FOUNDATION LESSON

WEEK 3 · LESSON 9

Read structured records and check their schema

CSV, JSON and validation

UNDERSTAND THE IDEA FIRST

MENTAL MODEL

Understand the idea first

Parsing answers whether the text follows a format. Schema validation answers whether the parsed structure has the fields and types our task requires. CSV readers usually give strings for field values. JSON preserves several kinds of value, including numbers, strings, lists, objects and null, which becomes None in Python. A successful parse alone does not make a record usable.

CSV HANDLES A QUOTED COMMA

CODE WALKTHROUGH W03-L09-S03

```
01  import csv
02  from pathlib import Path
03
04  with Path("data/invoices.csv").open("r", encoding="utf-8", newl
↪   ine="") as handle:
05      rows = list(csv.DictReader(handle))
06  print(rows[0]["customer"])
07  print(type(rows[0]["amount"]).__name__)
```

WHAT TO TRACK

Required columns: invoice_id, customer, amount. CSV amount begins as text.

PREDICT THE RESULT

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

CSV HANDLES A QUOTED COMMA — RESULT

CODE WALKTHROUGH W03-L09-S04

```
01  import csv
02  from pathlib import Path
03
04  with Path("data/invoices.csv").open("r", encoding="utf-8", newl
↪   ine="") as handle:
05      rows = list(csv.DictReader(handle))
06  print(rows[0]["customer"])
07  print(type(rows[0]["amount"]).__name__)
```

WHAT TO TRACK

Required columns: invoice_id, customer, amount. CSV amount begins as text.

VALIDATE THE RESULT

```
Acme, North
str
```

CONVERT AND VALIDATE A CSV IMPORT

CODE WALKTHROUGH W03-L09-S05

```
01  import csv
02  from pathlib import Path
03
04  with Path("data/invoices.csv").open("r", encoding="utf-8", newl
↪   ine="") as handle:
05      reader = csv.DictReader(handle)
06      required = {"invoice_id", "customer", "amount"}
07      if not required.issubset(set(reader.fieldnames or [])):
08          raise ValueError("missing invoice columns")
09      total = 0
10      for row in reader:
11          amount = int(row["amount"])
12          if amount <= 0:
13              raise ValueError("invoice amount must be positive")
14          total += amount
15  print("Total:", total)
```

WHAT TO TRACK

Two invoices: 120 and 250; positive integer amounts required.

PREDICT THE RESULT

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

CONVERT AND VALIDATE A CSV IMPORT — RESULT

CODE WALKTHROUGH W03-L09-S06

```
01  import csv
02  from pathlib import Path
03
04  with Path("data/invoices.csv").open("r", encoding="utf-8", newl
↪   ine="") as handle:
05      reader = csv.DictReader(handle)
06      required = {"invoice_id", "customer", "amount"}
07      if not required.issubset(set(reader.fieldnames or [])):
08          raise ValueError("missing invoice columns")
09      total = 0
10      for row in reader:
11          amount = int(row["amount"])
12          if amount <= 0:
13              raise ValueError("invoice amount must be positive")
14          total += amount
15  print("Total:", total)
```

WHAT TO TRACK

Two invoices: 120 and 250; positive integer amounts required.

VALIDATE THE RESULT

```
Total: 370
```

VALID JSON, UNSUITABLE SCHEMA

CODE WALKTHROUGH W03-L09-S07

```
01  import json
02  from pathlib import Path
03
04  with Path("data/bad_schema.json").open("r", encoding="utf-8") a
↪   s handle:
05      record = json.load(handle)
06  if not isinstance(record, dict):
07      raise ValueError("invoice must be an object")
08  if "amount" not in record:
09      raise ValueError("missing amount")
10  print(record["amount"])
```

WHAT TO TRACK

JSON text is valid; its object lacks the required amount key.

PREDICT THE RESULT

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

VALID JSON, UNSUITABLE SCHEMA — RESULT

CODE WALKTHROUGH W03-L09-S08

```
01  import json
02  from pathlib import Path
03
04  with Path("data/bad_schema.json").open("r", encoding="utf-8") a
↪   s handle:
05      record = json.load(handle)
06  if not isinstance(record, dict):
07      raise ValueError("invoice must be an object")
08  if "amount" not in record:
09      raise ValueError("missing amount")
10  print(record["amount"])
```

WHAT TO TRACK

JSON text is valid; its object lacks the required amount key.

VALIDATE THE RESULT

```
ValueError: missing amount
```

PAUSE AND APPLY

LEARNER PRACTICE

Pause and apply

Read the supplied invoices.json file. Require a list of objects, each with an invoice_id string and a positive integer amount. Return the sum. Test the normal file, an empty list and an object missing amount. Decide and document that an empty list has total zero.

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

CSV handles quoted fields and normally supplies text. JSON supplies several structured types, but parsing does not prove a business schema. Check required keys, types and values before calculating. Next, the same distinction helps us understand API responses.

UNDERSTAND APPLY VALIDATE EXPLAIN