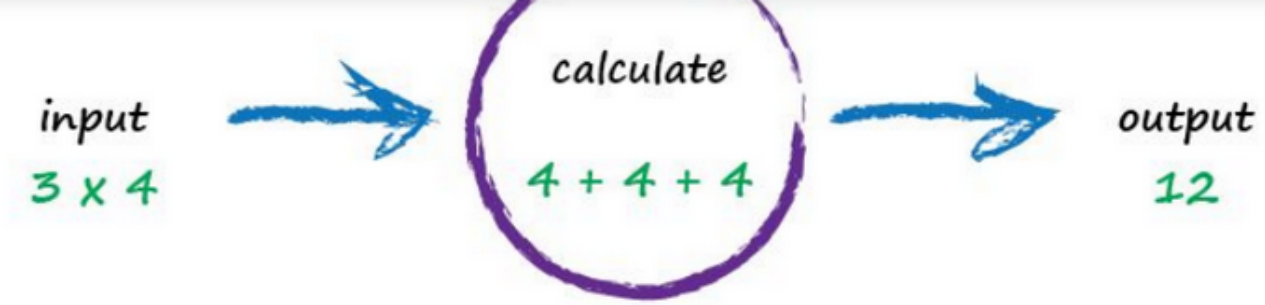
A Simple Predicting Machine

Q1: how do computers think?

We should know that they dont think they just process large amount of data

While we humans do the thinking part very well but calculation not very much

Computer are very good at doing arithmetics. so something like 4+4+4 = 12 sort of calculation



Now lets take an example to demonstrate this

Example 1

Truth Example	Kilometres	Miles
1	0	0
2	100	62.137

Take the example of miles and kilometers, they have a linear relation between them. so we know that

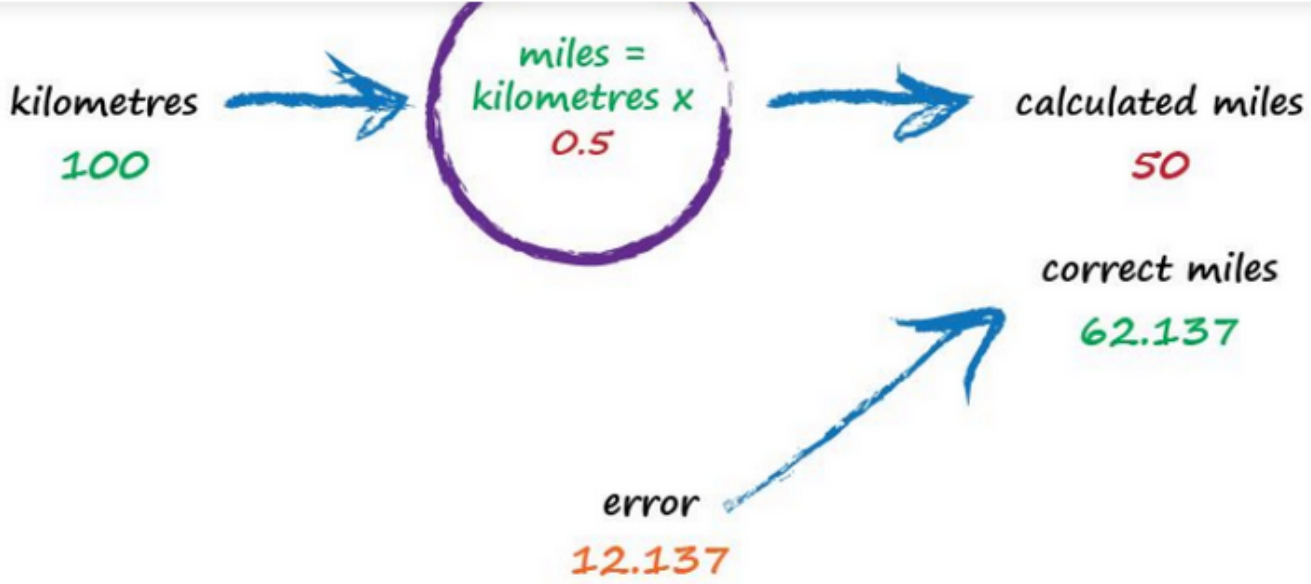
miles = kilometers * c where c is some constant

We guess that our constant c is 0.5 and calculate the result it turns out to be 50

error = desired - calculated

error = 62.137 - 50

error = 12.137



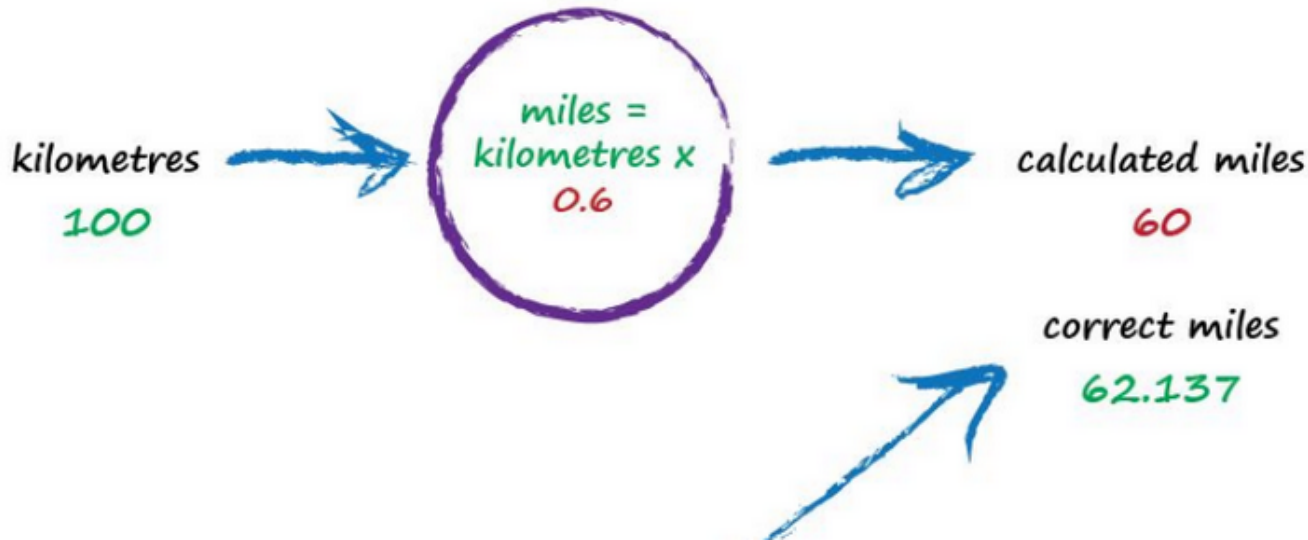
Lets give it another try

We guess that our constant c is 0.6 and calculate the result it turns out to be 60

error = desired - calculated

error = 62.137 - 60

error = 2.137



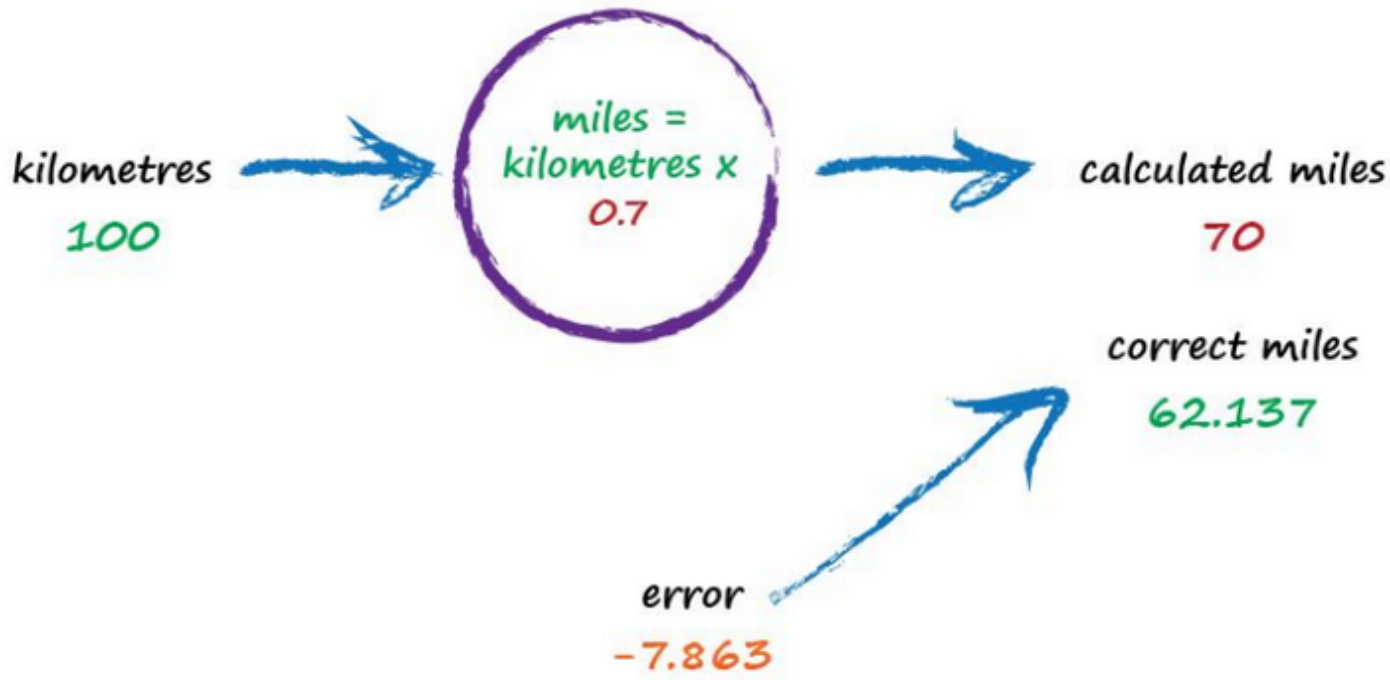
We see from this example that the error is getting progressively smaller

lets see what happens when we take c to be 0.7

error = desired - calculated

error = 62.137 - 70

error = -7.863



So we get the idea that a negative value means we have overshot, thus leading us to conclusion that the closer we get to lowering the error the smaller the increments

Classifying is Not too Different from Predicting

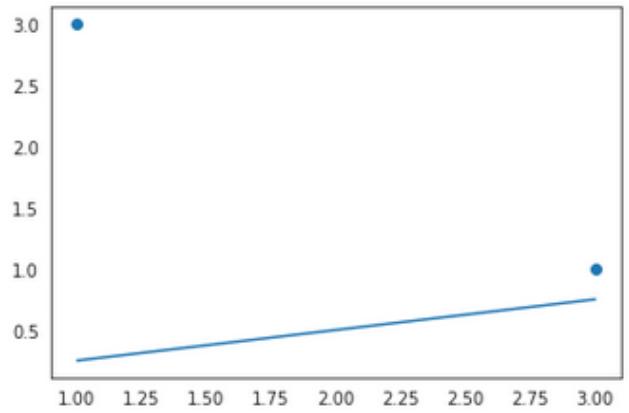
Example 2

In [63]:
```
import matplotlib.pyplot as plt
import numpy as np

x=np.array([1,3])
y=np.array([3,1])
```

Example	Width	Length	Bug
1	3.0	1.0	ladybird
2	1.0	3.0	caterpillar

In [57]:
```
plt.scatter(x,y)

#add line of best fit to plot
plt.plot(x, 0.25*x)
```

Out[57]:
```
[<matplotlib.lines.Line2D at 0x7f74fc39dc10>]
```



Notice Our Equation is similar to y = mx +c. In fact its the same, we just don't need to make the line pass through the origin.What will the error be in this case?

y = (0.25)*1 = 0.75

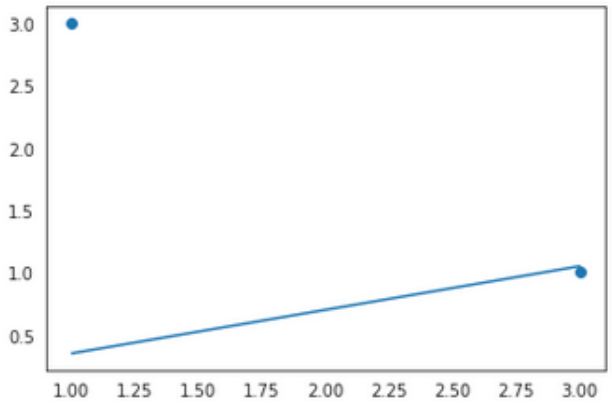
Thats our current slope but we dont want to adjust the error to 1 because we are not making a simple predicting machine but an actual classifier.So we will aim a little higher lets say 1.1 we dont want to aim to high, thus causing it to go above both the ladybird and caterpillar.

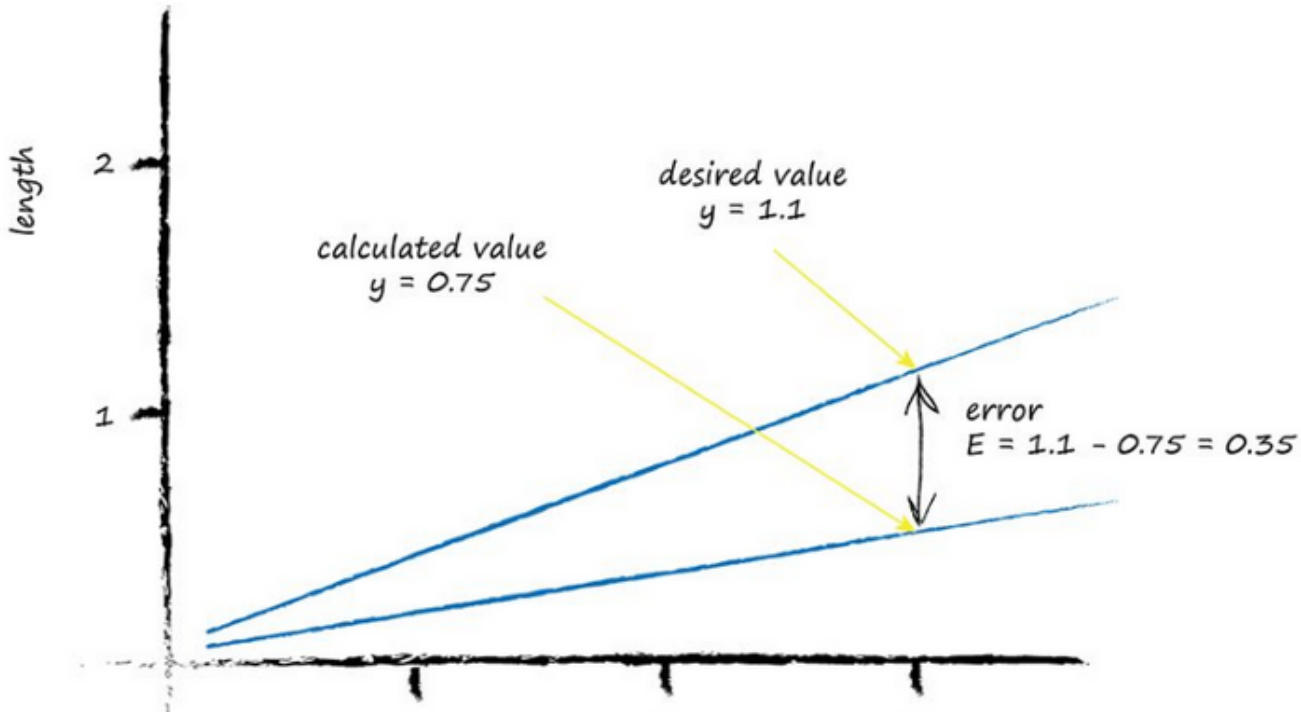
E = desired - original = 1.1 - 0.75 = 0.35

Now, we cant just kee manually adjusting the slope, we have to find an automatic way what we call an algorithm. First we will have to find the relationship between the error and the slope.

In [60]:
```
plt.scatter(x,y)

#add line of best fit to plot
plt.plot(x, 0.35*x)
```

Out[60]:
```
[<matplotlib.lines.Line2D at 0x7f74fcb47310>]
```



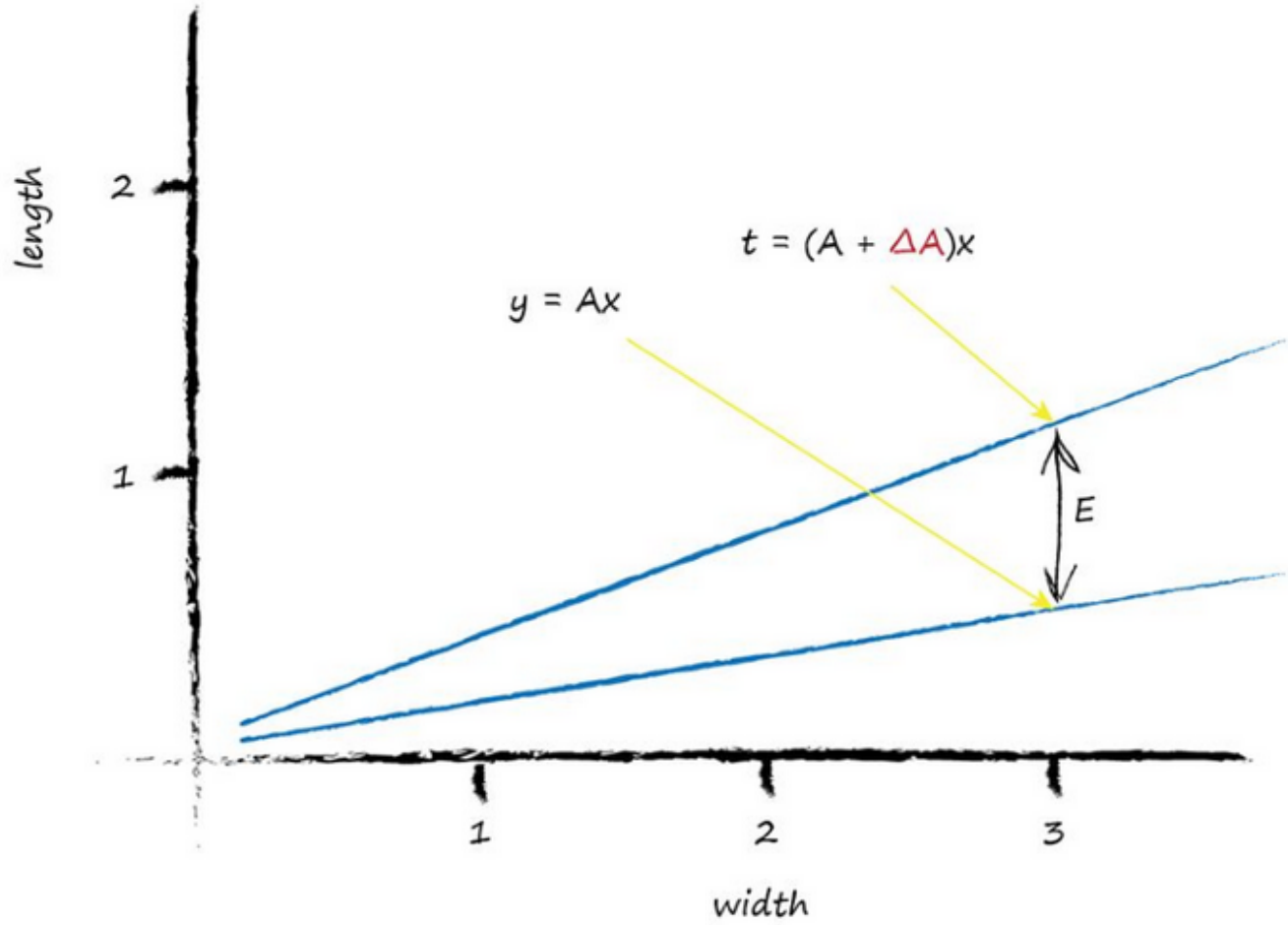


To find the relationship, let start with the linear equation

$$y = Ax$$

The desired target t will be some increment to A denoted by ΔA.

$$t = (A + \Delta A)x$$



We know that

```
E = t - y = (A + ΔA)x - Ax
E = ΔAx
```

Wow, the relationship is that simple, now all we need is to calculate the adjusted slope based on the error by

```
ΔA = E/x
```

So, in our example above, our adjusted slope will be

```
ΔA = Error / desired = 0.35 / 3.0 = 0.1667
```

We can then just add the value to our original slope by

```
y = (A + ΔA) = (0.25 + 0.1667) * x = 0.3667 * 3 = 1.1
```

Conclusion:

- We learned about how a simple Predictor is made.
- We learned how a predictor can become a classifier.
- An algorithm to adjust error automatically for linear relationships.