FOUNDATION LESSON

WEEK 3 · LESSON 12

Build one small ingestion summary object

A small class and an offline file/API exercise

UNDERSTAND THE IDEA FIRST

MENTAL MODEL

Understand the idea first

A class describes an object type; an instance is one object created from that class. Calling BatchSummary creates an instance, and __init__ initialises its state. self is the conventional first parameter referring to that instance. self dot amounts is an instance attribute. A method receives the instance automatically when called through it; you do not pass self yourself in that call.

ONE CLASS, ONE ATTRIBUTE, ONE METHOD

CODE WALKTHROUGH W03-L12-S03

```
01  class BatchSummary:
02      def __init__(self, amounts):
03          self.amounts = amounts.copy()
04
05      def total(self):
06          return sum(self.amounts)
07
08  batch = BatchSummary([120, 250])
09  print(batch.amounts)
10  print(batch.total())
```

WHAT TO TRACK

Input contract here: validated flat list of positive integers.

PREDICT THE RESULT

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

ONE CLASS, ONE ATTRIBUTE, ONE METHOD — RESULT

CODE WALKTHROUGH W03-L12-S04

```
01  class BatchSummary:
02      def __init__(self, amounts):
03          self.amounts = amounts.copy()
04
05      def total(self):
06          return sum(self.amounts)
07
08  batch = BatchSummary([120, 250])
09  print(batch.amounts)
10  print(batch.total())
```

WHAT TO TRACK

Input contract here: validated flat list of positive integers.

VALIDATE THE RESULT

```
[120, 250]
370
```

VALIDATE FILE RECORDS, THEN SUMMARISE

CODE WALKTHROUGH W03-L12-S05

```
01  import json
02  from pathlib import Path
03  from ingestion_tools import validated_amounts
04
05  class BatchSummary:
06      def __init__(self, amounts):
07          self.amounts = amounts.copy()
08
09      def total(self):
10          return sum(self.amounts)
11
12  with Path("data/invoices.json").open("r", encoding="utf-8") as
↪   handle:
13      records = json.load(handle)
14  amounts = validated_amounts(records)
15  batch = BatchSummary(amounts)
16  print("Accepted:", len(batch.amounts))
17  print("Total:", batch.total())
```

WHAT TO TRACK

Supplied dependency: validated_amounts in ingestion_tools.py.

PREDICT THE RESULT

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

VALIDATE FILE RECORDS, THEN SUMMARISE — RESULT

CODE WALKTHROUGH W03-L12-S06

```
import json
from pathlib import Path
from ingestion_tools import validated_amounts

class BatchSummary:
    def __init__(self, amounts):
        self.amounts = amounts.copy()

    def total(self):
        return sum(self.amounts)

with Path("data/invoices.json").open("r", encoding="utf-8") as handle:
    records = json.load(handle)
amounts = validated_amounts(records)
batch = BatchSummary(amounts)
print("Accepted:", len(batch.amounts))
print("Total:", batch.total())
```

WHAT TO TRACK

Supplied dependency: validated_amounts in ingestion_tools.py.

VALIDATE THE RESULT

```
Accepted: 2
Total: 370
```

A MOCK API CANNOT BYPASS VALIDATION

CODE WALKTHROUGH W03-L12-S07

```
01  import json
02  from pathlib import Path
03  from ingestion_tools import validated_amounts
04
05  with Path("data/mock_invoice_bad.json").open("r", encoding="utf
↪   -8") as handle:
06      response = json.load(handle)
07  if response["status"] != 200:
08      raise ValueError("unsuccessful status")
09  records = json.loads(response["body"])
10  print(validated_amounts(records))
```

WHAT TO TRACK

Successful mock response contains an amount with the wrong type.

PREDICT THE RESULT

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

A MOCK API CANNOT BYPASS VALIDATION — RESULT

CODE WALKTHROUGH W03-L12-S08

```
01 import json
02 from pathlib import Path
03 from ingestion_tools import validated_amounts
04
05 with Path("data/mock_invoice_bad.json").open("r", encoding="utf
↪ -8") as handle:
06     response = json.load(handle)
07 if response["status"] != 200:
08     raise ValueError("unsuccessful status")
09 records = json.loads(response["body"])
10 print(validated_amounts(records))
```

WHAT TO TRACK

Successful mock response contains an amount with the wrong type.

VALIDATE THE RESULT

```
ValueError: amount must be a
positive integer
```

PAUSE AND APPLY

LEARNER PRACTICE

Pause and apply

Use the supplied successful invoice mock. Check status 200, parse the body, validate all records and construct BatchSummary. Print its total and compare it with the file total. Test the empty list policy: an empty valid batch totals zero. Report malformed records rather than silently dropping them.

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

You can now define reusable functions, validate inputs, handle known errors and organise code into modules. You can read text, CSV and JSON, check mock API responses and protect secrets. A small class can keep related validated state and one operation together. Complete the weekly quiz and project; formative practice remains separate. Next week introduces NumPy and pandas.

UNDERSTAND APPLY VALIDATE EXPLAIN