Tables and Lists

8 February 2024
Where are we?
<table>
<thead>
<tr>
<th>player</th>
<th>team</th>
<th>pos</th>
<th>games</th>
<th>pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Lindsay Allen&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>29</td>
<td>179</td>
</tr>
<tr>
<td>&quot;Rebecca Allen&quot;</td>
<td>&quot;CON&quot;</td>
<td>&quot;G&quot;</td>
<td>40</td>
<td>255</td>
</tr>
<tr>
<td>&quot;Laeticia Amihere&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>20</td>
<td>56</td>
</tr>
<tr>
<td>&quot;Ariel Atkins&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;G&quot;</td>
<td>27</td>
<td>311</td>
</tr>
<tr>
<td>&quot;Shakira Austin&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;C-F&quot;</td>
<td>19</td>
<td>190</td>
</tr>
<tr>
<td>&quot;Rachel Banham&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>32</td>
<td>176</td>
</tr>
<tr>
<td>&quot;Kierstan Bell&quot;</td>
<td>&quot;LVA&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>132</td>
</tr>
<tr>
<td>&quot;Grace Berger&quot;</td>
<td>&quot;IND&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>151</td>
</tr>
<tr>
<td>&quot;Morgan Bertsch&quot;</td>
<td>&quot;CHI&quot;</td>
<td>&quot;F&quot;</td>
<td>28</td>
<td>122</td>
</tr>
<tr>
<td>&quot;Monique Billings&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>39</td>
<td>187</td>
</tr>
</tbody>
</table>

Click to show the remaining 165 rows...
<table>
<thead>
<tr>
<th>player</th>
<th>team</th>
<th>pos</th>
<th>games</th>
<th>pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Lindsay Allen&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>29</td>
<td>179</td>
</tr>
<tr>
<td>&quot;Rebecca Allen&quot;</td>
<td>&quot;CON&quot;</td>
<td>&quot;G&quot;</td>
<td>40</td>
<td>255</td>
</tr>
<tr>
<td>&quot;Laeticia Amihere&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>20</td>
<td>56</td>
</tr>
<tr>
<td>&quot;Ariel Atkins&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;G&quot;</td>
<td>27</td>
<td>311</td>
</tr>
<tr>
<td>&quot;Shakira Austin&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;C-F&quot;</td>
<td>19</td>
<td>190</td>
</tr>
<tr>
<td>&quot;Rachel Banham&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>32</td>
<td>176</td>
</tr>
<tr>
<td>&quot;Kierstan Bell&quot;</td>
<td>&quot;LVA&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>132</td>
</tr>
<tr>
<td>&quot;Grace Berger&quot;</td>
<td>&quot;IND&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>151</td>
</tr>
<tr>
<td>&quot;Morgan Bertsch&quot;</td>
<td>&quot;CHI&quot;</td>
<td>&quot;F&quot;</td>
<td>28</td>
<td>122</td>
</tr>
<tr>
<td>&quot;Monique Billings&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>39</td>
<td>187</td>
</tr>
</tbody>
</table>
How do I get just this row from `stats`?

```plaintext
stats =

<table>
<thead>
<tr>
<th>player</th>
<th>team</th>
<th>pos</th>
<th>games</th>
<th>pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Lindsay Allen&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>29</td>
<td>179</td>
</tr>
<tr>
<td>&quot;Rebecca Allen&quot;</td>
<td>&quot;CON&quot;</td>
<td>&quot;G&quot;</td>
<td>40</td>
<td>255</td>
</tr>
<tr>
<td>&quot;Laeticia Amihere&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>20</td>
<td>56</td>
</tr>
<tr>
<td>&quot;Ariel Atkins&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;G&quot;</td>
<td>27</td>
<td>311</td>
</tr>
<tr>
<td>&quot;Shakira Austin&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;C-F&quot;</td>
<td>19</td>
<td>190</td>
</tr>
<tr>
<td>&quot;Rachel Banham&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>32</td>
<td>176</td>
</tr>
<tr>
<td>&quot;Kierstan Bell&quot;</td>
<td>&quot;LVA&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>132</td>
</tr>
<tr>
<td>&quot;Grace Berger&quot;</td>
<td>&quot;IND&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>151</td>
</tr>
<tr>
<td>&quot;Morgan Bertsch&quot;</td>
<td>&quot;CHI&quot;</td>
<td>&quot;F&quot;</td>
<td>28</td>
<td>122</td>
</tr>
<tr>
<td>&quot;Monique Billings&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>39</td>
<td>187</td>
</tr>
</tbody>
</table>
```
How do I get just this row from `stats`?
How do I get just this row from `stats`?

```r
stats =
# stats

<table>
<thead>
<tr>
<th>player</th>
<th>team</th>
<th>pos</th>
<th>games</th>
<th>pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Lindsay Allen&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>29</td>
<td>179</td>
</tr>
<tr>
<td>&quot;Rebecca Allen&quot;</td>
<td>&quot;CON&quot;</td>
<td>&quot;G&quot;</td>
<td>40</td>
<td>255</td>
</tr>
<tr>
<td>&quot;Laetitia Amihere&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>20</td>
<td>56</td>
</tr>
<tr>
<td>&quot;Ariel Atkins&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;G&quot;</td>
<td>27</td>
<td>311</td>
</tr>
<tr>
<td>&quot;Shakira Austin&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;C-F&quot;</td>
<td>19</td>
<td>190</td>
</tr>
<tr>
<td>&quot;Rachel Banham&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>32</td>
<td>176</td>
</tr>
<tr>
<td>&quot;Kierstan Bell&quot;</td>
<td>&quot;LVA&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>132</td>
</tr>
<tr>
<td>&quot;Grace Berger&quot;</td>
<td>&quot;IND&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>151</td>
</tr>
<tr>
<td>&quot;Morgan Bertsch&quot;</td>
<td>&quot;CHI&quot;</td>
<td>&quot;F&quot;</td>
<td>28</td>
<td>122</td>
</tr>
<tr>
<td>&quot;Monique Billings&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>39</td>
<td>187</td>
</tr>
</tbody>
</table>
```

Click to show the remaining 165 rows...
<table>
<thead>
<tr>
<th>player</th>
<th>team</th>
<th>pos</th>
<th>games</th>
<th>pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Lindsay Allen&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>29</td>
<td>179</td>
</tr>
<tr>
<td>&quot;Rebecca Allen&quot;</td>
<td>&quot;CON&quot;</td>
<td>&quot;G&quot;</td>
<td>40</td>
<td>255</td>
</tr>
<tr>
<td>&quot;Laeticia Amihere&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>20</td>
<td>56</td>
</tr>
<tr>
<td>&quot;Ariel Atkins&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;G&quot;</td>
<td>27</td>
<td>311</td>
</tr>
<tr>
<td>&quot;Shakira Austin&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;C-F&quot;</td>
<td>19</td>
<td>190</td>
</tr>
<tr>
<td>&quot;Rachel Banham&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>32</td>
<td>176</td>
</tr>
<tr>
<td>&quot;Kierstan Bell&quot;</td>
<td>&quot;LVA&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>132</td>
</tr>
<tr>
<td>&quot;Grace Berger&quot;</td>
<td>&quot;IND&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>151</td>
</tr>
<tr>
<td>&quot;Morgan Bertsch&quot;</td>
<td>&quot;CHI&quot;</td>
<td>&quot;F&quot;</td>
<td>28</td>
<td>122</td>
</tr>
<tr>
<td>&quot;Monique Billings&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>39</td>
<td>187</td>
</tr>
</tbody>
</table>

How do I get just the rows for players who are guards?

Click to show the remaining 165 rows...
### Question
How do I get just the rows for players who are guards?

### Solution
```lisp
(filter-with (stats, is-guard)
  "Lindsay Allen" "MIN" "G" 29 179
  "Rebecca Allen" "CON" "G" 40 255
  "Laetitia Amihere" "ATL" "F" 20 56
  "Ariel Atkins" "WAS" "G" 27 311
  "Shakira Austin" "WAS" "C-F" 19 190
)```
fun is-guard(player :: Row) -> Boolean:
  doc: "Return true if the player's primary position is guard"
  player["pos"] == "G"
where:
  is-guard(stats.row-n(0)) is true
  is-guard(stats.row-n(2)) is false
end

filter-with(stats, is-guard)
<table>
<thead>
<tr>
<th>player</th>
<th>team</th>
<th>pos</th>
<th>games</th>
<th>pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Lindsay Allen&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>29</td>
<td>179</td>
</tr>
<tr>
<td>&quot;Rebecca Allen&quot;</td>
<td>&quot;CON&quot;</td>
<td>&quot;G&quot;</td>
<td>40</td>
<td>255</td>
</tr>
<tr>
<td>&quot;Laeticia Amihere&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>20</td>
<td>56</td>
</tr>
<tr>
<td>&quot;Ariel Atkins&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;G&quot;</td>
<td>27</td>
<td>311</td>
</tr>
<tr>
<td>&quot;Shakira Austin&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;C-F&quot;</td>
<td>19</td>
<td>190</td>
</tr>
<tr>
<td>&quot;Rachel Banham&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>32</td>
<td>176</td>
</tr>
<tr>
<td>&quot;Kierstan Bell&quot;</td>
<td>&quot;LVA&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>132</td>
</tr>
<tr>
<td>&quot;Grace Berger&quot;</td>
<td>&quot;IND&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>151</td>
</tr>
<tr>
<td>&quot;Morgan Bertsch&quot;</td>
<td>&quot;CHI&quot;</td>
<td>&quot;F&quot;</td>
<td>28</td>
<td>122</td>
</tr>
<tr>
<td>&quot;Monique Billings&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>39</td>
<td>187</td>
</tr>
</tbody>
</table>

Click to show the remaining 165 rows...
<table>
<thead>
<tr>
<th>player</th>
<th>team</th>
<th>pos</th>
<th>games</th>
<th>pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Lindsay Allen&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>29</td>
<td>179</td>
</tr>
<tr>
<td>&quot;Rebecca Allen&quot;</td>
<td>&quot;CON&quot;</td>
<td>&quot;G&quot;</td>
<td>40</td>
<td>255</td>
</tr>
<tr>
<td>&quot;Laetitia Amihere&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>20</td>
<td>56</td>
</tr>
<tr>
<td>&quot;Ariel Atkins&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;G&quot;</td>
<td>27</td>
<td>311</td>
</tr>
<tr>
<td>&quot;Shakira Austin&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;C-F&quot;</td>
<td>19</td>
<td>190</td>
</tr>
<tr>
<td>&quot;Rachel Banham&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>32</td>
<td>176</td>
</tr>
<tr>
<td>&quot;Kierstan Bell&quot;</td>
<td>&quot;LVA&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>132</td>
</tr>
<tr>
<td>&quot;Grace Berger&quot;</td>
<td>&quot;IND&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>151</td>
</tr>
<tr>
<td>&quot;Morgan Bertsch&quot;</td>
<td>&quot;CHI&quot;</td>
<td>&quot;F&quot;</td>
<td>28</td>
<td>122</td>
</tr>
<tr>
<td>&quot;Monique Billings&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>39</td>
<td>187</td>
</tr>
</tbody>
</table>

What about columns?
How can I add a new column like this?

<table>
<thead>
<tr>
<th>player</th>
<th>team</th>
<th>pos</th>
<th>games</th>
<th>pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Lindsay Allen&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>29</td>
<td>179</td>
</tr>
<tr>
<td>&quot;Rebecca Allen&quot;</td>
<td>&quot;CON&quot;</td>
<td>&quot;G&quot;</td>
<td>40</td>
<td>255</td>
</tr>
<tr>
<td>&quot;Laetitia Amihere&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>20</td>
<td>56</td>
</tr>
<tr>
<td>&quot;Ariel Atkins&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;G&quot;</td>
<td>27</td>
<td>311</td>
</tr>
<tr>
<td>&quot;Shakira Austin&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;C-F&quot;</td>
<td>19</td>
<td>190</td>
</tr>
<tr>
<td>&quot;Rachel Banham&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>32</td>
<td>176</td>
</tr>
<tr>
<td>&quot;Kierstan Bell&quot;</td>
<td>&quot;LVA&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>132</td>
</tr>
<tr>
<td>&quot;Grace Berger&quot;</td>
<td>&quot;IND&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>151</td>
</tr>
<tr>
<td>&quot;Morgan Bertsch&quot;</td>
<td>&quot;CHI&quot;</td>
<td>&quot;F&quot;</td>
<td>28</td>
<td>122</td>
</tr>
<tr>
<td>&quot;Monique Billings&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>39</td>
<td>187</td>
</tr>
</tbody>
</table>

frequent-player

"somewhat"  "very"  "somewhat"  "no"  "very"  "very"  "very"  "somewhat"  "very"

Click to show the remaining 165 rows...
<table>
<thead>
<tr>
<th>player</th>
<th>team</th>
<th>pos</th>
<th>games</th>
<th>pts</th>
<th>frequent-player</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Lindsay Allen&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>29</td>
<td>179</td>
<td>&quot;somewhat&quot;</td>
</tr>
<tr>
<td>&quot;Rebecca Allen&quot;</td>
<td>&quot;CON&quot;</td>
<td>&quot;G&quot;</td>
<td>40</td>
<td>255</td>
<td>&quot;very&quot;</td>
</tr>
<tr>
<td>&quot;Laeticia Amihere&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>20</td>
<td>56</td>
<td>&quot;somewhat&quot;</td>
</tr>
<tr>
<td>&quot;Ariel Atkins&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;G&quot;</td>
<td>27</td>
<td>311</td>
<td>&quot;somewhat&quot;</td>
</tr>
<tr>
<td>&quot;Shakira Austin&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;C-F&quot;</td>
<td>19</td>
<td>190</td>
<td>&quot;no&quot;</td>
</tr>
<tr>
<td>&quot;Rachel Banham&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>32</td>
<td>176</td>
<td>&quot;very&quot;</td>
</tr>
<tr>
<td>&quot;Kierstan Bell&quot;</td>
<td>&quot;LVA&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>132</td>
<td>&quot;very&quot;</td>
</tr>
</tbody>
</table>

How can I add a new column like this?

```
build-column(stats, "frequent-player", how-frequent)
```
fun how-frequent(player :: Row) -> String:
    if player["games"] >= 30:
        "very"
    else if player["games"] >= 20:
        "somewhat"
    else:
        "no"
end
end

build-column(stats, "frequent-player", how-frequent)
Changing a column
So, we’ve seen that we can build a new column based on the values in each row, but what if we just want to change an existing column?
A fake WNBA fan like me can’t remember what these team abbreviations stand for.

Let’s fill in the actual team names.

<table>
<thead>
<tr>
<th>player</th>
<th>team</th>
<th>pos</th>
<th>games</th>
<th>pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Lindsay Allen&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>29</td>
<td>179</td>
</tr>
<tr>
<td>&quot;Rebecca Allen&quot;</td>
<td>&quot;CON&quot;</td>
<td>&quot;G&quot;</td>
<td>40</td>
<td>255</td>
</tr>
<tr>
<td>&quot;Laeticia Amihere&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>20</td>
<td>56</td>
</tr>
<tr>
<td>&quot;Ariel Atkins&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;G&quot;</td>
<td>27</td>
<td>311</td>
</tr>
<tr>
<td>&quot;Shakira Austin&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;C-F&quot;</td>
<td>19</td>
<td>190</td>
</tr>
<tr>
<td>&quot;Rachel Banham&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>32</td>
<td>176</td>
</tr>
<tr>
<td>&quot;Kierstan Bell&quot;</td>
<td>&quot;LVA&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>132</td>
</tr>
<tr>
<td>&quot;Grace Berger&quot;</td>
<td>&quot;IND&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>151</td>
</tr>
<tr>
<td>&quot;Morgan Bertsch&quot;</td>
<td>&quot;CHI&quot;</td>
<td>&quot;F&quot;</td>
<td>28</td>
<td>122</td>
</tr>
<tr>
<td>&quot;Monique Billings&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>39</td>
<td>187</td>
</tr>
</tbody>
</table>

Click to show the remaining 165 rows...
What *are* the team names?
<table>
<thead>
<tr>
<th>WESTERN</th>
<th></th>
<th>EASTERN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DAL</strong></td>
<td>![logo]</td>
<td><strong>ATL</strong></td>
</tr>
<tr>
<td></td>
<td>Dallas Wings</td>
<td>Atlanta Dream</td>
</tr>
<tr>
<td><strong>LVA</strong></td>
<td>![logo]</td>
<td><strong>CHI</strong></td>
</tr>
<tr>
<td></td>
<td>Las Vegas Aces</td>
<td>Chicago Sky</td>
</tr>
<tr>
<td><strong>LAS</strong></td>
<td>![logo]</td>
<td><strong>CON</strong></td>
</tr>
<tr>
<td></td>
<td>Los Angeles Sparks</td>
<td>Connecticut Sun</td>
</tr>
<tr>
<td><strong>MIN</strong></td>
<td>![logo]</td>
<td><strong>IND</strong></td>
</tr>
<tr>
<td></td>
<td>Minnesota Lynx</td>
<td>Indiana Fever</td>
</tr>
<tr>
<td><strong>PHO</strong></td>
<td>![logo]</td>
<td><strong>NYL</strong></td>
</tr>
<tr>
<td></td>
<td>Phoenix Mercury</td>
<td>New York Liberty</td>
</tr>
<tr>
<td><strong>SEA</strong></td>
<td>![logo]</td>
<td><strong>WAS</strong></td>
</tr>
<tr>
<td></td>
<td>Seattle Storm</td>
<td>Washington Mystics</td>
</tr>
</tbody>
</table>
fun team-name(abbr :: String) -> String:
    doc: "Return the name of the team with the given abbreviation"
    ...
where:
    team-name("NYL") is "New York Liberty"
    team-name("CHI") is "Chicago Sky"
    ...
end
fun team-name(abbr :: String) -> String:
    doc: "Return the name of the team with the given abbreviation"
    if abbr == "DAL": "Dallas Wings"
    else if abbr == "LVA": "Las Vegas Aces"
    ...
end
where:
    team-name("NYL") is "New York Liberty"
    team-name("CHI") is "Chicago Sky"
    ...
end
fun team-name(abbr :: String) -> String:
  doc: "Return the name of the team with the given abbreviation"
  if abbr == "DAL": "Dallas Wings"
  else if abbr == "LVA": "Las Vegas Aces"
  ...
  end

where:
  team-name("NYL") is "New York Liberty"
  team-name("CHI") is "Chicago Sky"
  ...
  end

This will work, but remember what we said when we introduced tables for looking up population: We want to separate data from computation.
teams =
  table: abbr, name
  row: "DAL", "Dallas Wings"
  row: "LVA", "Las Vegas Aces"
  row: "LAS", "Los Angeles Sparks"
  row: "MIN", "Minnesota Lynx"
  row: "PHO", "Phoenix Mercury"
  row: "SEA", "Seattle Storm"
  row: "ATL", "Atlanta Dream"
  row: "CHI", "Chicago Sky"
  row: "CON", "Connecticut Sun"
  row: "IND", "Indiana Fever"
  row: "NYL", "New York Liberty"
  row: "WAS", "Washington Mystics"
end

Advantage: This makes it easy to add new teams or more information about these teams, in a central place.
fun team-name(abbr :: String) -> String:
  doc: "Return the name of the team with the given abbreviation"

  # 1. Get the row with abbreviation `abbr`
  # 2. Return the value in the `name` column

where:
  team-name("NYL") is "New York Liberty"
  team-name("CHI") is "Chicago Sky"

...
teams = ...

fun team-name(abbr :: String) -> String:
    doc: "Return the name of the team with the given abbreviation"

    matches = filter-with(teams, has-abbr)
    team = matches.row-n(0)
    # 2. Return the value in the `name` column

where:
    team-name("NYL") is "New York Liberty"
    team-name("CHI") is "Chicago Sky"
    ...
end

# Wishlist:
# - has-abbr :: Row -> Boolean
teams = ...

fun team-name(abbr :: String) -> String:
   doc: "Return the name of the team with the given abbreviation"

   matches = filter-with(teams, has-abbr)
   team = matches.row-n(0)
   team["name"]

where:
   team-name("NYL") is "New York Liberty"
   team-name("CHI") is "Chicago Sky"
   ...
end

# Wishlist:
# - has-abbr :: Row -> Boolean
teams = ...

fun team-name(abbr :: String) -> String:
    doc: "Return the name of the team with the given abbreviation"

    matches = filter-with(teams, has-abbr)
    team = matches.row-n(0)
    team["name"]

where:
    team-name("NYL") is "New York Liberty"
    team-name("CHI") is "Chicago Sky"
    ...
end

fun has-abbr(r :: Row) -> Boolean:
    doc: "Return true if the row has the given abbreviation"
    r["abbr"] == abbr

where:
    has-abbr(teams.row-n(0)) is ...
end

Wait – what’s the abbr we’re checking?
teams = ...

fun team-name(abbr :: String) -> String:
  doc: "Return the name of the team with the given abbreviation"

  matches = filter-with(teams, has-abbr)
team = matches.row-n(0)
team["name"]

where:
  team-name("NYL") is "New York Liberty"
  team-name("CHI") is "Chicago Sky"
...
end

fun has-abbr(r :: Row) -> Boolean:
  doc: "Return true if the row has the given abbreviation"

  r["abbr"] == abbr

where:
  has-abbr(teams.row-n(0)) is ...
end
fun team-name(abbr :: String) -> String:
    doc: "Return the name of the team with the given abbreviation"

matches = filter-with(teams, has-abbr)
team = matches.row-n(0)
team["name"]

where:
    team-name("NYL") is "New York Liberty"
    team-name("CHI") is "Chicago Sky"
...
end

fun has-abbr(r :: Row, abbr :: String) -> Boolean:
    doc: "Return true if the row has the given abbreviation"
    r["abbr"] == abbr
where:
    has-abbr(teams.row-n(0)) is ...
end

Why can't I do this – make abbr another input to the has-abbr function?
teams = ...

fun team-name(abbr :: String) -> String:
  doc: "Return the name of the team with the given abbreviation"

    matches = filter-with(teams, has-abbr)
    team = matches.row-n(0)
    team["name"]

where:
  team-name("NYL") is "New York Liberty"
  team-name("CHI") is "Chicago Sky"
 ...
end

fun has-abbr(r :: Row, abbr :: String) -> Boolean:
  doc: "Return true if the row has the given abbreviation"

    r["abbr"] == abbr

where:
  has-abbr(teams.row-n(0)) is ...
end

When `filter-with` calls `has-abbr`, the only input it provides is the row.
teams = ...

fun team-name(abbr :: String) -> String:
    doc: "Return the name of the team with the given abbreviation"

    matches = filter-with(teams, has-abbr)
    team = matches.row-n(0)
    team["name"]

    where:
        team-name("NYL") is "New York Liberty"
        team-name("CHI") is "Chicago Sky"

    end

end

fun has-abbr(r :: Row) -> Boolean:
    doc: "Return true if the row has the given abbreviation"
    r["abbr"] == abbr

    where:
        has-abbr(teams.row-n(0)) is ...

    end

So, what can we do so that abbr is defined when we need to use it in has-abbr?
teams = ...

fun team-name(abbr :: String) -> String:
  doc: "Return the name of the team with the given abbreviation"

  r["abbr"] == abbr

end

matches = filter-with(teams, has-abbr)
team = matches.row-n(0)
team["name"]

where:
  team-name("NYL") is "New York Liberty"
  team-name("CHI") is "Chicago Sky"
  ...

end

We can define a function inside another function. Now has-abbr can use any of the inputs we gave to team-name.
teams = ...

fun team-name(abbr :: String) -> String:
  doc: "Return the name of the team with the given abbreviation"

  fun has-abbr(r :: Row) -> Boolean:
    doc: "Return true if the row has the given abbreviation"
    r["abbr"] == abbr
  end

  matches = filter-with(teams, has-abbr)
  team = matches.row-n(0)
  team["name"]

  where:
    team-name("NYL") is "New York Liberty"
    team-name("CHI") is "Chicago Sky"
    ...

end

But, actually, has-abbr is a very small function, and we only use it in one place.
teams = ...

fun team-name(abbr :: String) -> String:
  doc: "Return the name of the team with the given abbreviation"

  fun has-abbr(r :: Row) -> Boolean:
    doc: "Return true if the row has the given abbreviation"
    r["abbr"] == abbr
    end

  matches = filter-with(teams, lam(r): r["abbr"] == abbr end)
  team = matches.row-n(0)
  team["name"]

  where:
    team-name("NYL") is "New York Liberty"
    team-name("CHI") is "Chicago Sky"
    ...
  end

This means we can just write the function body inline as a lambda expression, without giving a name to it.
We can name a number value and then use it:

\[
width = 10 \\
2 \times width
\]

Or we can just use it without giving a name:

\[
2 \times 10
\]

Likewise, we can name a function value and then use it:

fun add-3(x): x + 3 end
add-3(10)

or we can just use it:

(lam(x): x + 3 end)(10)
teams = ... 

fun team-name(abbr :: String) -> String:
    doc: "Return the name of the team with the given abbreviation"
    
    fun has-abbr(r :: Row) -> Boolean:
        doc: "Return true if the row has the given abbreviation"
        r["abbr"] == abbr
        end

    matches = filter-with(teams, lam(r): r["abbr"] == abbr end)
    team = matches.row-n(0)
    team["name"]

where:
    team-name("NYL") is "New York Liberty"
    team-name("CHI") is "Chicago Sky"
    ...
end

The lambda expression removes the need for the full function definition.
teams = ...

fun team-name(abbr :: String) -> String:
  doc: "Return the name of the team with the given abbreviation"

  matches = filter-with(teams, lam(r): r["abbr"] == abbr end)
  team = matches.row-n(0)
  team["name"]

where:
  team-name("NYL") is "New York Liberty"
  team-name("CHI") is "Chicago Sky"
...
teams = ...

fun team-name(abbr :: String) -> String:
  doc: "Return the name of the team with the given abbreviation"

  matches = filter-with(teams, lam(r): r["abbr"] == abbr end)
  team = matches.row-n(0)
  team["name"]

where:
  team-name("NYL") is "New York Liberty"
  team-name("CHI") is "Chicago Sky"

end

transform-column(stats, "team", team-name)
row-n-too-large

(Show program evaluation trace...)

oh no
teams = ...

fun team-name(abbr :: String) -> String:
  doc: "Return the name of the team with the given abbreviation"

  matches = filter-with(teams, lam(r): r["abbr"] == abbr end)
  team = matches.row-n(0)
  team["name"]

  where:
    team-name("NYL") is "New York Liberty"
    team-name("CHI") is "Chicago Sky"
    ...

  end

transform-column(stats, "team", team-name)

0 is too big? That means there were no matching rows! An abbreviation not in our table – what is it?
teams = ...

fun team-name(abbr :: String) -> String:
  doc: "Return the name of the team with the given abbreviation"

  matches = filter-with(teams, lam(r): r["abbr"] == abbr end)
  if matches.length() == 0:
    abbr
  else:
    team = matches.row-n(0)
    team["name"]
  end

where:
  team-name("NYL") is "New York Liberty"
  team-name("CHI") is "Chicago Sky"
  ...
end

transform-column(stats, "team", team-name)
<table>
<thead>
<tr>
<th>Name</th>
<th>Team</th>
<th>Position</th>
<th>Games</th>
<th>Minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jackie Young</td>
<td>Las Vegas Aces</td>
<td>G</td>
<td>40</td>
<td>704</td>
</tr>
<tr>
<td>Kristine Anigwe</td>
<td>TOT</td>
<td>F-C</td>
<td>13</td>
<td>25</td>
</tr>
<tr>
<td>Alaina Coates</td>
<td>TOT</td>
<td>C</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>Liz Dixon</td>
<td>TOT</td>
<td>F</td>
<td>14</td>
<td>27</td>
</tr>
<tr>
<td>Queen Egbo</td>
<td>TOT</td>
<td>F-C</td>
<td>37</td>
<td>178</td>
</tr>
<tr>
<td>Destanni Henderson</td>
<td>TOT</td>
<td>G</td>
<td>15</td>
<td>57</td>
</tr>
<tr>
<td>Joyner Holmes</td>
<td>TOT</td>
<td></td>
<td></td>
<td>113</td>
</tr>
<tr>
<td>Ashley Joens</td>
<td>TOT</td>
<td>G</td>
<td>19</td>
<td>39</td>
</tr>
<tr>
<td>Evina Westbrook</td>
<td>TOT</td>
<td>G</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>Amanda Zahui B.</td>
<td>TOT</td>
<td>C</td>
<td>34</td>
<td>57</td>
</tr>
<tr>
<td>Kristine Anigwe</td>
<td>Chicago Sky</td>
<td>F-C</td>
<td>10</td>
<td>25</td>
</tr>
</tbody>
</table>

**Total? Players who played for more than one team?**
What’s a column anyway?
We've seen that when you want a row of a table, you use `.row-n` and get a Row.

What about getting a column?
How do I get just the points column?

```python
stats =
```

<table>
<thead>
<tr>
<th>player</th>
<th>team</th>
<th>pos</th>
<th>games</th>
<th>pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Lindsay Allen&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>29</td>
<td>179</td>
</tr>
<tr>
<td>&quot;Rebecca Allen&quot;</td>
<td>&quot;CON&quot;</td>
<td>&quot;G&quot;</td>
<td>40</td>
<td>255</td>
</tr>
<tr>
<td>&quot;Laeticia Amihere&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>20</td>
<td>56</td>
</tr>
<tr>
<td>&quot;Ariel Atkins&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;G&quot;</td>
<td>27</td>
<td>311</td>
</tr>
<tr>
<td>&quot;Shakira Austin&quot;</td>
<td>&quot;WAS&quot;</td>
<td>&quot;C-F&quot;</td>
<td>19</td>
<td>190</td>
</tr>
<tr>
<td>&quot;Rachel Banham&quot;</td>
<td>&quot;MIN&quot;</td>
<td>&quot;G&quot;</td>
<td>32</td>
<td>176</td>
</tr>
<tr>
<td>&quot;Kierstan Bell&quot;</td>
<td>&quot;LVA&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>132</td>
</tr>
<tr>
<td>&quot;Grace Berger&quot;</td>
<td>&quot;IND&quot;</td>
<td>&quot;G&quot;</td>
<td>36</td>
<td>151</td>
</tr>
<tr>
<td>&quot;Morgan Bertsch&quot;</td>
<td>&quot;CHI&quot;</td>
<td>&quot;F&quot;</td>
<td>28</td>
<td>122</td>
</tr>
<tr>
<td>&quot;Monique Billings&quot;</td>
<td>&quot;ATL&quot;</td>
<td>&quot;F&quot;</td>
<td>39</td>
<td>187</td>
</tr>
</tbody>
</table>
stats =

```python
stats.get_column("pts")
```
```python
>>> stats.get-column("pts")
[179, 255, 56, 311, 190, ...]
```
The data type isn’t Column; it’s List!
A *List* is an ordered sequence of data.

For example,

\[
\text{grades} = \text{list:} \ 0.96, 0.73, 1.0, 0.5
\]

\[
\text{fellowship} = \text{list:} \\
\quad \text{"Frodo", "Sam", "Merry", "Pippin", "Gandalf", "Legolas", "Gimli", "Aragorn", "Boromir"}
\]
So, what good is a List?
fun is-nearby(town :: String) -> String:

doc: ...

(town == "Hyde Park") or
(town == "Pleasant Valley") or
(town == "Poughkeepsie") or
(town == "LaGrange")

where:

...
nearby-towns = [list:
    "Hyde Park",
    "Pleasant Valley",
    "Poughkeepsie",
    "LaGrange"
]

fun is-nearby(town :: String) -> String:
    doc: ...
    member(nearby-towns, town)
where:
    ...
end
Mad Libs!
Thousands of Plural-Noun ago, there were calendars that enabled the ancient Plural-Noun to divide a year into twelve Plural-Noun, each month into Number, and each Plural-Noun into seven Plural-Noun. At fi Plural-Noun, people told time by a sun clock, sometimes known as the Noun. Ultimately, they invented the great timekeeping devices of today, such as the grandfather Noun, the pocket Noun, the alarm Noun, and, of course, the Body-Part watch. Children learn about clocks and time almost before they learn their A-B- Alphabet-Letter. They are taught that a day consists of 24 Plural-Noun, an hour has 60 Plural-Noun, and a minute has 60 Plural-Noun. By the time they are in Kindergarten, they know if the big Body-Part is at twelve and the little Body-Part is at three, that it is Number o'clock. I wish we could continue this Adjective lesson, but we've run out of Noun.
Thousands of years ago, there were calendars that enabled the ancient people to divide a year into twelve months, each month into four weeks, and each week into seven days. At first, people told time by a sun clock, sometimes known as the sundial. Ultimately, they invented the great timekeeping devices of today, such as the grandfather clock, the pocket watch, the alarm clock, and, of course, the body watch. Children learn about clocks and time almost before they learn their A-B-Cs. They are taught that a day consists of 24 hours, an hour has 60 minutes, and a minute has 60 seconds. By the time they are in Kindergarten, they know if the big hand is at twelve and the little hand is at three, that it is 12 o’clock. I wish we could continue this lesson, but we’ve run out of time.
How can we represent a text?
Thousands of Plural-Noun ago, there were calendars that enabled the ancient Plural-Noun to divide a year into twelve Plural-Noun, each month into Number weeks, and each week into seven Plural-Noun. At first, people told time by a sun clock, sometimes known as the Noun dial. Ultimately, they invented the great timekeeping devices of today, such as the grandfather Noun, the pocket Noun, the alarm Noun, and, of course, the Body-Part watch. Children learn about clocks and time almost before they learn their A-B- Alphabet-Letter s. They are taught that a day consists of 24 Plural-Noun, an hour has 60 Plural-Noun, and a minute has 60 Plural-Noun. By the time they are in Kindergarten, they know if the big Body-Part is at twelve and the little Body-Part is at three, that it is Number o'clock. I wish we could continue this Adjective lesson, but we’ve run out of Noun.
template = "Thousands of Plural-Noun ago, ..."

template-words = string-split-all(template, " ")

>>> template-words
["Thousands", "of", "Plural-Noun", "ago," , ...]
We need to substitute a random plural noun here!
"Thousands of Plural-Noun ago, ..."

[string-split-all]

[list: "Thousands", "of", "Plural-Noun", "ago", ...]
"Thousands of Plural-Noun ago, ..."

string-split-all

[ list: "Thousands", "of", "Plural-Noun", "ago," , ... ]

Something like transform-column but for lists

[ list: "Thousands", "of", "gazebos", "ago," , ... ]
"Thousands of Plural-Noun ago, ...

string-split-all

[li: "Thousands", "of", "Plural-Noun", "ago", ...

Something like transform-column but for lists

[li: "Thousands", "of", "gazebos", "ago", ...

Needs a helper function!
"Thousands of Plural-Noun ago, ..."

string-split-all


Something like transform-column but for lists using

[list: "Thousands", "of", "gazebos", "ago," , ...]

substitute-word

"Thousands" -> "Thousands"
"Plural-Noun" -> "gazebos"
I’d write the helper function first!
fun substitute-word(w :: String) -> String:
   doc: "Substitute a random word if w is a category"
   ...
where:
   substitute-word("Thousands") is "Thousands"
   substitute-word("Plural-Noun") is ...
end

Uh oh! We don’t know what particular word it will be!
fun substitute-word(w :: String) -> String:
    doc: "Substitute a random word if w is a category"
    ...
where:
    substitute-word("Thousands") is "Thousands"
    substitute-word("Plural-Noun") is-not "Plural-Noun"
end

We know what it isn’t!
plural-nouns = [list: "gazebos", "avocados", "pandas"]

fun substitute-word(w :: String) -> String:
    doc: "Substitute a random word if w is a category"
    ...
where:
    substitute-word("Thousands") is "Thousands"
    substitute-word("Plural-Noun") is-not "Plural-Noun"
    member(plural-nouns, substitute-word("Plural-Noun")) is true
end

And we know it’s one of the right choices!
plural-nouns = [list: "gazebos", "avocados", "pandas"]

fun substitute-word(w :: String) -> String:
  doc: "Substitute a random word if w is a category"
  ...
  where:
  substitute-word("Thousands") is "Thousands"
  substitute-word("Plural-Noun") is-not "Plural-Noun"
  member(plural-nouns, substitute-word("Plural-Noun"))
    is true
end

The left part of an example can be any expression!
plural-nouns = [list: "gazebos", "avocados", "pandas"]

fun substitute-word(w :: String) -> String:
    doc: "Substitute a random word if w is a category"
    if w == "Plural-Noun":
        ...
    else:
        w
    end
end

where:
    ...
end
We need a random element of a list.

Time to check the Pyret documentation!
3.2.5 Random Numbers

```
num-random :: (max :: Number) -> Number
```

Returns a pseudo-random positive integer from 0 to \( \text{max} - 1 \).

Examples:
```
check:
  fun between(min, max):
    lam(v) { v >= min and v <= max } end
  end
for each i from range(0, 100):
  block:
    n = num-random(10)
    print(n)
    n satisfies between(0, 10 - 1)
  end
end
```

```
num-random-seed :: (seed :: Number) -> Nothing
```

Sets the random seed. Setting the seed to a particular number makes all future uses of random produce the same sequence of numbers. Useful for testing and debugging functions that have random behavior.

Examples:
```
check:
  num-random-seed(0)
  n = num-random(1000)
```
We didn’t find a built-in way to get a random element of a list, but we found a way to get a random number.

How could we use this?
.get :: (n :: Number) -> a

Returns the n-th element of the given List, or raises an error if n is out of range.

Examples:

check:
   [list: 1, 2, 3].get(0) is 1
   [list: ].get(0) raises "too large"
   [list: 1, 2, 3].get(-1) raises "invalid argument"
   end

.set :: (n :: Number, e :: a) -> List<a>

Returns a new List with the same values as the given List but with the n-th element set to the given value, or raises an error if n is out of range.

Examples:

check:
   [list: 1, 2, 3].set(0, 5) is [list: 5, 2, 3]
   [list: ].set(0, 5) raises "too large"
   end

.foldl :: (f :: (a, Base -> Base), base :: Base) -> Base

Computes f(last-elt, ... f(second-elt, f(first-elt, base))...). For empty, returns base.

In other words, .foldl uses the function f, starting with the base value, of type Base, to
With a table, we could use `.row-n` to get a specific row by its index number.

With a list, we can use `.get` to get an item.
Get a random number

Get the list element positioned at that number
plural-nouns = [list: "gazebos", "avocados", "pandas"]

fun substitute-word(w :: String) -> String:
    doc: "Substitute a random word if w is a category"
    if w == "Plural-Noun":
        rand = num-random(3)
        plural-nouns.get(rand)
    else:
        w
    end
where:
    ...
end
plural-nouns = [list: "gazebos", "avocados", "pandas"]

fun substitute-word(w :: String) -> String:
    doc: "Substitute a random word if w is a category"
    if w == "Plural-Noun":
        rand = num-random(3)
        plural-nouns.get(rand)
    else:
        w
    end
where:
    ...
end
```plaintext
plural-nouns = [list: "gazebos", "avocados", "pandas", "quokkas"]

fun substitute-word(w :: String) -> String:
  doc: "Substitute a random word if w is a category"
  if w == "Plural-Noun":
    rand = num-random(3)
    plural-nouns.get(rand)
  else:
    w
  end
where:
  ...
end
```
plural-nouns = [list: "gazebos", "avocados", "pandas", "quokkas"]

fun substitute-word(w :: String) -> String:
    doc: "Substitute a random word if w is a category"
    if w == "Plural-Noun":
        rand = num-random(length(plural-nouns))
        plural-nouns.get(rand)
    else:
        w
    end
end

where:
...
Thousands of Plural-Noun ago, there were calendars that enabled the ancient Plural-Noun to divide a year into twelve Plural-Noun, each month into Number weeks, and each week into seven Plural-Noun. At first, people told time by a sun clock, sometimes known as the Noun dial. Ultimately, they invented the great timekeeping devices of today, such as the grandfather Noun, the pocket Noun, the alarm Noun, and, of course, the Body-Part watch. Children learn about clocks and time almost before they learn their A-B- Alphabet-Letter s. They are taught that a day consists of 24 Plural-Noun, an hour has 60 Plural-Noun, and a minute has 60 Plural-Noun. By the time they are in Kindergarten, they know if the big Body-Part is at twelve and the little Body-Part is at three, that it is Number o’clock. I wish we could continue this Adjective lesson, but we’ve run out of Noun.
plural-nouns =
   [list: "gazebos", "avocados", "pandas", "quokkas"]

numbers =
   [list: "-1", "42", "a billion"]

nouns =
   [list: "apple", "computer", "borscht"]

body-parts =
   [list: "elbow", "head", "spleen"]

alphabet-letters =
   [list: "A", "C", "Z"]

adjectives =
   [list: "funky", "boring"]
fun substitute-word(w :: String) -> String:
    doc: "Substitute a random word if w is a category"
    if w == "Plural-Noun":
        rand = num-random(length(plural-nouns))
        plural-nouns.get(rand)
    else if w == "Number":
        rand = ...
    else:
        w
    end

where:
    ...
end
fun substitute-word(w :: String) -> String:
  doc: "Substitute a random word if w is a category"
  if w == "Plural-Noun":
    rand = num-random(length(plural-nouns))
    plural-nouns.get(rand)
  else if w == "Number":
    rand = ...
  else:
    w
  end
where:
  ...
end

Don’t repeat yourself!
fun rand-word(l :: List<String>) -> String:
  doc: "Return a random word in the given list"
  rand = num-random(length(l))
  l.get(rand)
where:
  member(plural-nouns, rand-word(plural-nouns)) is true
  member(numbers, rand-word(numbers)) is true
fun substitute-word(w :: String) -> String:
    doc: "Substitute a random word if w is a category"
    if w == "Plural-Noun":
        rand-word(plural-nouns)
    else if w == "Number":
        rand-word(numbers)
    else if w == "Noun":
        rand-word(nouns)
    else if w == "Body-Part":
        rand-word(body-parts)
    else if w == "Alphabet-Letter":
        rand-word(alphabet-letters)
    else if w == "Adjective":
        rand-word(adjectives)
    else:
        w
    end
end

This is still a bit repetitious – but it’s good enough for today!
Go back to the task plan.

We’ve completed our helper, and now we need to

split the input into words

run the helper on every word in the list

Similar to how `transform-column` runs a function on every row of a table.
fun mad-libs(t :: String) -> String:
    doc: "Randomly fill in the blanks in the mad libs template"
    words = string-split-all(t, " ")
    ...
end
Go back to the task plan.

We’ve completed our helper, and now we need to

- split the input into words
- run the helper on every word in the list

Similar to how `transform-column` runs a function on every row of a table.
Go back to the task plan.

We’ve completed our helper, and now we need to

- split the input into words
- run the helper on every word in the list

Similar to how `transform-column` runs a function on every row of a table.
fun mad-libs(t :: String) -> String:
  doc: "Randomly fill in the blanks in the mad libs template"
  words = string-split-all(t, " ")
  map(substitute-word, words)
...
end
Go back to the task plan.

We’ve completed our helper, and now we need to

- split the input into words
- run the helper on every word in the list

Similar to how `transform-column` runs a function on every row of a table.

Ok – are we done?
fun mad-libs(t :: String) -> String:
    doc: "Randomly fill in the blanks in the mad libs template"
    words = string-split-all(t, " ")
    map(substitute-word, words)
    ...
end

This gives us a list of strings. How can we join it back into a single string?
fun mad-libs(t :: String) -> String:
    doc: "Randomly fill in the blanks in the mad libs template"
    words = string-split-all(t, " ")
    words-sub = map(substitute-word, words)
    join-str(words-sub, " ")
end
fun mad-libs(t :: String) -> String:
    doc: "Randomly fill in the blanks in the mad libs template"
    words = string-split-all(t, " ")
    words-sub = map(substitute-word, words)
    join-str(words-sub, " ")

What do we know is true about the output?
fun mad-libs(t :: String) -> String:
    doc: "Randomly fill in the blanks in the mad libs template"
    words = string-split-all(t, " ")
    words-sub = map(substitute-word, words)
    join-str(words-sub, " ")
    where:
        mad-libs(template) is-not template
fun mad-libs(t :: String) -> String:
  doc: "Randomly fill in the blanks in the mad libs template"
  words = string-split-all(t, " ")
  words-sub = map(substitute-word, words)
  join-str(words-sub, " ")
where:
  mad-libs(template) is-not template
  string-contains(mad-libs(template), "Plural-Noun")
    is false
end
Class code:

tinyurl.com/101-2024-02-08