QA Automation Testing Cheat Sheet (with Mock API)

TOOLS & LIBRARIES USED

- pytest : Python test runner
- requests : Send HTTP requests
- responses : Mock HTTP responses
- pytest-html : Generate HTML reports

TEST STRUCTURE

1. Arrange
   - Setup test data (JSON or hardcoded)
   - Setup mock responses (if needed)
2. Act
   - Send request via requests.post/get/etc.
3. Assert
   - Check status_code, keys in response, etc.

TEST DATA FORMAT (login_data.json)

```
[
  {"username": "kminchelle", "password": "0lelplR"},
  {"username": "emilys", "password": "emilyspass"}
]
```

BASIC PYTEST + REQUESTS

```
def test_login():
    response = requests.post("https://api.com/login", json={...})
    assert response.status_code == 200
```

PARAMETRIZE TEST CASES

```
@pytest.mark.parametrize("username, password", [("user", "pass"), ...])
def test_login(username, password):
    ...
```

LOAD FROM JSON

```
def load_data(file):
    with open(file) as f:
        return [(i["username"], i["password"]) for i in json.load(f)]
```

MOCK API TEST (responses)

```
@responses.activate
def test_mock_api():
    responses.add(responses.GET, "https://api.com", json={"msg": "ok"}, status=200)
    r = requests.get("https://api.com")
    assert r.json()["msg"] == "ok"
```

MOCK WITH CALLBACK

```
def callback(request):
    data = json.loads(request.body)
    if data["username"] == "admin":
        return (200, {}, json.dumps({"token": "mocked"}))
    return (401, {}, json.dumps({"error": "invalid"}))

responses.add_callback(..., callback=callback, ...)
```

GENERATE HTML REPORT

```
pytest test_file.py --html=report.html --self-contained-html --metadata tester jamei --metadata project "Mock Login Test"
```

TIPS

- @responses.activate must be above @pytest.mark.parametrize
- Use relative paths via os.path.join(__file__)
- Always validate input before mocking response
- Store test data separate from code (JSON preferred)