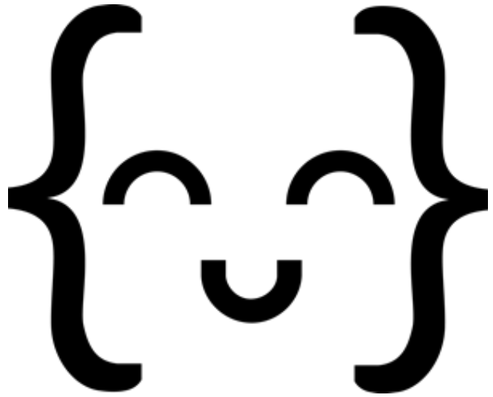
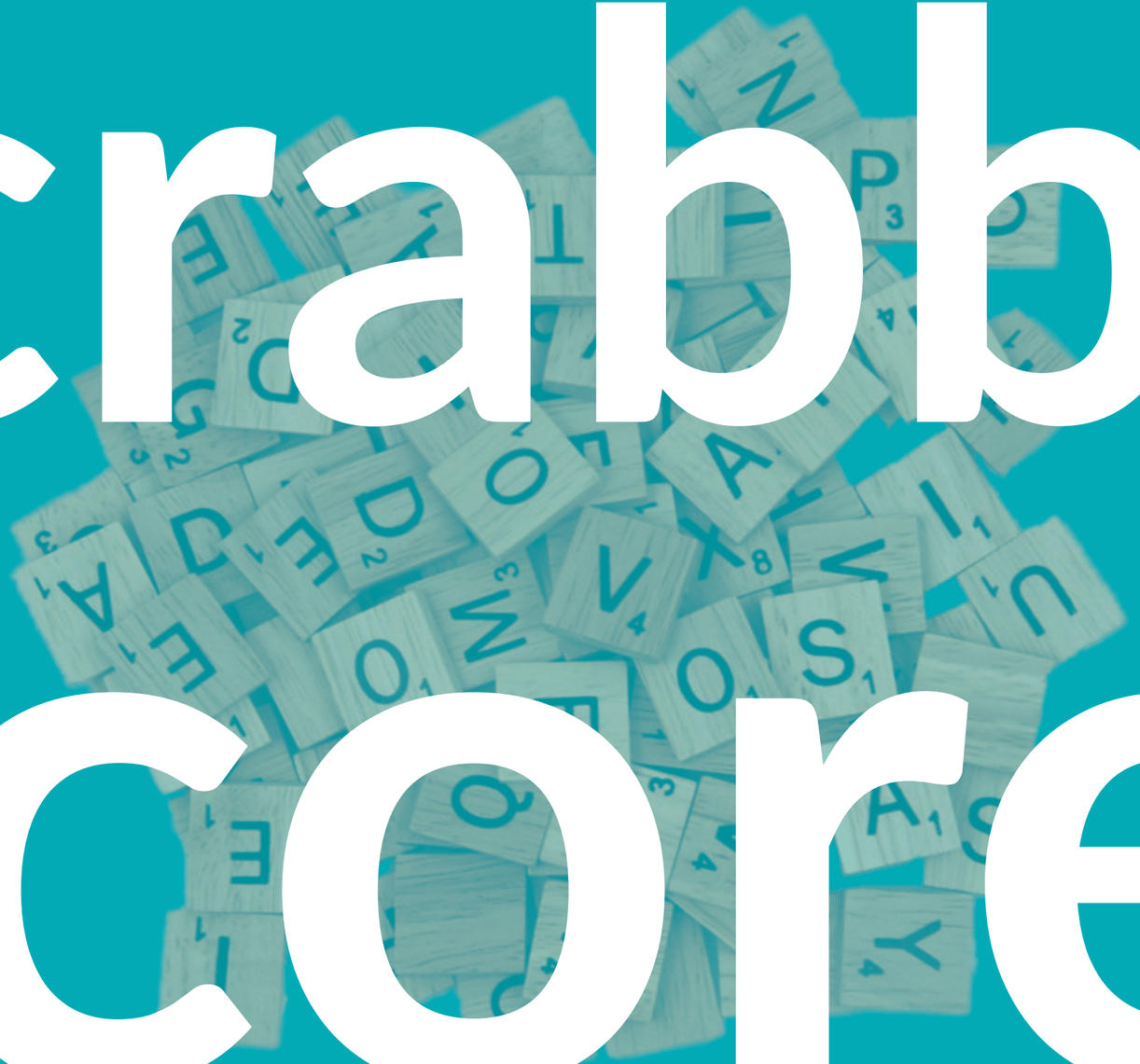


exercism

Scrabble Score

Scrabble Scorer



Scrabble Scorer

Scrabble Scorer

🔡 Score a letter

Scrabble Scorer

- 🔤 Score a letter
- 🔤 Score a word

Scrabble Scorer

- 🔤 Score a letter
- 🔤 Score a word
- 🔤 Score case-insensitively

Scrabble Scorer

- 🔡 Score a letter
- 🔡 Score a word
- 🔡 Score case-insensitively
- 🔡 Score non-words

Take 1

Create Score Mapping

```
defmodule Scrabble
  @scores %{
    "a" => 1, "b" => 3, "c" => 3,
    "d" => 2, "e" => 1, "f" => 4,
    # ...
    "x" => 8, "y" => 4, "z" => 10
  }
  # ...
end
```

Create Score Function

```
defmodule Scrabble
  @scores %{...}

  def score(word) do
    # ...
  end
end
```

Create Score Function

```
defmodule Scrabble
  @scores %{...}

  def score("elixir") do
    # ...
  end
end
```

Create Score Function

```
defmodule Scrabble
  @scores %{...}

  def score(word) do
    # ...
  end
end
```

Create Score Function

```
defmodule Scrabble
  @scores %{...}

  def score(word) do
    word




  end
end
```

Case-Insensitive Scoring

```
defmodule Scrabble
  @scores %{...}

  def score(word) do
    word
    |> String.downcase()



  end
end
```

Create List of Letters

```
defmodule Scrabble
  @scores %{...}

  def score(word) do
    word
    |> String.downcase()
    |> String.split("", trim: true)


  end
end
```

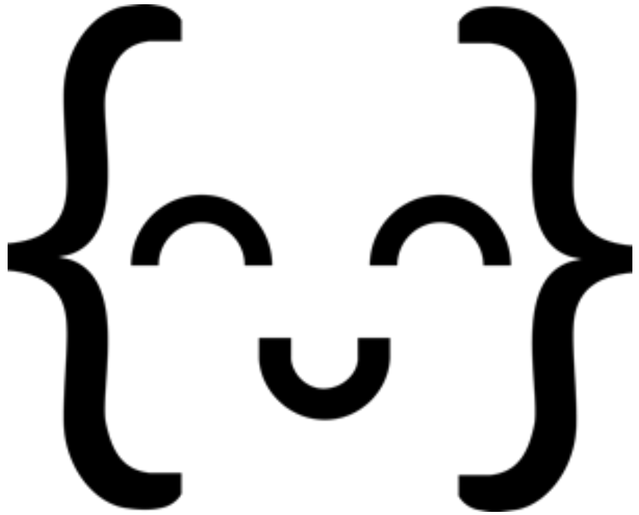
Score Letters with Fallback

```
defmodule Scrabble
  @scores %{...}

  def score(word) do
    word
    |> String.downcase()
    |> String.split("", trim: true)
    |> Enum.map(fn char -> Map.get(@scores, char, 0) end)

  end
end
```

Sum all Scores

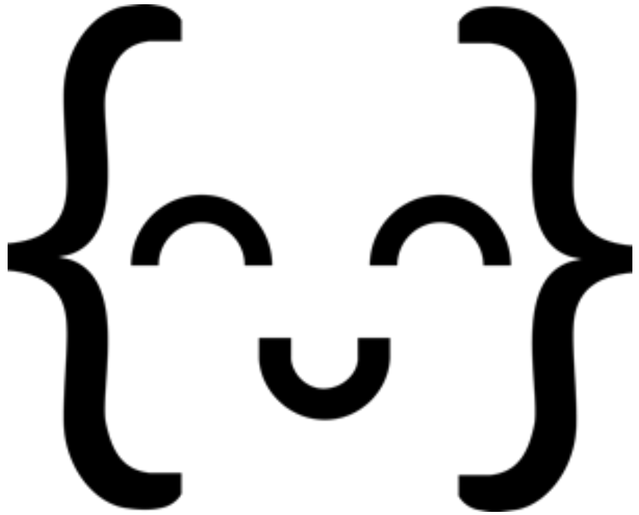
```
defmodule Scrabble
  @scores %{...}

  def score(word) do
    word
    |> String.downcase()
    |> String.split("", trim: true)
    |> Enum.map(fn char -> Map.get(@scores, char, 0) end)
    |> Enum.sum()
  end
end
```

```
➜ [scrabble-score (master) ✗]$ elixir scrabble_score_test.exs
Excluding tags: [:pending]


ScrabbleScoreTest
  * test convenient scoring (11.8ms)
  * test scores very short word (0.01ms)
  * test complicated word scores more (0.01ms)
  * test empty word scores zero (0.00ms)
  * test simple word scores the number of letters (0.01ms)
  * test scores other very short word (0.00ms)
  * test whitespace scores zero (0.00ms)
  * test scores are case insensitive (0.02ms)


Finished in 0.05 seconds (0.04s on load, 0.01s on tests)
8 tests, 0 failures

Randomized with seed 825627
```

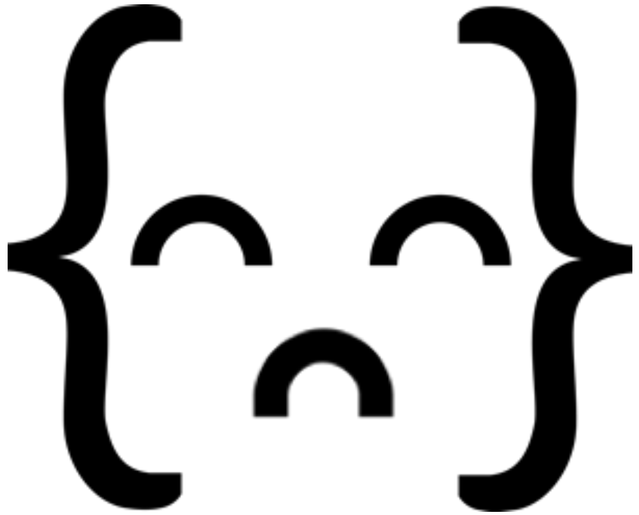


exercism

```
exercism submit scrabble.exs
```



Done!



Done?

Take 2

Scrabble Scorer

```
defmodule Scrabble
  @scores %{...}

  def score(word) do
    word
    |> String.downcase()
    |> String.split("", trim: true)
    |> Enum.map(fn char -> Map.get(@scores, char, 0) end)
    |> Enum.sum()
  end
end
```

Better List Conversion?

```
defmodule Scrabble
  @scores %{...}

  def score(word) do
    word
    |> String.downcase()
    |> String.split("", trim: true)
    |> Enum.map(fn char -> Map.get(@scores, char, 0) end)
    |> Enum.sum()
  end
end
```

Better List Conversion?

```
defmodule Scrabble
  @scores %{...}

  def score(word) do
    word
    |> String.downcase()
    |> String.codepoints()
    |> Enum.map(fn char -> Map.get(@scores, char, 0) end)
    |> Enum.sum()
  end
end
```

Better List Conversion?

```
defmodule Scrabble
  @scores %{...}

  def score(word) do
    word
    |> String.downcase()
    |> String.graphemes()
    |> Enum.map(fn char -> Map.get(@scores, char, 0) end)
    |> Enum.sum()
  end
end
```

```
iex> string = "\u0065\u0301"
"é"
```

```
iex> string = "\u0065\u0301"
"é"
iex> byte_size(string)
3
```

```
iex> string = "\u0065\u0301"
"é"
iex> byte_size(string)
3
iex> String.length(string)
1
```

```
iex> string = "\u0065\u0301"
"é"
```

```
iex> string = "\u0065\u0301"
"é"
iex> String.codepoints(string)
["e", "́"]
```

```
iex> string = "\u0065\u0301"
"é"
iex> String.codepoints(string)
["e", "́"]
iex> String.graphemes(string)
["é"]
```

Better List Conversion?

```
defmodule Scrabble
  @scores %{"a" => 1, "b" => 3, "c" => 3}

  def score(word) do
    word
    |> String.downcase()
    |> String.graphemes()
    |> Enum.map(fn char -> Map.get(@scores, char, 0) end)
    |> Enum.sum()
  end
end
```

Use Codepoints

```
defmodule Scrabble
  @scores %{?a => 1, ?b => 3, ?c => 3}

  def score(word) do
    word
    |> String.downcase()
    |> String.to_charlist()
    |> Enum.map(fn char -> Map.get(@scores, char, 0) end)
    |> Enum.sum()
  end
end
```

Multiple Traversal

```
defmodule Scrabble
  @scores %{?a => 1, ?b => 3, ?c => 3}

  def score(word) do
    word
    |> String.downcase()
    |> String.to_charlist()
    |> Enum.map(fn char -> Map.get(@scores, char, 0) end)
    |> Enum.sum()
  end
end
```

Multiple Traversal

```
defmodule Scrabble
  @scores %{?a => 1, ?b => 3, ?c => 3}

  def score(word) do
    word
    |> String.downcase()
    |> String.to_charlist()
    |> Stream.map(fn char -> Map.get(@scores, char, 0) end)
    |> Enum.sum()
  end
end
```

Multiple Traversal

```
defmodule Scrabble
  @scores %{?a => 1, ?b => 3, ?c => 3}

  def score(word) do
    word
    |> String.downcase()
    |> String.to_charlist()
    |> Enum.reduce(0, fn char, acc ->
      acc + Map.get(@scores, char, 0)
    end)
  end
end
```

Multiple Traversal

```
defmodule Scrabble
  @scores %{?a => 1, ?b => 3, ?c => 3}

  def score(word) do
    word
    |> String.downcase()
    |> String.to_charlist()
    |> Enum.reduce(0, &add_score/2)
  end

  defp add_score(char, acc) do
    acc + Map.get(@scores, char, 0)
  end
end
```

Scrabble Scorer

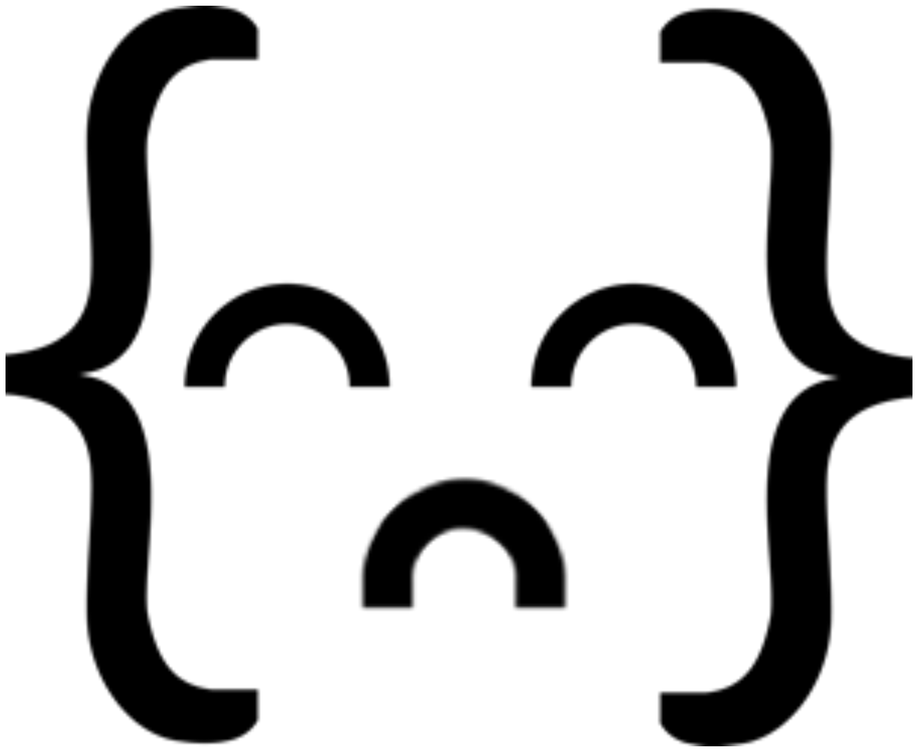
```
defmodule Scrabble
  @scores %{?a => 1, ?b => 3, ?c => 3}

  def score(word) do
    word
    |> String.downcase()
    |> String.to_charlist()
    |> Enum.reduce(0, &add_score/2)
  end

  defp add_score(char, acc) do
    acc + Map.get(@scores, char, 0)
  end
end
```

Scrabble Scorer

```
defmodule Scrabble
  @scores %{?a => 1, ?b => 3, ?c => 3}

  def score(word) do
    word
    |> String.downcase()
    |> String.to_charlist()
    |> Enum.reduce(0, &add_score/2)
  end

  defp add_score(char, acc) do
    acc + Map.get(@scores, char, 0)
  end
end
```

Huuuge Map

```
defmodule Scrabble
  @scores %{
    ?a => 1, ?b => 3, ?c => 3,
    ?d => 2, ?e => 1, ?f => 4,
    ?g => 2, ?h => 4, ?i => 1,
    ?j => 8, ?k => 5, ?l => 1,
    ?m => 3, ?n => 1, ?o => 1,
    ?p => 3, ?q => 10, ?r => 1,
    ?s => 1, ?t => 1, ?u => 1,
    ?v => 4, ?w => 4, ?x => 8,
    ?y => 4, ?z => 10
  }
end
```



Take 3

Huuuge Map

```
defmodule Scrabble
  @scores %{
    ?a => 1, ?b => 3, ?c => 3,
    ?d => 2, ?e => 1, ?f => 4,
    ?g => 2, ?h => 4, ?i => 1,
    ?j => 8, ?k => 5, ?l => 1,
    ?m => 3, ?n => 1, ?o => 1,
    ?p => 3, ?q => 10, ?r => 1,
    ?s => 1, ?t => 1, ?u => 1,
    ?v => 4, ?w => 4, ?x => 8,
    ?y => 4, ?z => 10
  }
end
```

Lists of Letter Types

```
defmodule Scrabble
  @one_point_letters [?a, ?e, ?i, ?o, ?u, ?l, ?n, ?r, ?s, ?t]
  @two_point_letters [?d, ?g]
  @three_point_letters [?b, ?c, ?m, ?p]
  @four_point_letters [?f, ?h, ?v, ?w, ?y]
  @five_point_letters [?k]
  @eight_point_letters [?j, ?x]
  @ten_point_letters [?q, ?z]
end
```

```
iex> [?a, ?b, ?c, ?d] == 'abcd'
true
```

Lists of Letter Types

```
defmodule Scrabble
  @one_point_letters [?a, ?e, ?i, ?o, ?u, ?l, ?n, ?r, ?s, ?t]
  @two_point_letters [?d, ?g]
  @three_point_letters [?b, ?c, ?m, ?p]
  @four_point_letters [?f, ?h, ?v, ?w, ?y]
  @five_point_letters [?k]
  @eight_point_letters [?j, ?x]
  @ten_point_letters [?q, ?z]
end
```

Lists of Letter Types

```
defmodule Scrabble
  @one_point_letters 'aeioulnrst'
  @two_point_letters 'dg'
  @three_point_letters 'bcmp'
  @four_point_letters 'fhvwy'
  @five_point_letters 'k'
  @eight_point_letters 'jx'
  @ten_point_letters 'qz'
end
```

Fix add_score/2

```
defmodule Scrabble
  @one_point_letters 'aeioulnrst'
  # ...

  def score(word) do
    # ...
    |> Enum.reduce(0, &add_score/2)
  end

  defp add_score(char, acc) do
    acc + Map.get(@scores, char, 0)
  end
end
```

Fix add_score/2

```
defmodule Scrabble
  @one_point_letters 'aeioulnrst'
  # ...

  def score(word) do
    # ...
    |> Enum.reduce(0, &add_score/2)
  end

  defp add_score(char, acc) do
    acc + Map.get(@scores, char, 0)
  end
end
```

Switch on char

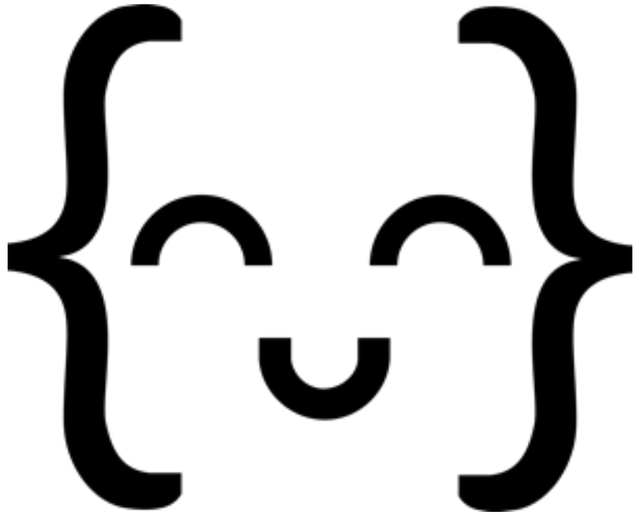
```
defmodule Scrabble
  defp add_score(char, acc) do
    case char do
      char when char in @one_point_letters ->
        acc + 1
      char when char in @two_point_letters ->
        acc + 2
      # ...
      char when char in @ten_point_letters ->
        acc + 10
      _ ->
        acc
    end
  end
end
```

Convert to Function Heads

```
defmodule Scrabble
  def score(word) do
    # ...
    |> Enum.reduce(0, &add_score/2)
  end

  defp add_score(char, acc) when char in @one_point_letters, do: acc + 1
  defp add_score(char, acc) when char in @two_point_letters, do: acc + 2
  defp add_score(char, acc) when char in @three_point_letters, do: acc + 3
  defp add_score(char, acc) when char in @four_point_letters, do: acc + 4
  defp add_score(char, acc) when char in @five_point_letters, do: acc + 5
  defp add_score(char, acc) when char in @eight_point_letters, do: acc + 8
  defp add_score(char, acc) when char in @ten_point_letters, do: acc + 10
  defp add_score(_char, acc), do: acc
end
```

Add Readable Guards

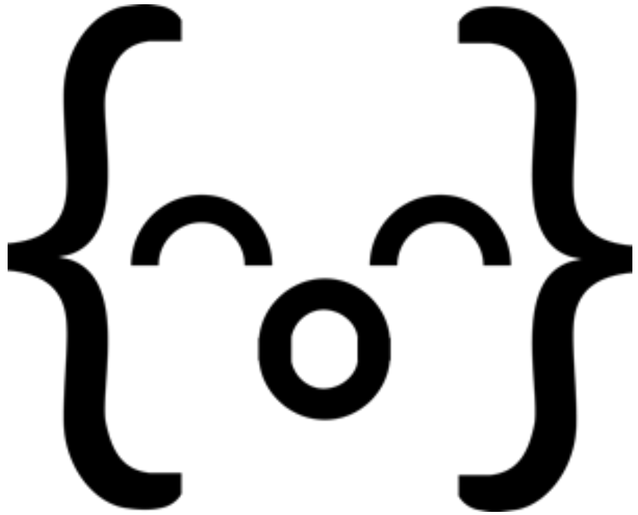
```
defmodule Scrabble
  defguardp one_point(char) when char in @one_point_letters
  # ...
  defguardp ten_points(char) when char in @ten_point_letters

  def score(word) do
    # ...
    |> Enum.reduce(0, &add_score/2)
  end

  defp add_score(char, acc) when one_point(char), do: acc + 1
  # ...
  defp add_score(char, acc) when ten_points(char), do: acc + 10
  defp add_score(_char, acc), do: acc
end
```



Done!

```elixir
defmodule Scrabble do
  @scores %{
    "a" => 1, "b" => 3, "c" => 3, "d" => 2, "e" => 1, "f" => 4,
    "g" => 2, "h" => 4, "i" => 1, "j" => 8, "k" => 5, "l" => 1,
    "m" => 3, "n" => 1, "o" => 1, "p" => 3, "q" => 10, "r" => 1,
    "s" => 1, "t" => 1, "u" => 1, "v" => 4, "w" => 4, "x" => 8,
    "y" => 4, "z" => 10
  }

  @doc """
  Calculate the scrabble score for the word.
  """
  @spec score(String.t()) :: non_neg_integer
  def score(word) do
    word
    |> String.downcase()
    |> String.split("", trim: true)
    |> Enum.map(fn char -> Map.get(@scores, char, 0) end)
    |> Enum.sum()
  end
end
```

```elixir
defmodule Scrabble do
  @one_point_letters 'aeioulnrst'
  @two_point_letters 'dg'
  @three_point_letters 'bcmp'
  @four_point_letters 'fhvwy'
  @five_point_letters 'k'
  @eight_point_letters 'jx'
  @ten_point_letters 'qz'

  defguardp one_point(letter) when letter in @one_point_letters
  defguardp two_points(letter) when letter in @two_point_letters
  defguardp three_points(letter) when letter in @three_point_letters
  defguardp four_points(letter) when letter in @four_point_letters
  defguardp five_points(letter) when letter in @five_point_letters
  defguardp eight_points(letter) when letter in @eight_point_letters
  defguardp ten_points(letter) when letter in @ten_point_letters

  @doc """
  Calculate the scrabble score for the word.
  """
  @spec score(String.t()) :: non_neg_integer
  def score(word) do
    word
    |> String.downcase()
    |> String.to_charlist()
    |> Enum.reduce(0, &add_score_for_letter/2)
  end

  defp add_score_for_letter(letter, acc) when one_point(letter), do: acc + 1
  defp add_score_for_letter(letter, acc) when two_points(letter), do: acc + 2
  defp add_score_for_letter(letter, acc) when three_points(letter), do: acc + 3
  defp add_score_for_letter(letter, acc) when four_points(letter), do: acc + 4
  defp add_score_for_letter(letter, acc) when five_points(letter), do: acc + 5
  defp add_score_for_letter(letter, acc) when eight_points(letter), do: acc + 8
  defp add_score_for_letter(letter, acc) when ten_points(letter), do: acc + 10
  defp add_score_for_letter(_letter, acc), do: acc
end
```



Great!

But...?

Map vs List Lookup Time

From

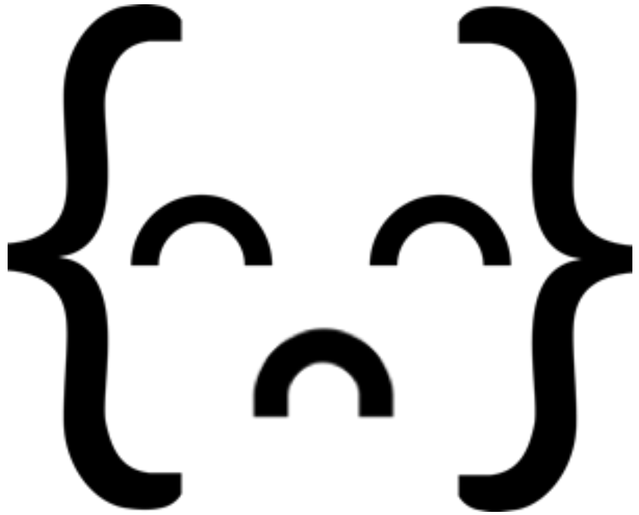
```
@scores %{?a => 1, ?b => 3}
Map.get(@scores, char, 0)
```

From

```
@scores %{?a => 1, ?b => 3}
Map.get(@scores, char, 0)
```

To

```
@one_point_letters 'aeioulnrst'
char when char in @one_point_letters
```



Slower..?

{-_-}

Benchee!

```
##### With input empty #####
Name                                          ips        average  deviation         median         99th %
function head refactor (codepoint list)      3.97 M      251.68 ns   ±956.65%          0 ns        1000 ns
Reduce refactor (codepoint map)              3.96 M      252.78 ns  ±1181.44%          0 ns        1000 ns
case refactor (codepoint list)               3.95 M      253.11 ns   ±937.02%          0 ns        1000 ns
guard refactor (codepoint list)              3.94 M      254.08 ns  ±1015.42%          0 ns        1000 ns
First attempt (string map)                   2.31 M      432.03 ns  ±7615.54%          0 ns        1000 ns
Stream refactor (string map)                 1.39 M      720.55 ns  ±3838.17%       1000 ns        1000 ns

Comparison:
function head refactor (codepoint list)      3.97 M
Reduce refactor (codepoint map)              3.96 M - 1.00x slower +1.09 ns
case refactor (codepoint list)               3.95 M - 1.01x slower +1.42 ns
guard refactor (codepoint list)              3.94 M - 1.01x slower +2.40 ns
First attempt (string map)                   2.31 M - 1.72x slower +180.34 ns
Stream refactor (string map)                 1.39 M - 2.86x slower +468.87 ns

Memory usage statistics:

Name                                   Memory usage
function head refactor (codepoint list)          0 B
Reduce refactor (codepoint map)                   0 B - 1.00x memory usage +0 B
case refactor (codepoint list)                    0 B - 1.00x memory usage +0 B
guard refactor (codepoint list)                   0 B - 1.00x memory usage +0 B
First attempt (string map)                       40 B - ∞ x memory usage +40 B
Stream refactor (string map)                    496 B - ∞ x memory usage +496 B
```

```
##### With input empty #####
Name                                          ips        average  deviation         median         99th %
function head refactor (codepoint list)    3.97 M      251.68 ns   ±956.65%          0 ns        1000 ns
Reduce refactor (codepoint map)            3.96 M      252.78 ns  ±1181.44%          0 ns        1000 ns
case refactor (codepoint list)             3.95 M      253.11 ns   ±937.02%          0 ns        1000 ns
guard refactor (codepoint list)            3.94 M      254.08 ns  ±1015.42%          0 ns        1000 ns
First attempt (string map)                 2.31 M      432.03 ns  ±7615.54%          0 ns        1000 ns
Stream refactor (string map)               1.39 M      720.55 ns  ±3838.17%       1000 ns        1000 ns

Comparison:
function head refactor (codepoint list)    3.97 M
Reduce refactor (codepoint map)            3.96 M - 1.00x slower +1.09 ns
case refactor (codepoint list)             3.95 M - 1.01x slower +1.42 ns
guard refactor (codepoint list)            3.94 M - 1.01x slower +2.40 ns
First attempt (string map)                 2.31 M - 1.72x slower +180.34 ns
Stream refactor (string map)               1.39 M - 2.86x slower +468.87 ns

Memory usage statistics:

Name                                   Memory usage
function head refactor (codepoint list)          0 B
Reduce refactor (codepoint map)                   0 B - 1.00x memory usage +0 B
case refactor (codepoint list)                    0 B - 1.00x memory usage +0 B
guard refactor (codepoint list)                   0 B - 1.00x memory usage +0 B
First attempt (string map)                       40 B - ∞ x memory usage +40 B
Stream refactor (string map)                    496 B - ∞ x memory usage +496 B
```

```
##### With input empty #####
Name                                          ips        average  deviation         median         99th %
function head refactor (codepoint list)    3.97 M      251.68 ns   ±956.65%          0 ns        1000 ns
Reduce refactor (codepoint map)            3.96 M      252.78 ns  ±1181.44%          0 ns        1000 ns
case refactor (codepoint list)             3.95 M      253.11 ns   ±937.02%          0 ns        1000 ns
guard refactor (codepoint list)            3.94 M      254.08 ns  ±1015.42%          0 ns        1000 ns
First attempt (string map)                 2.31 M      432.03 ns  ±7615.54%          0 ns        1000 ns
Stream refactor (string map)               1.39 M      720.55 ns  ±3838.17%       1000 ns        1000 ns

Comparison:
function head refactor (codepoint list)    3.97 M
Reduce refactor (codepoint map)            3.96 M - 1.00x slower +1.09 ns
case refactor (codepoint list)             3.95 M - 1.01x slower +1.42 ns
guard refactor (codepoint list)            3.94 M - 1.01x slower +2.40 ns
First attempt (string map)                 2.31 M - 1.72x slower +180.34 ns
Stream refactor (string map)               1.39 M - 2.86x slower +468.87 ns

Memory usage statistics:

Name                                   Memory usage
function head refactor (codepoint list)          0 B
Reduce refactor (codepoint map)                   0 B - 1.00x memory usage +0 B
case refactor (codepoint list)                    0 B - 1.00x memory usage +0 B
guard refactor (codepoint list)                   0 B - 1.00x memory usage +0 B
First attempt (string map)                       40 B - ∞ x memory usage +40 B
Stream refactor (string map)                    496 B - ∞ x memory usage +496 B
```

```
##### With input empty #####
Name                                          ips        average  deviation         median         99th %
function head refactor (codepoint list)    3.97 M      251.68 ns   ±956.65%           0 ns        1000 ns
Reduce refactor (codepoint map)            3.96 M      252.78 ns  ±1181.44%           0 ns        1000 ns
case refactor (codepoint list)             3.95 M      253.11 ns   ±937.02%           0 ns        1000 ns
guard refactor (codepoint list)            3.94 M      254.08 ns  ±1015.42%           0 ns        1000 ns
First attempt (string map)                 2.31 M      432.03 ns  ±7615.54%           0 ns        1000 ns
Stream refactor (string map)               1.39 M      720.55 ns  ±3838.17%        1000 ns        1000 ns

Comparison:
function head refactor (codepoint list)    3.97 M
Reduce refactor (codepoint map)            3.96 M - 1.00x slower +1.09 ns
case refactor (codepoint list)             3.95 M - 1.01x slower +1.42 ns
guard refactor (codepoint list)            3.94 M - 1.01x slower +2.40 ns
First attempt (string map)                 2.31 M - 1.72x slower +180.34 ns
Stream refactor (string map)               1.39 M - 2.86x slower +468.87 ns

Memory usage statistics:

Name                                   Memory usage
function head refactor (codepoint list)          0 B
Reduce refactor (codepoint map)                   0 B - 1.00x memory usage +0 B
case refactor (codepoint list)                    0 B - 1.00x memory usage +0 B
guard refactor (codepoint list)                   0 B - 1.00x memory usage +0 B
First attempt (string map)                       40 B - ∞ x memory usage +40 B
Stream refactor (string map)                    496 B - ∞ x memory usage +496 B
```

```
##### With input entire alphabet #####
Name                                           ips        average  deviation         median         99th %
case refactor (codepoint list)              521.63 K        1.92 µs  ±1038.69%           2 µs           3 µs
function head refactor (codepoint list)     514.75 K        1.94 µs   ±981.41%           2 µs           3 µs
guard refactor (codepoint list)             508.82 K        1.97 µs  ±1022.73%           2 µs           3 µs
Reduce refactor (codepoint map)             420.84 K        2.38 µs   ±901.53%           2 µs           4 µs
First attempt (string map)                  103.55 K        9.66 µs    ±47.21%           9 µs          16 µs
Stream refactor (string map)                 94.19 K       10.62 µs    ±87.99%          10 µs          15 µs

Comparison:
case refactor (codepoint list)              521.63 K
function head refactor (codepoint list)     514.75 K - 1.01x slower +0.0256 µs
guard refactor (codepoint list)             508.82 K - 1.03x slower +0.0482 µs
Reduce refactor (codepoint map)             420.84 K - 1.24x slower +0.46 µs
First attempt (string map)                  103.55 K - 5.04x slower +7.74 µs
Stream refactor (string map)                 94.19 K - 5.54x slower +8.70 µs

Memory usage statistics:

Name                                    Memory usage
case refactor (codepoint list)              1.30 KB
function head refactor (codepoint list)     1.30 KB - 1.00x memory usage +0 KB
guard refactor (codepoint list)             1.30 KB - 1.00x memory usage +0 KB
Reduce refactor (codepoint map)             1.30 KB - 1.00x memory usage +0 KB
First attempt (string map)                     5 KB - 3.83x memory usage +3.70 KB
Stream refactor (string map)                6.05 KB - 4.64x memory usage +4.75 KB
```

```
##### With input entire alphabet #####
Name                                                ips        average  deviation         median         99th %
case refactor (codepoint list)                   521.63 K        1.92 µs  ±1038.69%           2 µs           3 µs
function head refactor (codepoint list)          514.75 K        1.94 µs   ±981.41%           2 µs           3 µs
guard refactor (codepoint list)                  508.82 K        1.97 µs  ±1022.73%           2 µs           3 µs
Reduce refactor (codepoint map)                  420.84 K        2.38 µs   ±901.53%           2 µs           4 µs
First attempt (string map)                       103.55 K        9.66 µs    ±47.21%           9 µs          16 µs
Stream refactor (string map)                      94.19 K       10.62 µs    ±87.99%          10 µs          15 µs

Comparison:
case refactor (codepoint list)                   521.63 K
function head refactor (codepoint list)          514.75 K - 1.01x slower +0.0256 µs
guard refactor (codepoint list)                  508.82 K - 1.03x slower +0.0482 µs
Reduce refactor (codepoint map)                  420.84 K - 1.24x slower +0.46 µs
First attempt (string map)                       103.55 K - 5.04x slower +7.74 µs
Stream refactor (string map)                      94.19 K - 5.54x slower +8.70 µs

Memory usage statistics:

Name                                         Memory usage
case refactor (codepoint list)                    1.30 KB
function head refactor (codepoint list)           1.30 KB - 1.00x memory usage +0 KB
guard refactor (codepoint list)                   1.30 KB - 1.00x memory usage +0 KB
Reduce refactor (codepoint map)                   1.30 KB - 1.00x memory usage +0 KB
First attempt (string map)                           5 KB - 3.83x memory usage +3.70 KB
Stream refactor (string map)                      6.05 KB - 4.64x memory usage +4.75 KB
```

```
##### With input entire alphabet #####
Name                                          ips        average  deviation         median         99th %
case refactor (codepoint list)             521.63 K        1.92 µs  ±1038.69%           2 µs           3 µs
function head refactor (codepoint list)    514.75 K        1.94 µs   ±981.41%           2 µs           3 µs
guard refactor (codepoint list)            508.82 K        1.97 µs  ±1022.73%           2 µs           3 µs
Reduce refactor (codepoint map)            420.84 K        2.38 µs   ±901.53%           2 µs           4 µs
First attempt (string map)                 103.55 K        9.66 µs    ±47.21%           9 µs          16 µs
Stream refactor (string map)                94.19 K       10.62 µs    ±87.99%          10 µs          15 µs

Comparison:
case refactor (codepoint list)             521.63 K
function head refactor (codepoint list)    514.75 K - 1.01x slower +0.0256 µs
guard refactor (codepoint list)            508.82 K - 1.03x slower +0.0482 µs
Reduce refactor (codepoint map)            420.84 K - 1.24x slower +0.46 µs
First attempt (string map)                 103.55 K - 5.04x slower +7.74 µs
Stream refactor (string map)                94.19 K - 5.54x slower +8.70 µs

Memory usage statistics:

Name                                   Memory usage
case refactor (codepoint list)              1.30 KB
function head refactor (codepoint list)     1.30 KB - 1.00x memory usage +0 KB
guard refactor (codepoint list)             1.30 KB - 1.00x memory usage +0 KB
Reduce refactor (codepoint map)             1.30 KB - 1.00x memory usage +0 KB
First attempt (string map)                     5 KB - 3.83x memory usage +3.70 KB
Stream refactor (string map)                6.05 KB - 4.64x memory usage +4.75 KB
```

```
##### With input entire alphabet #####
Name                                          ips        average  deviation         median         99th %
case refactor (codepoint list)             521.63 K        1.92 µs  ±1038.69%           2 µs           3 µs
function head refactor (codepoint list)    514.75 K        1.94 µs   ±981.41%           2 µs           3 µs
guard refactor (codepoint list)            508.82 K        1.97 µs  ±1022.73%           2 µs           3 µs
Reduce refactor (codepoint map)            420.84 K        2.38 µs   ±901.53%           2 µs           4 µs
First attempt (string map)                 103.55 K        9.66 µs    ±47.21%           9 µs          16 µs
Stream refactor (string map)                94.19 K       10.62 µs    ±87.99%          10 µs          15 µs

Comparison:
case refactor (codepoint list)             521.63 K
function head refactor (codepoint list)    514.75 K - 1.01x slower +0.0256 µs
guard refactor (codepoint list)            508.82 K - 1.03x slower +0.0482 µs
Reduce refactor (codepoint map)            420.84 K - 1.24x slower +0.46 µs
First attempt (string map)                 103.55 K - 5.04x slower +7.74 µs
Stream refactor (string map)                94.19 K - 5.54x slower +8.70 µs

Memory usage statistics:

Name                                   Memory usage
case refactor (codepoint list)              1.30 KB
function head refactor (codepoint list)     1.30 KB - 1.00x memory usage +0 KB
guard refactor (codepoint list)             1.30 KB - 1.00x memory usage +0 KB
Reduce refactor (codepoint map)             1.30 KB - 1.00x memory usage +0 KB
First attempt (string map)                     5 KB - 3.83x memory usage +3.70 KB
Stream refactor (string map)                6.05 KB - 4.64x memory usage +4.75 KB
```

Possible Conclusions?

Possible Conclusions?

🔠 Map lookup not necessarily more performant

Possible Conclusions?

- Map lookup not necessarily more performant
- Compiler optimises O(n) actions in case/function heads/guards

Possible Conclusions?

- 🔠 Map lookup not necessarily more performant
- 🔠 Compiler optimises O(n) actions in case/function heads/guards
- 🔠 Refactor was justified by science

github.com/paulfioravanti/
exercism_scrabble_benchmark

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

Thanks!

@paulfioravanti