Sliceline: order a pizza (*free!*)

A CRUD CLI Program by Lief Friedrichs

Code I found Challenging

JOINER TABLES: it took me a while to understand how to implement many-to-many with ActiveRecord

```
class Topping < ActiveRecord::Base
    has_many :pizza_toppings
    has_many :pizzas, through: :pizza_toppings
end
```

```
class PizzaTopping < ActiveRecord::Base
    belongs_to :pizza
    belongs_to :topping
end
```

```
class Pizza < ActiveRecord::Base
    has_many :orders
    has_many :pizza_toppings
    has_many :toppings, through: :pizza_toppings
```

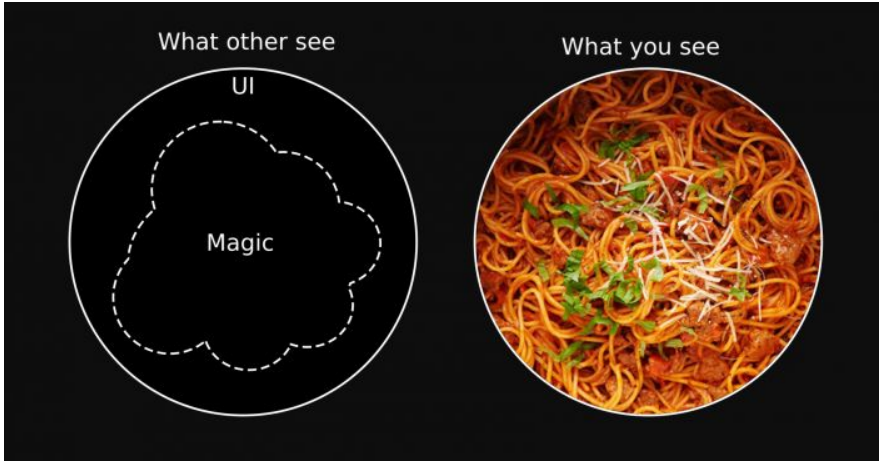
```
class Order < ActiveRecord::Base
    belongs_to :user
    has_one :pizza
```

Things I Wish I Knew

1. Calling a function as a parameter for another function will evaluate that function
    a. Function(parameter, function2) -> function 2 will run when Function is called

2. Write CLEAR and WELL NAMED CODE from the beginning

3. Completely define your scope, goals, and desired methods BEFORE you start coding



Sliceline 2.0

For version 2.0:

I would add the option for a user to create an order with more than one kind of pizza

I would add the option for a user to have multiple orders

I would have a defined list of toppings and a menu of pizzas

A Selection of Code to Chew on

```
def run
   banner = Api.new.get_programs.colorize(:light_blue)
   puts "\n\n#{banner}\n\n"
   str = "Sliceline".colorize(:red)
   puts "\n\nWelcome to #{str}, the best way to order pizza!\n\n"
   Cli.do_you_have_an_order(nil)
End

def self.goodbye(user)
   if user != nil
      puts "Goodbye #{user.name},"
   end
   str = "Sliceline".colorize(:red)
   puts "\nThanks for using #{str}!\n\n"
end
```

The code loops back to the first promt, but the user may have already given their name. By defining *user = nil* inside of *def run*, the memory of the user giving their name is kept by the program. The same idea was implemented for the user's order after one is created or found.