Ruby Strings

String interpolation and syntax

Coder Factory Academy

Strings are flexible

- Strings are one of the most flexible types
- They support a wide range of alphabets from around the world, using the Unicode standard
- Strings can be converted into numbers, arrays, and symbols
- These types can also be converted back into a string using the method **.to_s**

Interpolation

- Interpolation is used to combine several values into one string
- You use double quoted strings, and insert a value using **#{ … }**
- The value is interpreted as Ruby, and the entire result becomes a String

```ruby
year = 2017
message = "Welcome to #{year}"
# => "Welcome to 2017"

easy_maths = "1 + 1 = #{ 1 + 1 }"
# => "1 + 1 = 2"

hard_maths = "The square root of 500
is #{ Math.sqrt(500) }"
# => "The square root of 500 is
22.360679774997898"

name = gets.chomp
hello = "Hello, #{ name }"
# => Hello, Queen Elizabeth II
```

Single vs double quotes

- Double quotes support interpolation
- Single quote do not, they literally take their contents
- Prefer single quotes unless interpolating, as it says ‘this is a simple string’

```
year = 2017
message = "Welcome to #{year}"
# => "Welcome to 2017"

message_failed = 'Welcome to
#{year}'
# => "Welcome to #{year}"
```

Converting to numbers

- Strings can be converted into numbers
- You can convert to a whole number (Integer) or real number (Float)
- For integers, use **.to_i**
- For floats, use **.to_f**

```
'109'.to_i # => 109
'3.14'.to_i # => 3
'0.5'.to_i # => 0
'101 Dalmatians'.to_i # => 101
'Dog'.to_i # => 0

'109'.to_f # => 109.0
'3.14'.to_f # => 3.14
'0.5'.to_f # => 0.5
'101 Dalmatians'.to_i # => 101.0
'Dog'.to_i # => 0.0
```

Converting case

- Strings can easily be converted to either uppercase or lowercase
- Use these methods:
  - Uppercase: **.upcase**
  - Lowercase: **.downcase**

```
'The quick brown fox jumps over the lazy dog'.upcase
# => "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG"

'The quick brown fox jumps over the lazy dog'.downcase
# => "the quick brown fox jumps over the lazy dog"
```

Arrays + Strings

- A string can be separated into chunks using the **.split** method
- This produces an array of strings
- An array of strings can be joined together into one using **.join**

```
names = 'Alice, John, Lucy'.split(', ')
# => ['Alice', 'John', 'Lucy']

names.join('_')
# => "Alice_John_Lucy"

names.join('_').downcase
# => "alice_john_lucy"
```

Trimming whitespace

- Whitespace — spaces, tabs **\t**, and various new lines **\n \r**
- It can be useful to trim this from strings, especially user input
- **.strip** trims whitespace from beginning and end
- **.chomp** trims new lines from end

```
'   Some text   '.strip
# => "Some text"

input = gets
# => "Entered text\n"

input = gets.chomp
# => "Entered text"
```

Decorative formatting

- You can repeat a string using *
- You can add decoration using **.center**
- You can add to the beginning or end of a string using **.prepend** or **.concat**

```
'=' * 15
# => "==============="

'Welcome'.center(30, '=')
# => "===========Welcome============"

'Welcome'.prepend('  ').concat('
').center(30, '#')
# => "########## Welcome ###########"
```