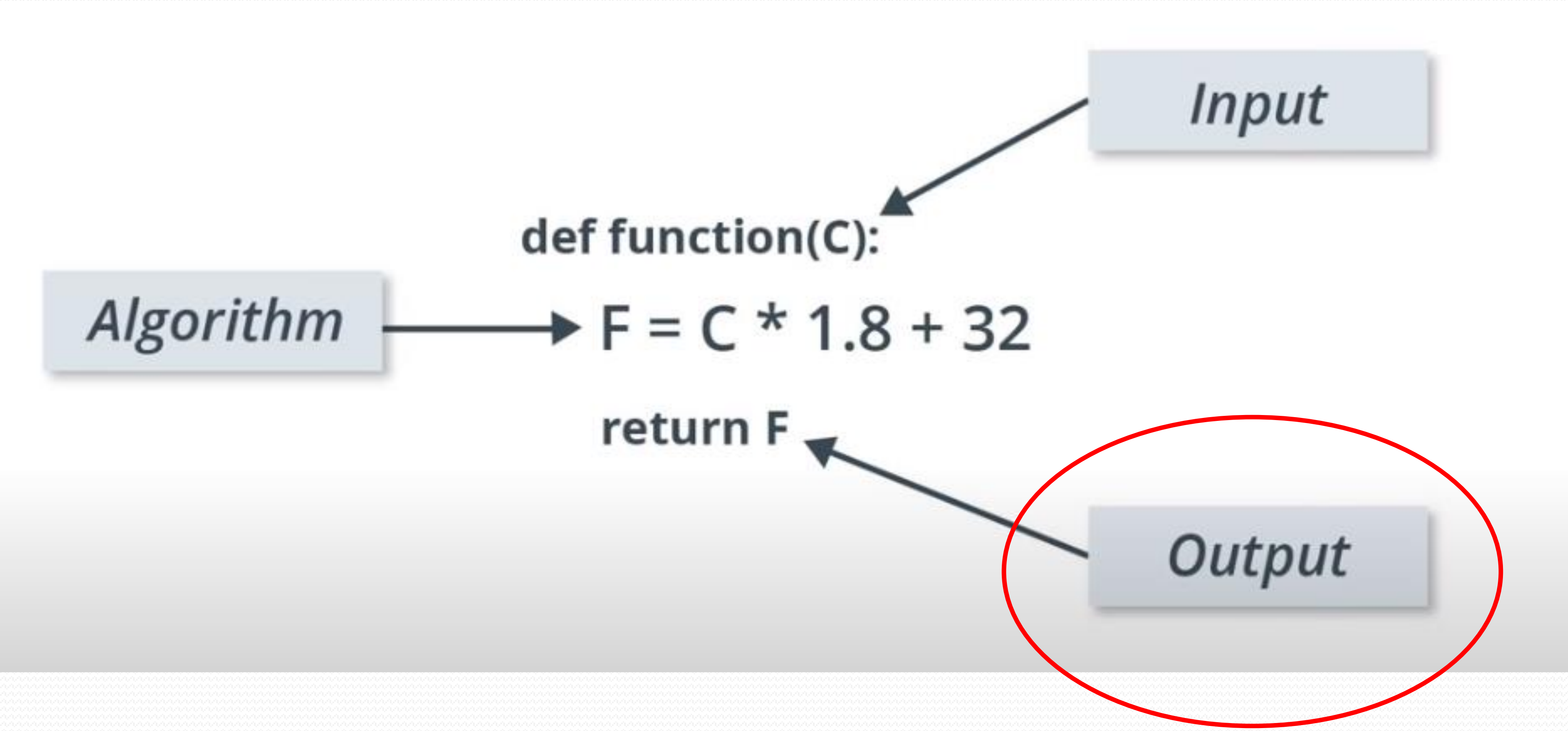
- Input data
- Apply logic to it
- Which produces a result

Machine Learning

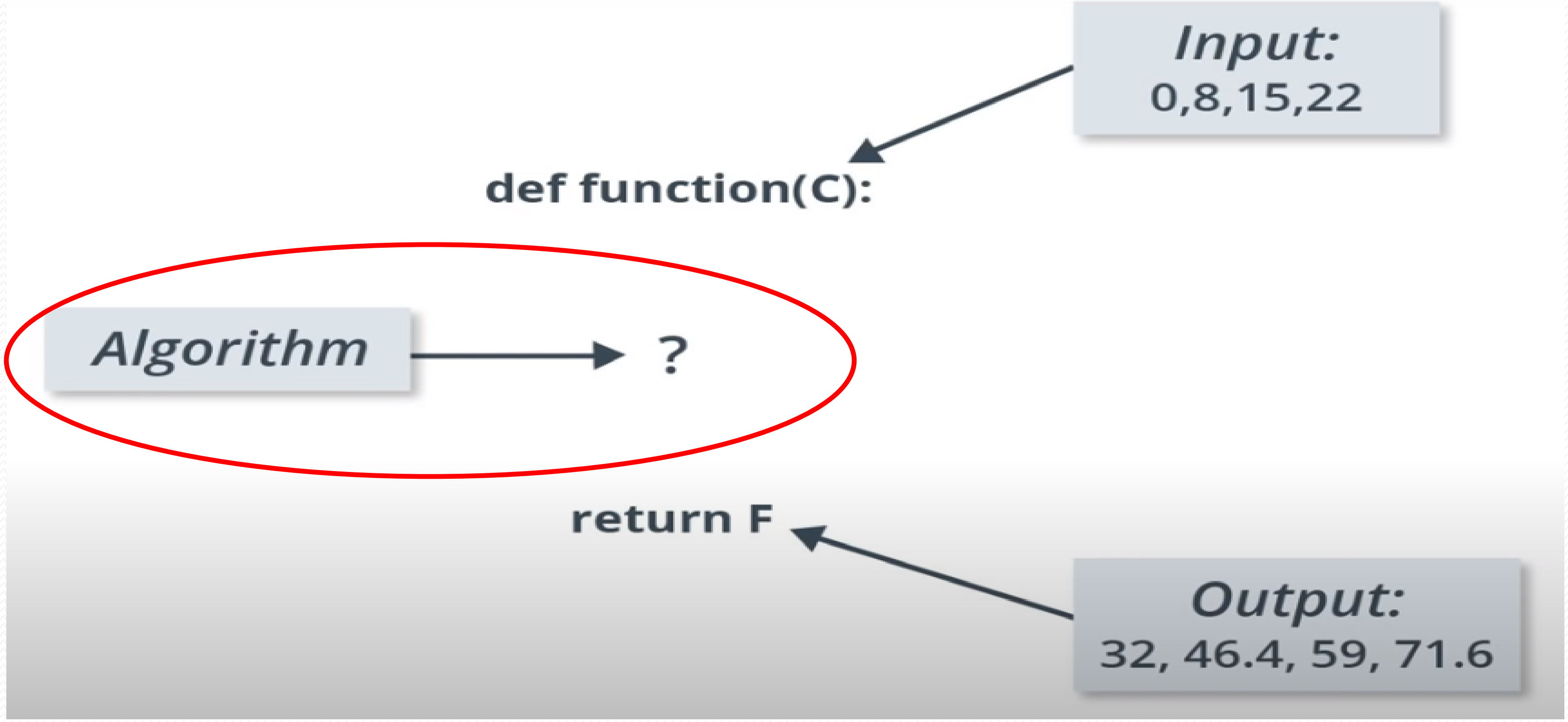
You know the input and the output, but you don't know the algorithm that creates the output given the input

- Take pairs of input and output data
- Figure out the algorithm

Traditional software Development



```
def function(C):
   F = C * 1.8 + 32
   return F
```



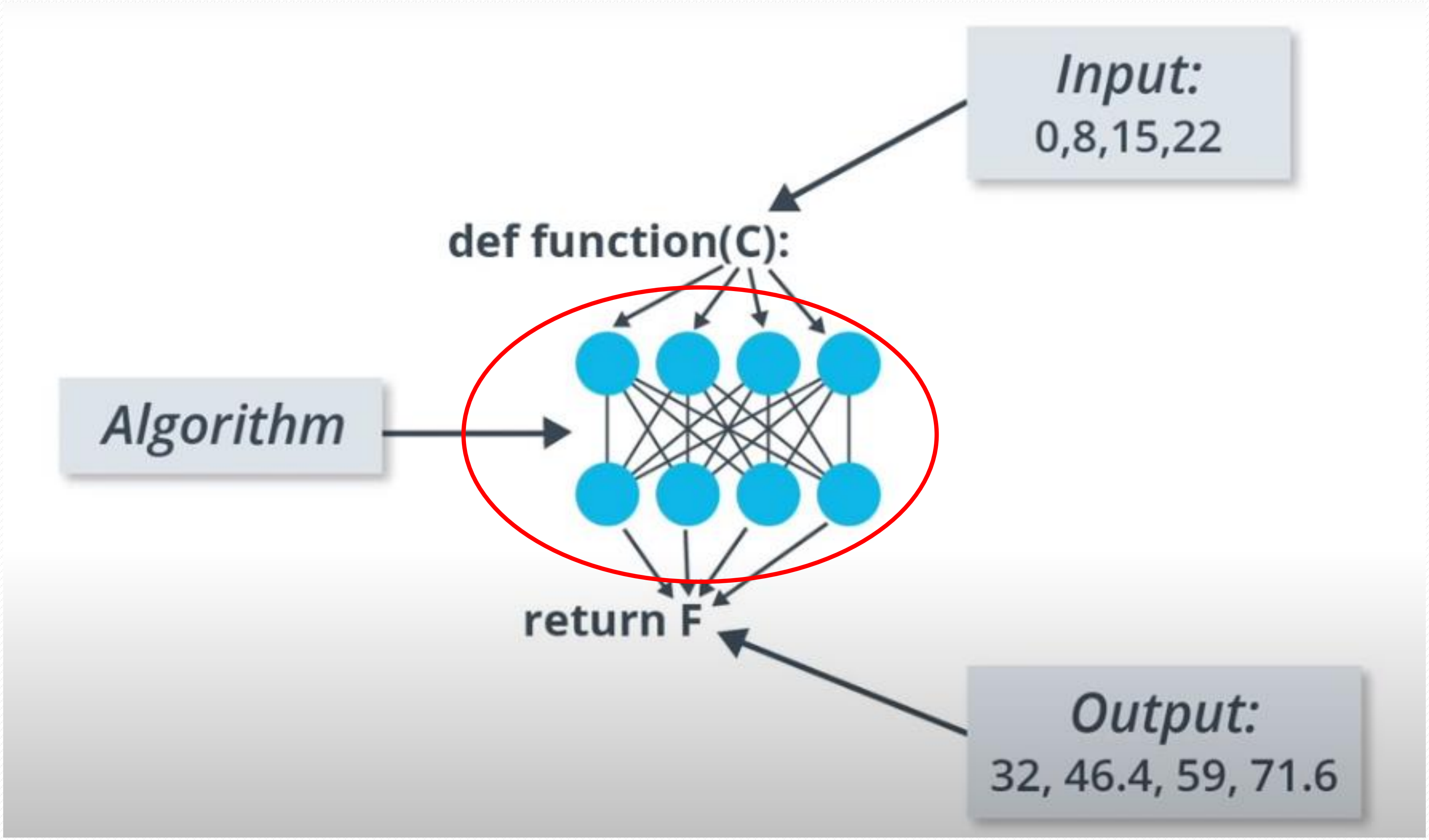
Input:
0,8,15,22

```
def function(C):
```

Algorithm → ?

```
return F
```

Output:
32, 46.4, 59, 71.6



Input:
0,8,15,22

def function(C):

Algorithm

return F

Output:
32, 46.4, 59, 71.6

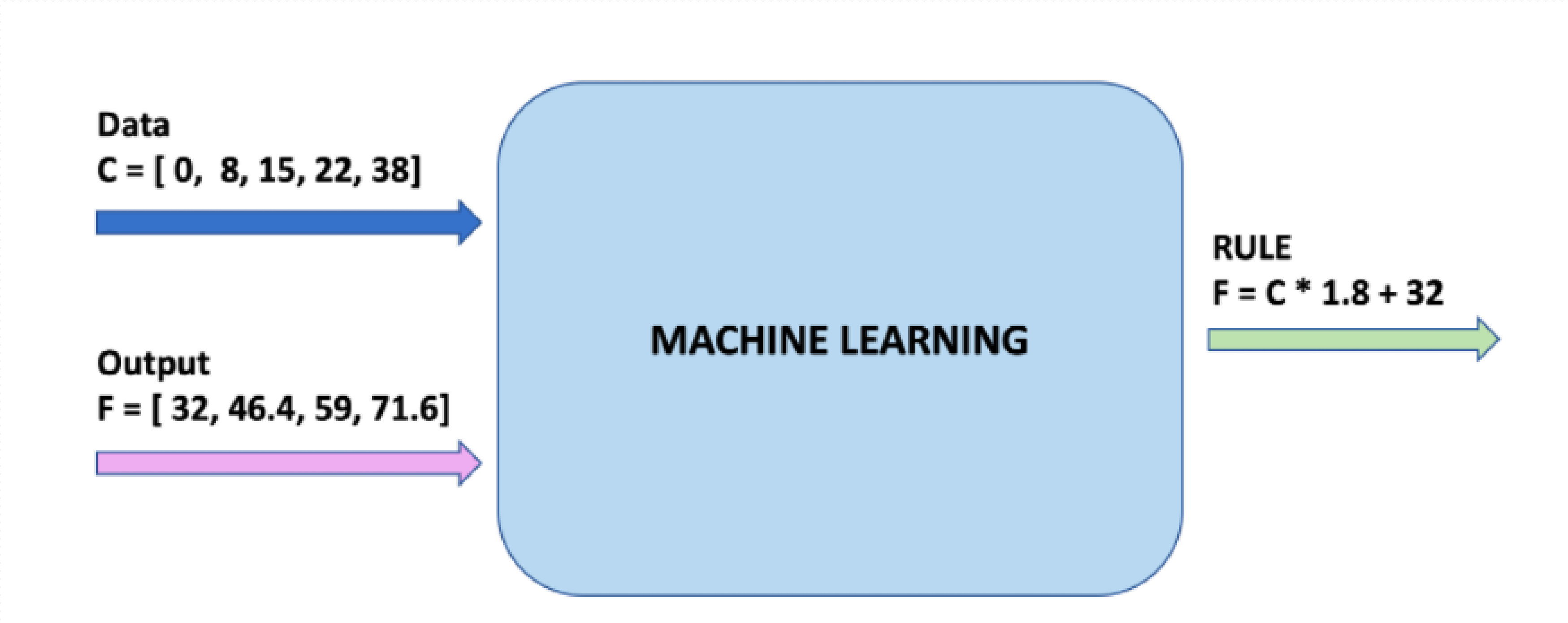
Training process

- The training process (happening in model.fit(…)).the math it contains (what calculations are being performed, and in what order) is not modified during training. Instead, it is the internal variables (“weights” and “biases”) which are updated during training.

- The model starts by multiplying the input by some number (the weight) and adding another number (the bias). Training the model involves finding the right values for these variables, not changing from multiplication and addition to some other operation.

To do machine learning

- Example.



Data

C = [0, 8, 15, 22, 38]

Output

F = [32, 46.4, 59, 71.6]

MACHINE LEARNING

RULE

F = C * 1.8 + 32

To do machine learning

- Gradient descent iteratively adjusts parameters, nudging them in the correct direction a bit at a time until they reach the best values. In this case "best values" means that nudging them any more would make the model perform worse. The function that measures how good or bad the model is during each iteration is called the "loss function", and the goal of each nudge is to "minimize the loss function."

- The training process starts with a forward pass, where the input data is fed to the neural network. Then the model applies its internal math on the input and internal variables to predict an answer

Forward pass

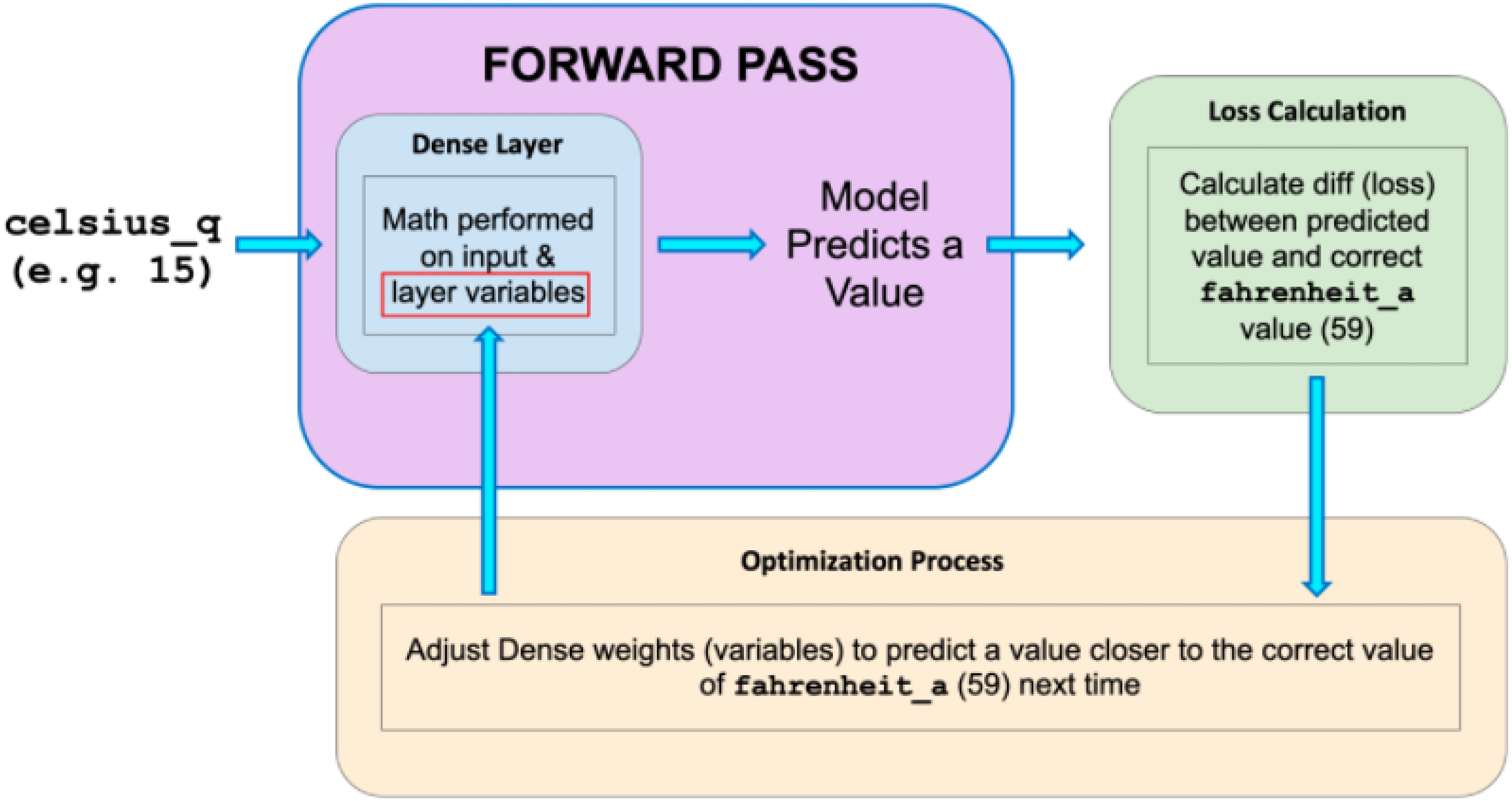
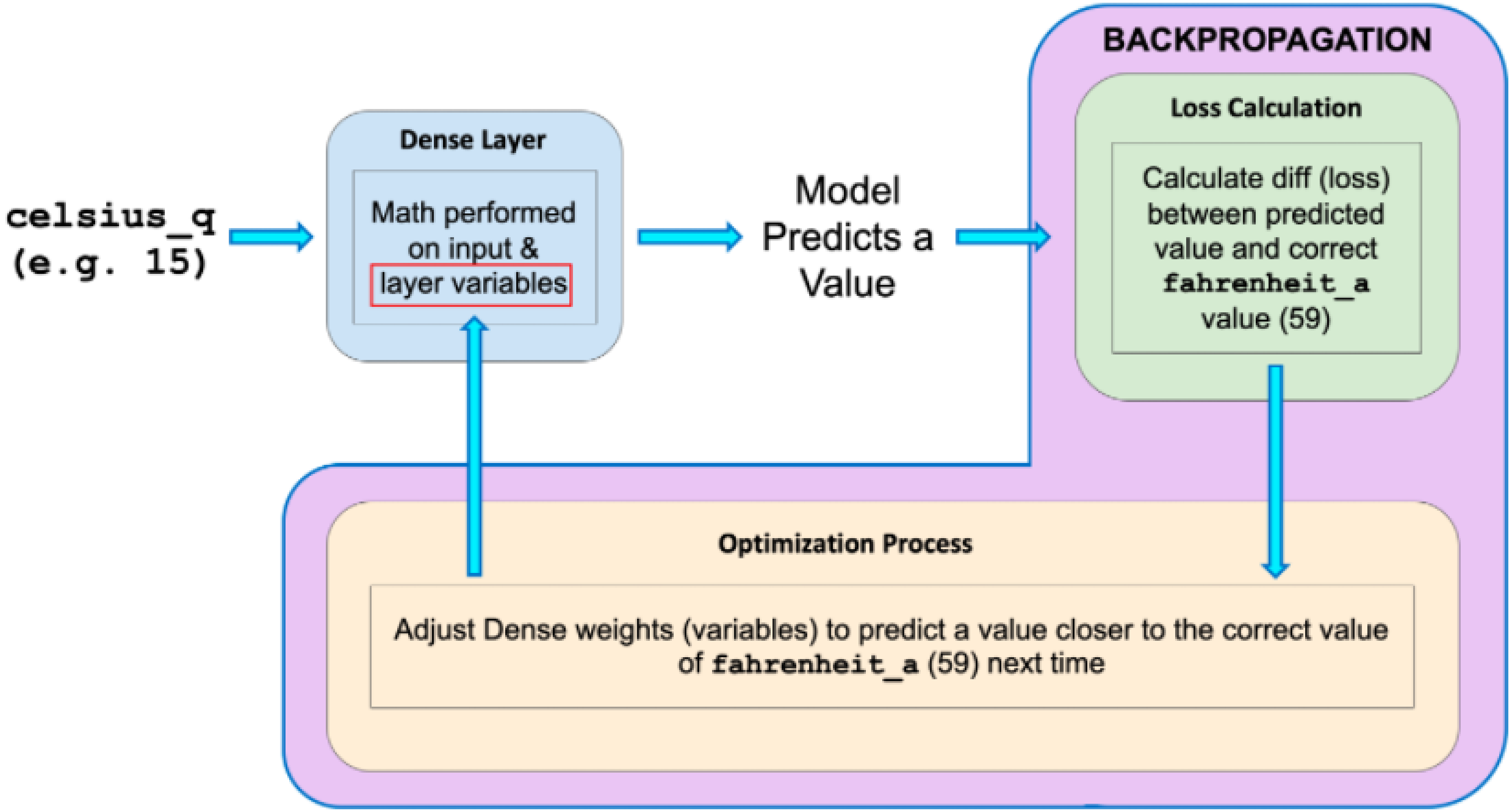

FORWARD PASS
Dense Layer
Math performed on input & layer variables
celsius_q (e.g. 15)
Model Predicts a Value
Loss Calculation
Calculate diff (loss) between predicted value and correct fahrenheit_a value (59)
Optimization Process
Adjust Dense weights (variables) to predict a value closer to the correct value of fahrenheit_a (59) next time

Figure 1. Forward Pass

Forward pass

- Once a value is predicted, the difference between that predicted value and the correct value is calculated. This difference is called the loss, and it's a measure of how well the model performed the mapping task. The value of the loss is calculated using a loss function, which we specified with the loss parameter when calling model.compile().

backpropagation



`celsius_q`
`(e.g. 15)`

Dense Layer

Math performed on input & layer variables

Model Predicts a Value

BACKPROPAGATION

Loss Calculation

Calculate diff (loss) between predicted value and correct **`fahrenheit_a`** value (59)

Optimization Process

Adjust Dense weights (variables) to predict a value closer to the correct value of **`fahrenheit_a`** (59) next time

Figure 2. Backpropagation

Activate Windows

backpropagation

- After the loss is calculated, the internal variables (weights and biases) of all the layers of the neural network are adjusted, so as to minimize this loss — that is, to make the output value closer to the correct value.

- This optimization process is called Gradient Descent. The specific algorithm used to calculate the new value of each internal variable is specified by the optimizer parameter when calling model.compile(...). In this example we used the Adam optimizer.

ML terms

- **Feature:** The input(s) to our model
- **Examples:** An input/output pair used for training
- **Labels:** The output of the model
- **Layer:** A collection of nodes connected together within a neural network.
- **Model:** The representation of your neural network
- **Dense and Fully Connected (FC):** Each node in one layer is connected to each node in the previous layer.
- **Weights and biases:** The internal variables of model
- **Loss:** The discrepancy between the desired output and the actual output

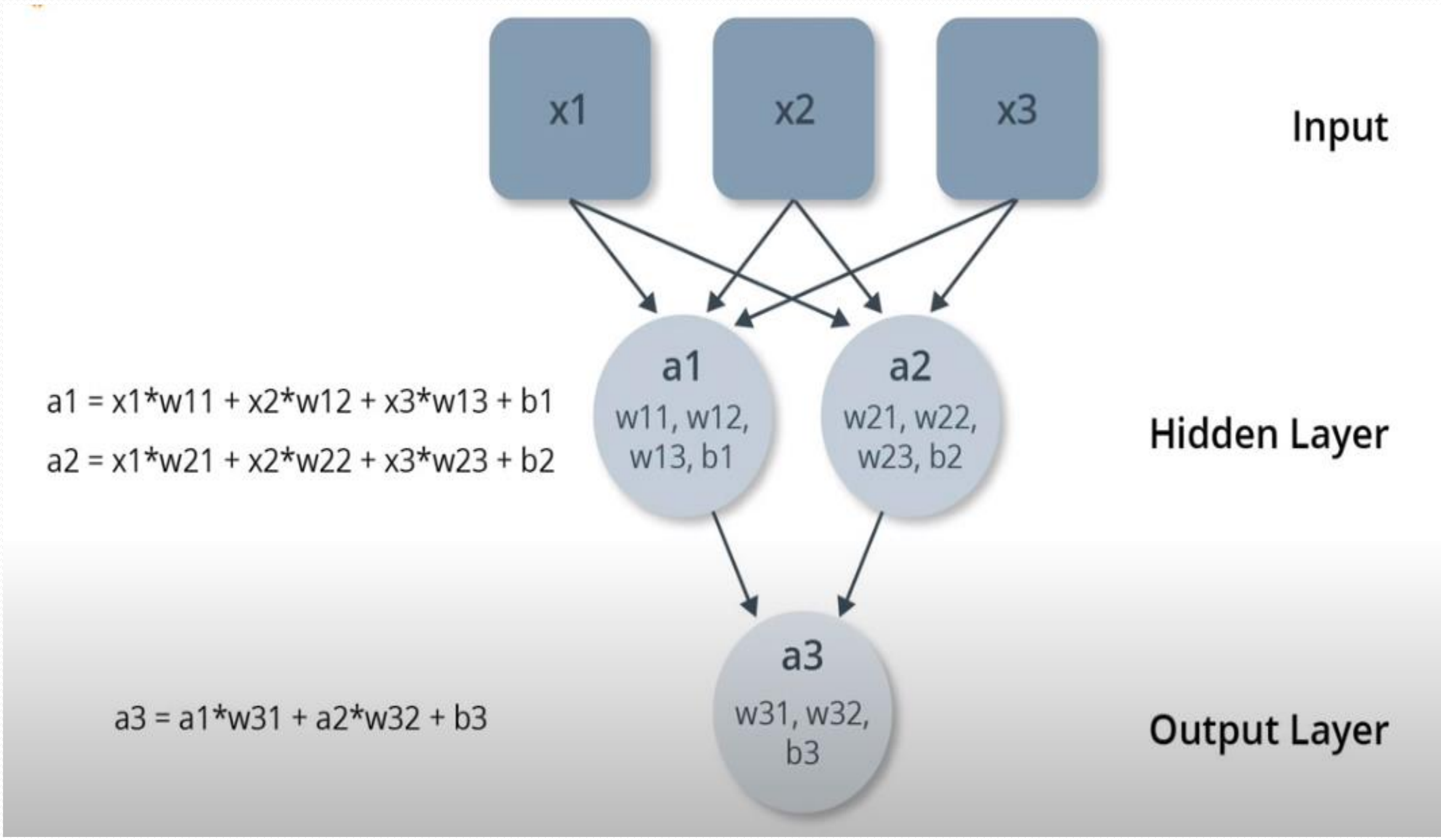
ML terms(cont..)

- **MSE:** Mean squared error, a type of loss function that counts a small number of large discrepancies as worse than a large number of small ones.
- **Gradient Descent:** An algorithm that changes the internal variables a bit at a time to gradually reduce the loss function.
- **Optimizer:** A specific implementation of the gradient descent algorithm. (There are many algorithms for this. In this course we will only use the "Adam" Optimizer, which stands for *ADAptive with Momentum*. It is considered the best-practice optimizer.)
- **Learning rate:** The "step size" for loss improvement during gradient descent.

ML terms(cont..)

- **Batch:** The set of examples used during training of the neural network
- **Epoch:** A full pass over the entire training dataset
- **Forward pass:** The computation of output values from input
- **Backward pass (backpropagation):** The calculation of internal variable adjustments according to the optimizer algorithm, starting from the output layer and working back through each layer to the input.

Dense layer: fully connected layer



a1 = x1*w11 + x2*w12 + x3*w13 + b1

a2 = x1*w21 + x2*w22 + x3*w23 + b2

a3 = a1*w31 + a2*w32 + b3

- ***Code:***

- what’s the general form of most ML models.

let’s see how to predict water temperature from water salinity.