FOUNDATION LESSON

WEEK 3 · LESSON 10

Treat API responses as untrusted inputs

HTTP, query parameters, timeouts and errors

UNDERSTAND THE IDEA FIRST

MENTAL MODEL

Understand the idea first

Think of sending a labelled request and receiving an envelope. The status describes the delivery outcome; the body contains the actual data. A query parameter narrows the request. A timeout limits waiting for a network operation, though it is not a promise that every whole workflow finishes by that instant. Required exercises here use local fixtures and make no live requests.

ENCODE QUERY PARAMETERS SAFELY

CODE WALKTHROUGH W03-L10-S03

```
01  from urllib.parse import urlencode
02
03  params = {"region": "North East", "limit": 2}
04  query = urlencode(params)
05  print(query)
```

WHAT TO TRACK

Encode parameter values rather than manually joining arbitrary text.

PREDICT THE RESULT

Predict the output.
Explain each operation.
Pause before continuing.

ENCODE QUERY PARAMETERS SAFELY — RESULT

CODE WALKTHROUGH W03-L10-S04

```
01  from urllib.parse import urlencode
02
03  params = {"region": "North East", "limit": 2}
04  query = urlencode(params)
05  print(query)
```

WHAT TO TRACK

Encode parameter values rather than manually joining arbitrary text.

VALIDATE THE RESULT

```
region=North+East&limit=2
```

CHECK STATUS, PARSE AND VALIDATE

CODE WALKTHROUGH W03-L10-S05

```
01 import json
02 from pathlib import Path
03
04 with Path("data/mock_success.json").open("r", encoding="utf-8")
↪   as handle:
05     response = json.load(handle)
06 if response["status"] != 200:
07     raise ValueError("request did not succeed")
08 payload = json.loads(response["body"])
09 amount = payload.get("amount")
10 if type(amount) is not int or amount < 0:
11     raise ValueError("amount must be a non-negative integer")
12 print("Amount:", amount)
```

WHAT TO TRACK

Mock envelope: status 200 and a JSON body containing amount 370.

PREDICT THE RESULT

Predict the output.
Explain each operation.
Pause before continuing.

CHECK STATUS, PARSE AND VALIDATE — RESULT

CODE WALKTHROUGH W03-L10-S06

```
import json
from pathlib import Path

with Path("data/mock_success.json").open("r", encoding="utf-8")
 as handle:
    response = json.load(handle)
if response["status"] != 200:
    raise ValueError("request did not succeed")
payload = json.loads(response["body"])
amount = payload.get("amount")
if type(amount) is not int or amount < 0:
    raise ValueError("amount must be a non-negative integer")
print("Amount:", amount)
```

WHAT TO TRACK

Mock envelope: status 200 and a JSON body containing amount 370.

VALIDATE THE RESULT

```
Amount: 370
```

HTTP SUCCESS DOES NOT PROVE THE SCHEMA

CODE WALKTHROUGH W03-L10-S07

```
01 import json
02 from pathlib import Path
03
04 with Path("data/mock_bad_schema.json").open("r", encoding="utf-
↪ 8") as handle:
05     response = json.load(handle)
06 if response["status"] != 200:
07     raise ValueError("request did not succeed")
08 payload = json.loads(response["body"])
09 if not isinstance(payload, dict) or "amount" not in payload:
10     raise ValueError("response is missing amount")
11 print(payload["amount"])
```

WHAT TO TRACK

Mock status is 200, body parses, required key is absent.

PREDICT THE RESULT

Predict the output.
Explain each operation.
Pause before continuing.

HTTP SUCCESS DOES NOT PROVE THE SCHEMA — RESULT

CODE WALKTHROUGH W03-L10-S08

```
01 import json
02 from pathlib import Path
03
04 with Path("data/mock_bad_schema.json").open("r", encoding="utf-
↪ 8") as handle:
05     response = json.load(handle)
06 if response["status"] != 200:
07     raise ValueError("request did not succeed")
08 payload = json.loads(response["body"])
09 if not isinstance(payload, dict) or "amount" not in payload:
10     raise ValueError("response is missing amount")
11 print(payload["amount"])
```

WHAT TO TRACK

Mock status is 200, body parses, required key is absent.

VALIDATE THE RESULT

```
ValueError: response is
missing amount
```

PAUSE AND APPLY

LEARNER PRACTICE

Pause and apply

Write an offline response validator. Require status 200, a JSON object body and a non-negative integer amount. Test mock_success, mock_bad_schema and mock_http_error. Report the failure reason without a live request or credentials.

PREDICT · MODIFY · RUN

01 Restate the supplied rules.

02 Predict the result.

03 Modify and run the code.

04 Compare actual output.

05 Explain any difference.

RECAP AND NEXT STEP

Recap and next step

Separate request status, body parsing and schema validation. Encode query parameters, bound live network waits and avoid endless retries. All required exercises remain offline. Next, we discuss secrets and the object vocabulary used by paths and response values.

UNDERSTAND APPLY VALIDATE EXPLAIN