Key-Value Databases

• A key-value database is perhaps the conceptually simplest database paradigm currently available

• With a key-value database, values are assigned to unique identifiers called, um…keys

• You can then retrieve by key or search by key

• You can also delete by key; update is a matter of setting a new value

    …and that’s it 😊 (conceptually)

You can try one live!

• Redis, one of the most popular key-value databases currently in use, has a fiddle-like website where you can try it out:

    https://try.redis.io

• It’s a command-line interface that replicates the actual Redis CLI application
Things to try

- **GET key** will display the value at that key
- **SET key** sets the value at that key
- **DEL key** deletes that key (and thus its value)
- **KEYS pattern** will list all keys that match the given pattern (wildcards are allowed, like *)
- There’s more, but those are the basics—other commands are similar but are specialized for certain data types like numbers or hashes

Yes this is still useful

- Key-value databases are meant to be fast, in exchange for being relatively simple
- They’re meant for applications whose data needs match this simplicity
- They can also be used as supplements to more sophisticated databases
- Their speed makes them well-suited as caches—copies of other databases that are structured for faster access
Feeling some déjà vu?

- If you’re thinking “hey those look a lot like a JavaScript object or a dictionary in Python, Swift, and other languages”—yes, you aren’t that far off

- Key-value databases are pretty much those data structures, but optimized for speed and scale

- This similarity also allows us to explore making our own implementation of an in-memory key-value database—which will be our next assignment

How do we key-value thee? Let us count the ways:

<table>
<thead>
<tr>
<th>Define/initialize</th>
<th>JavaScript</th>
<th>Python</th>
<th>Swift</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>let db = {}</td>
<td>db = {}</td>
<td>var db:[type:type] = []</td>
</tr>
<tr>
<td>Get a value</td>
<td>db[key]</td>
<td>db[key]</td>
<td>db[key]</td>
</tr>
<tr>
<td>Set a value</td>
<td>db[key] = value</td>
<td>db[key] = value</td>
<td>db[key] = value</td>
</tr>
<tr>
<td>Get keys</td>
<td>Object.keys(db)</td>
<td>db.keys()</td>
<td>db.keys</td>
</tr>
<tr>
<td>Delete a key</td>
<td>delete db[key]</td>
<td>del db[key]</td>
<td>db.removeValue(forKey: key)</td>
</tr>
</tbody>
</table>

- Fire up your favorite REPL and give these a try!

- You can then use your favorite iteration technique to find matching keys

- Regular expressions help to match against patterns