Preface

This is a short bit about functions. If you're confident in your knowledge of functions, skip on ahead!

Actually, let's not think about functions for a minute; instead, let's think about something fantastic.

Once upon a time, there existed a mysterious race of cats that live forever.. Maybe they live forever because they're really a drawing of my roommate's plushy.

Anyway, this one cat in particular **wasn't given a name** by his parents. He was a young and wild and free spirit, and he was special in that **if you gave him food, he would meow at you**.

This cat travelled all over the world, and the people he met there **gave him names**. When he swung round the north pole....

...they called him Claus.

When he trekked through America...

....they called him Abe.

But whether he was known as Abe or Claus or known by no name at all, he was still the same cat. The same cat that would accept food and **return** a meow.

Functions

If you haven't realized yet, our immortal cat was an analogy of a **function**.

He starts out like a **lambda** function: nameless.

```
lambda food: meow
```

This function, although having no name, accepts **food** as a parameter, and **returns a meow**.

Even though immortal-cat has no name, he can still meow for us:

```
(lambda food: meow)(cat_food) #returns a meow
```

This passes **cat_food** into the **food** parameter, and then **returns** a meow.

When our immortal cat swung by the north pole, they named him Claus.

```
claus = lambda food : meow
```

Now, we can call Claus by name, and give him food that way.

```
1 claus = lambda food : meow
2 claus(cat_food) #returns a meow
```

In America, where they're a bit more advanced, they called him Abe. But they did so in a different way:

```
1 def abe(food):
2     return meow
```

Our immortal cat accepts **food** as a parameter, and **returns a meow.** His name is now abe. But it's still (basically) the same cat.

```
1 def abe(food):
2     return meow
3
4 abe(cat_food) #returns a meow
```

Concretely, functions are **objects just like any other**. We can name them, rename them, compare them, and even **return** them. That's an important foundation for functional programming.