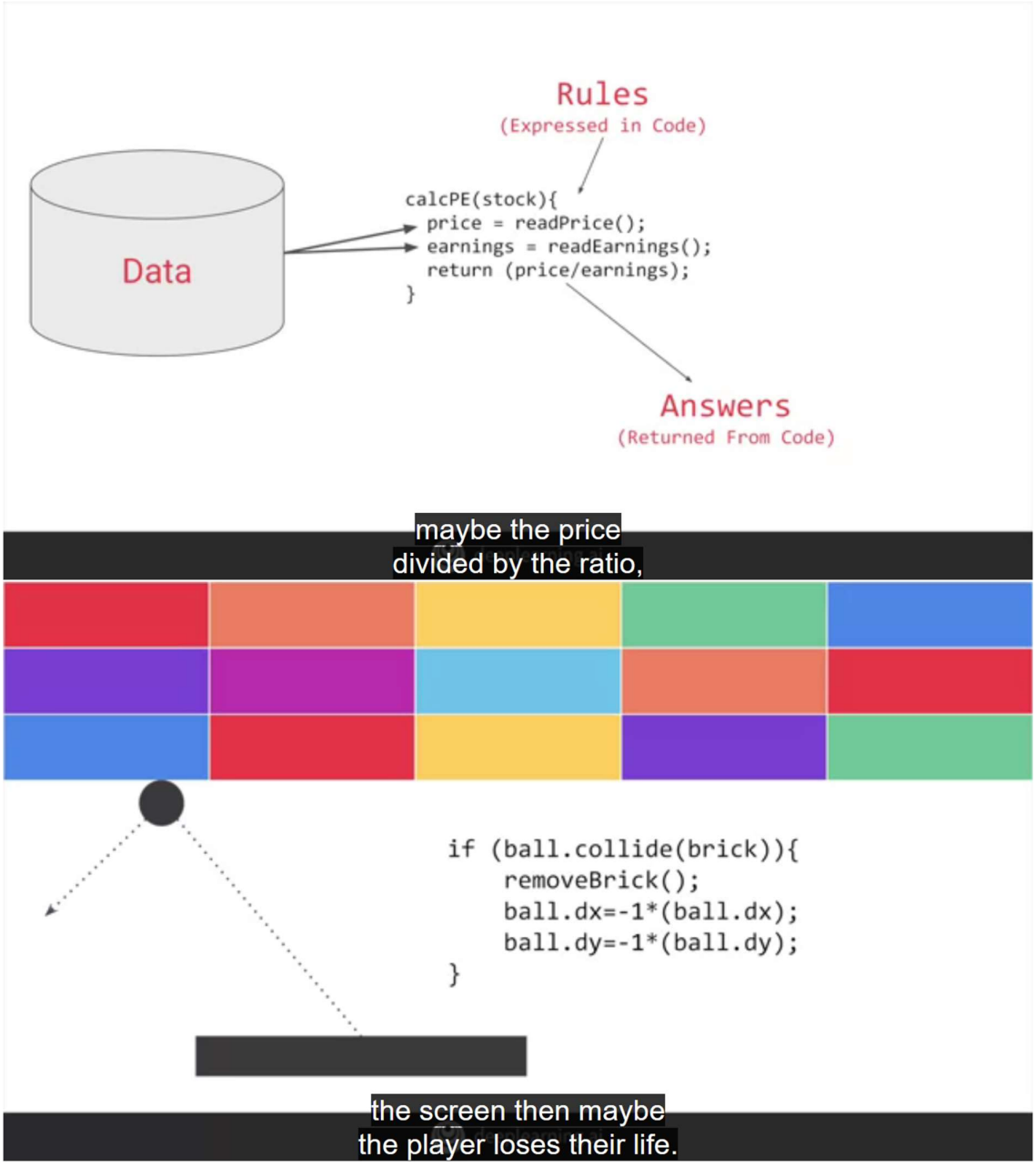
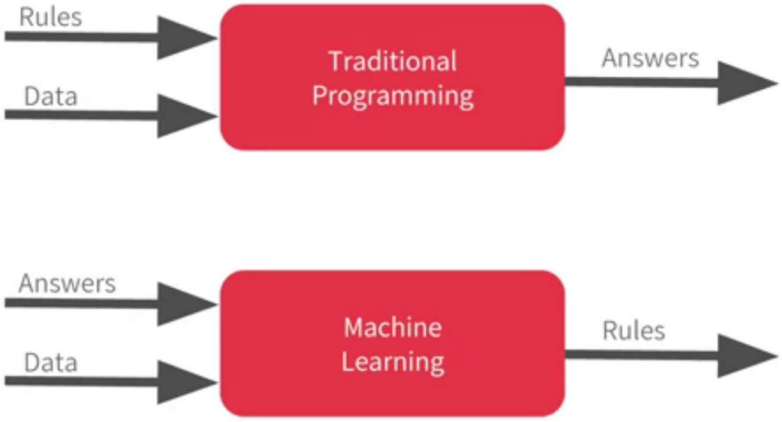

Rules
(Expressed in Code)
Data
calcPE(stock){
price = readPrice();
earnings = readEarnings();
return (price/earnings);
}
Answers
(Returned From Code)
maybe the price
divided by the ratio,
if (ball.collide(brick)){
removeBrick();
ball.dx=-1*(ball.dx);
ball.dy=-1*(ball.dy);
}
the screen then maybe
the player loses their life.

a programming language
and data can come from



where we put answers in data

Activity Recognition

```
if(speed<4){
    status=WALKING;
}
```

```
if(speed<4){
    status=WALKING;
} else {
    status=RUNNING;
}
```

```
if(speed<4){
    status=WALKING;
} else if(speed<12){
    status=RUNNING;
} else {
    status=BIKING;
}
```

```
// Oh crap
```

downhill and other people

Activity Recognition

```
01010010101001010101
10010101010010111101
01001010100101010010
01010010101001010101
```

Label = WALKING

```
10101001010010101010
01010100100100010001
00100111110101011111
10101001001111010111
```

Label = RUNNING

```
10010100111110101010
11010101110101011100
10101011110101010111
11111100011110101010
```

Label = BIKING

```
11111111101001110100
00111110101111101010
01011101010101011100
10101010100111110
```

Label = GOLFING
(Sort of)

figures out the
specific patterns in

X = -1, 0, 1, 2, 3, 4

Y = -3, -1, 1, 3, 5, 7

Can you spot it? Take a moment.

```
model = keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
```

Successive layers are
defined in sequence,

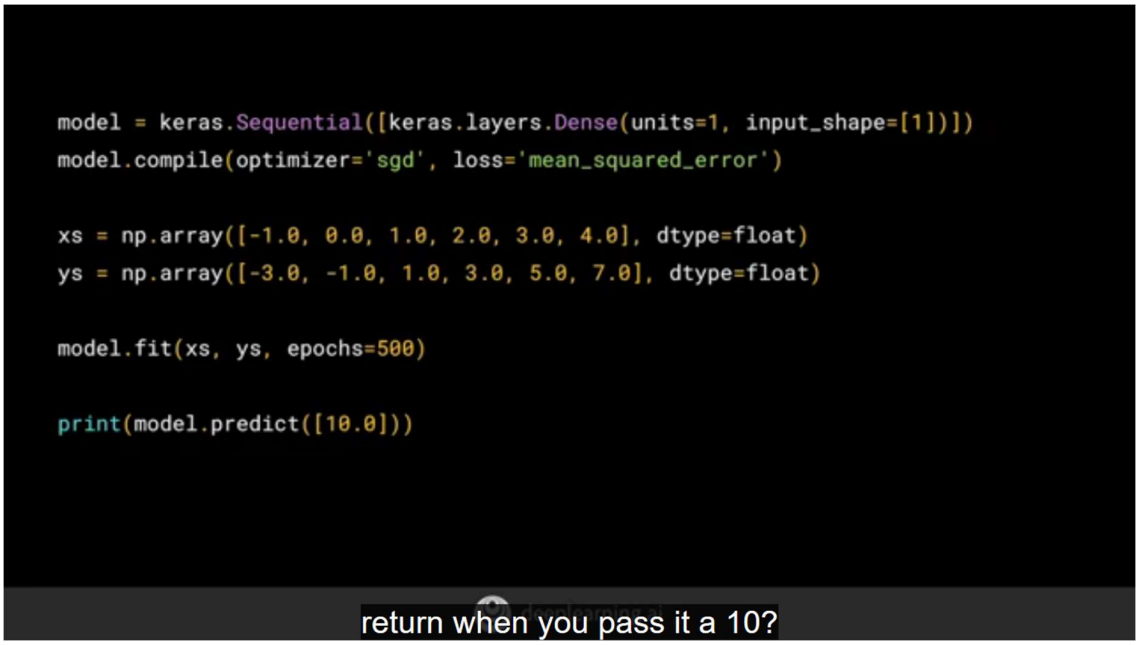
```
model = keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')
```

The neural network has

```
model = keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)
```

particularly enlists much easier.


model = keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')
xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)
model.fit(xs, ys, epochs=500)
print(model.predict([10.0]))
return when you pass it a 10?