exercism

Armstrong Numbers


Narcissist
Number

Armstrong Number

Sum of its own digits

Raised to the power of the number of its digits

Armstrong Numbers

Armstrong Numbers

✅ 9

Armstrong Numbers

✅ 9 = 9^1 = 9

Armstrong Numbers

✅ 9 = 9^1 = 9

❌ 10

Armstrong Numbers

✅ 9 = 9^1 = 9

❌ 10 != 1^2 + 0^2 = 1

Armstrong Numbers

✅ 153 = 1^3 + 5^3 + 3^3 = 153

Armstrong Numbers

✅ 153 = 1^3 + 5^3 + 3^3 = 153

❌ 154 != 1^3 + 5^3 + 4^3 = 190

Armstrong Numbers

Armstrong Numbers

🔢 Get the number of digits

Armstrong Numbers

- 🔢 Get the number of digits
- 🔢 Calculate the power of each digit

Armstrong Numbers

- 🔢 Get the number of digits
- 🔢 Calculate the power of each digit
- 🔢 Sum all the powers

Take 1

Get number of digits

Get number of digits

```
module ArmstrongNumbers
  module_function

  def include?(number)





  end
end
```

```
ArmstrongNumbers.include?(154)
```

Get number of digits

```
module ArmstrongNumbers
  module_function

  def include?(number)





  end
end
```

Get number of digits

```
module ArmstrongNumbers
  module_function

  def include?(number)
    number




  end
end
```

Get number of digits

```
module ArmstrongNumbers
  module_function

  def include?(number)
    number
      .digits




  end
end
```

Sum Powers

```
module ArmstrongNumbers
  module_function

  def include?(number)
    number
      .digits
      .then(&method(:sum_powers))



  end
end
```

Sum Powers

```
module ArmstrongNumbers
  module_function

  def include?(number)
    sum =
      number
      .digits
      .then(&method(:sum_powers))


  end
end
```

Sum Powers

```
module ArmstrongNumbers
  module_function

  def include?(number)
    sum =
      number
      .digits
      .then(&method(:sum_powers))

    sum == number
  end
end
```

Sum Powers

```
module ArmstrongNumbers
  module_function
  # ...

  def sum_powers(digits)




  end
  private_class_method :sum_powers
end
```

Sum Powers

```
module ArmstrongNumbers
  module_function
  # ...

  def sum_powers(digits)
    digits




  end
  private_class_method :sum_powers
end
```

Sum Powers

```
module ArmstrongNumbers
  module_function
  # ...

  def sum_powers(digits)
    digits
      .each



  end
  private_class_method :sum_powers
end
```

Sum Powers

```
module ArmstrongNumbers
  module_function
  # ...

  def sum_powers(digits)
    digits
      .each
      .with_object(digits.length)

  end
  private_class_method :sum_powers
end
```

Sum Powers

```
module ArmstrongNumbers
  module_function
  # ...

  def sum_powers(digits)
    digits
      .each
      .with_object(digits.length)
      .sum(&method(:power))
  end
  private_class_method :sum_powers
end
```

Calculate Power of Each Digit

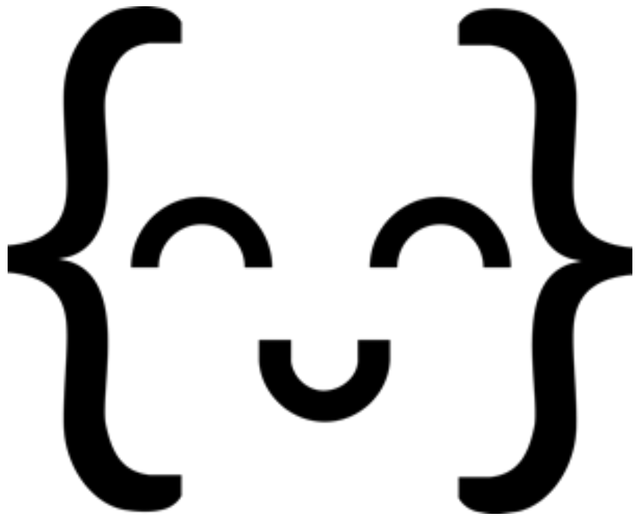
```
module ArmstrongNumbers
  module_function
  # ...

  def power((digit, length))
    digit**length
  end
  private_class_method :power
end
```

```ruby
module ArmstrongNumbers
  module_function

  def include?(number)
    sum =
      number
      .digits
      .then(&method(:sum_powers))

    sum == number
  end

  def sum_powers(digits)
    digits
      .each
      .with_object(digits.length)
      .sum(&method(:power))
  end
  private_class_method :sum_powers

  def power((digit, length))
    digit**length
  end
  private_class_method :power
end
```

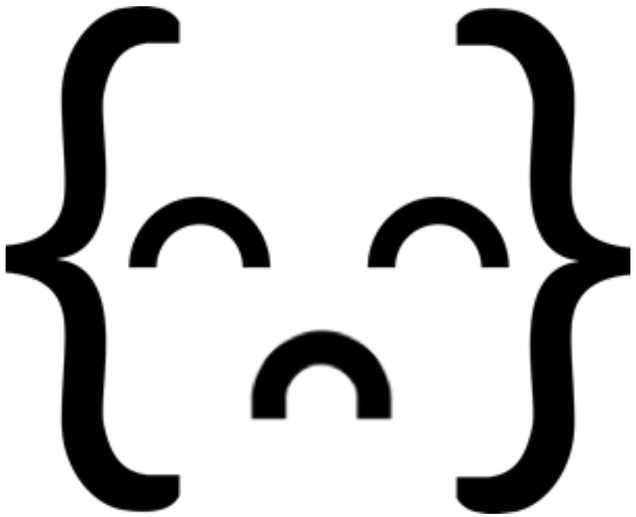
```
18:19:29 - INFO - Running: all tests
Run options: --guard --seed 33059

# Running:

.........

Finished in 0.020076s, 448.2965 runs/s, 448.2965 assertions/s.
9 runs, 9 assertions, 0 failures, 0 errors, 0 skips
18:19:29 - INFO - Inspecting Ruby code style: armstrong_numbers.rb
Inspecting 1 file
.

1 file inspected, no offenses detected
[1] guard(main)>
```

exercism

```
exercism submit armstrong_numbers.rb
```



Done!



Done?

Complex…?

Take 2

```ruby
module ArmstrongNumbers
  module_function

  def include?(number)



  end
end
```

```ruby
module ArmstrongNumbers
  module_function

  def include?(number)
    digits = number.digits



  end
end
```

```ruby
module ArmstrongNumbers
  module_function

  def include?(number)
    digits = number.digits
    length = digits.length

  end
end
```

```ruby
module ArmstrongNumbers
  module_function

  def include?(number)
    digits = number.digits
    length = digits.length
    number == digits.sum { |digit| digit**length }
  end
end
```

```
18:19:29 - INFO - Running: all tests
Run options: --guard --seed 33059

# Running:

.........

Finished in 0.020076s, 448.2965 runs/s, 448.2965 assertions/s.
9 runs, 9 assertions, 0 failures, 0 errors, 0 skips
18:19:29 - INFO - Inspecting Ruby code style: armstrong_numbers.rb
Inspecting 1 file
.

1 file inspected, no offenses detected
[1] guard(main)>
```

Choose Your Own Adventure

You'll rarely get it right the first time

Submit often, get feedback

Read other people's code

Refactor until you're happy

github.com/paulfioravanti/exercism

BONUS FACTOIDS!

BONUS FACTOIDS!

🔢 Number of Armstrong numbers is finite

BONUS FACTOIDS!

- Number of Armstrong numbers is finite
- Only 88 Armstrong numbers in Base 10

115,132,219,018,
763,992,565,095,
597,973,971,522,401

```
irb(main):001:0> require './armstrong_numbers'
true
irb(main):002:0> ArmstrongNumbers.include?(115132219018763992565095597973971522401)
true
irb(main):003:0>
```

Thanks!

@paulfioravanti