

# Python Technical Assessment

Estimated time to complete: 1 - 2 hours

## 1 — Challenge: “JSON Redactor”

Write a small command-line tool that **redacts or hashes sensitive data inside an arbitrarily nested JSON document** while preserving the document’s structure and formatting.

### Functional requirements

#	Requirement	Details
1	CLI entry-point	Provide an executable script or <code>python -m json_redactor</code> interface named <code>json-redactor</code> (or similar).
2	Input	<ul style="list-style-type: none"><li>Accept a JSON file path <i>or</i> read from <code>stdin</code>.</li><li>Reject invalid JSON with a helpful error message and non-zero exit code.</li></ul>
3	Sensitive-key list	<ul style="list-style-type: none"><li>Accept a comma-separated list of keys via <code>--keys email,password,ssn</code> <i>or</i> a text file via <code>--key-file &lt;path&gt;</code>.</li><li>Matching should be case-insensitive.</li></ul>
4	Redaction modes	Two mutually exclusive flags: <code>--mask</code> — replace each sensitive value with <code>****REDACTED****</code> <code>--hash</code> — replace each sensitive value with <b>adeterministic SHA-256 hash</b> of the original value (same input ⇒ same output). Default is <code>--mask</code> .
5	Preserve structure & order	The output JSON must retain: <ul style="list-style-type: none"><li>All non-sensitive keys/values unchanged</li><li>Original key order (use <code>object_pairs_hook</code> or Python 3.7+ insertion order)</li></ul>
6	Streaming-safe	Handle files ≥ 500 MB without loading the entire file into memory (use a generator / incremental parser, e.g. <code>ijson</code> or <code>json.loads</code> on chunks).
7	Exit codes	0 on success, non-zero on any error (bad JSON, missing file, bad args, etc.).

### Nice-to-have stretch goals (optional)

- Allow regex patterns for sensitive keys (`--keys-regex`)
- Parallelise hashing with `concurrent.futures` when `--hash`

Treat this assessment as if it were a feature that you were going to commit to a production code base. Your code should be neat and readable and include comments where necessary. Use this as an opportunity to show case the standard of code you would contribute to Mission Mobile.

## 2 — Example

### Command

```
$ cat people.json | json-redactor --keys email,ssn --hash > output.json
```

Input - `people.json`

```
[
  { "name": "Anna", "email": "anna@example.com", "ssn": "123-45-6789" },
  { "name": "Ben", "email": "ben@example.com", "ssn": "987-65-4321" }
]
```

Output - `output.json`

```
[
  {
    "name": "Anna",
    "email": "06f14e0a6a3376d910...",      <- SHA-256 hash
    "ssn": "5bf64d14db16bb7c8c..."      <- SHA-256 hash
  },
  {
    "name": "Ben",
    "email": "7b9504a34aad0b772f...",
    "ssn": "e324b3399a4281cfa5..."
  }
]
```