FOUNDATION LESSON

WEEK 6 · LESSON 10

Validation, reconciliation and visible evidence

Successful execution is one check; analytical validity requires schema, business rules and reconciliations.

THE MENTAL MODEL

MENTAL MODEL

The mental model

Successful execution is one check; analytical validity requires schema, business rules and reconciliations.

A FLOATING-POINT RECONCILIATION

CODE WALKTHROUGH W06-L10-S03

```
01  import math
02  revenue=0.3;cost=0.1;profit=revenue-cost
03  print("Exact equality:",profit+cost==revenue)
04  print("Reconciles:",math.isclose(profit+cost,revenue,
05                                   rel_tol=1e-9,abs_tol=1e-9))
```

WHAT TO TRACK

A floating-point reconciliation

PREDICT THE RESULT

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

A FLOATING-POINT RECONCILIATION · RESULT

CODE WALKTHROUGH W06-L10-S04

```
01  import math
02  revenue=0.3;cost=0.1;profit=revenue-cost
03  print("Exact equality:",profit+cost==revenue)
04  print("Reconciles:",math.isclose(profit+cost,revenue,
05                                    rel_tol=1e-9,abs_tol=1e-9))
```

WHAT TO TRACK

A floating-point reconciliation

VALIDATE THE RESULT

```
Exact equality: True
Reconciles: True
```

RECONCILE THE SUPPLIED PIPELINE

CODE WALKTHROUGH W06-L10-S05

```
from pathlib import Path
import sys
sys.path.insert(0,str(Path("src").resolve()))
from pipeline_tools import compute_metrics, summarise
import pandas as pd
d=pd.DataFrame({"entity":["A","A"],"month":["2025-01","2025-02"
],"revenue":[100,120],"cost":[70,80],"cfo":[25,35],"capex":[5,1
0]})
import numpy as np
r=compute_metrics(d)
if not np.isclose(r["fcf"]+r["capex"],r["cfo"]).all():
    raise ValueError("Cash reconciliation failed")
print(summarise(r)[["revenue","profit","fcf"]].to_string())
print("Cash reconciliation passed")
```

WHAT TO TRACK

Dependency: supplied learner/src/pipeline_tools.py

PREDICT THE RESULT

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

RECONCILE THE SUPPLIED PIPELINE · RESULT

CODE WALKTHROUGH W06-L10-S06

```
from pathlib import Path
import sys
sys.path.insert(0,str(Path("src").resolve()))
from pipeline_tools import compute_metrics, summarise
import pandas as pd
d=pd.DataFrame({"entity":["A","A"],"month":["2025-01","2025-02"],"revenue":[100,120],"cost":[70,80],"cfo":[25,35],"capex":[5,10]})
import numpy as np
r=compute_metrics(d)
if not np.isclose(r["fcf"]+r["capex"],r["cfo"]).all():
    raise ValueError("Cash reconciliation failed")
print(summarise(r)[["revenue","profit","fcf"]].to_string())
print("Cash reconciliation passed")
```

WHAT TO TRACK

Dependency: supplied learner/src/pipeline_tools.py

VALIDATE THE RESULT

```
        revenue  profit  fcf
entity
A           220    70.0   45
Cash reconciliation passed
```

A DUPLICATED REPORTING KEY

CODE WALKTHROUGH W06-L10-S07

```
01  from pathlib import Path
02  import sys
03  sys.path.insert(0,str(Path("src").resolve()))
04  from pipeline_tools import compute_metrics, summarise
05  import pandas as pd
06  d=pd.DataFrame({"entity":["A","A"],"month":["2025-01","2025-02"
    ],"revenue":[100,120],"cost":[70,80],"cfo":[25,35],"capex":[5,1
    0]})
07  d=pd.concat([d,d.iloc[[0]]],ignore_index=True)
08  compute_metrics(d)
```

WHAT TO TRACK

A duplicated reporting key

PREDICT THE RESULT

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

A DUPLICATED REPORTING KEY · RESULT

CODE WALKTHROUGH W06-L10-S08

```
01  from pathlib import Path
02  import sys
03  sys.path.insert(0,str(Path("src").resolve()))
04  from pipeline_tools import compute_metrics, summarise
05  import pandas as pd
06  d=pd.DataFrame({"entity":["A","A"],"month":["2025-01","2025-02"
↪   ],"revenue":[100,120],"cost":[70,80],"cfo":[25,35],"capex":[5,1
↪   0]})
07  d=pd.concat([d,d.iloc[[0]]],ignore_index=True)
08  compute_metrics(d)
```

WHAT TO TRACK

A duplicated reporting key

VALIDATE THE RESULT

```
ValueError: Empty data or
repeated entity-month
```

A LABELLED QUARTERLY EVIDENCE CHART

CODE WALKTHROUGH W06-L10-S09

```
01 import matplotlib
02 matplotlib.use("Agg")
03 import matplotlib.pyplot as plt
04 profit={"A":140,"B":105}
05 fig,ax=plt.subplots(figsize=(12,6.75))
06 ax.bar(list(profit),list(profit.values()))
07 ax.set(title="Quarter profit · 2025 Q1 · synthetic data",
08        xlabel="Entity",ylabel="Profit (EUR)")
09 ax.set_ylim(bottom=0)
10 fig.tight_layout()
11 fig.savefig("figure.png",dpi=320)
12 plt.close(fig)
13 print("Quarter profit EUR:",sum(profit.values()))
```

WHAT TO TRACK

2025 Q1 capstone variant A; A=140 EUR, B=105 EUR

PREDICT THE RESULT

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

A LABELLED QUARTERLY EVIDENCE CHART · RESULT

CODE WALKTHROUGH W06-L10-S10

```
01 import matplotlib
02 matplotlib.use("Agg")
03 import matplotlib.pyplot as plt
04 profit={"A":140,"B":105}
05 fig,ax=plt.subplots(figsize=(12,6.75))
06 ax.bar(list(profit),list(profit.values()))
07 ax.set(title="Quarter profit · 2025 Q1 · synthetic data",
08        xlabel="Entity",ylabel="Profit (EUR)")
09 ax.set_ylim(bottom=0)
10 fig.tight_layout()
11 fig.savefig("figure.png",dpi=320)
12 plt.close(fig)
13 print("Quarter profit EUR:",sum(profit.values()))
```

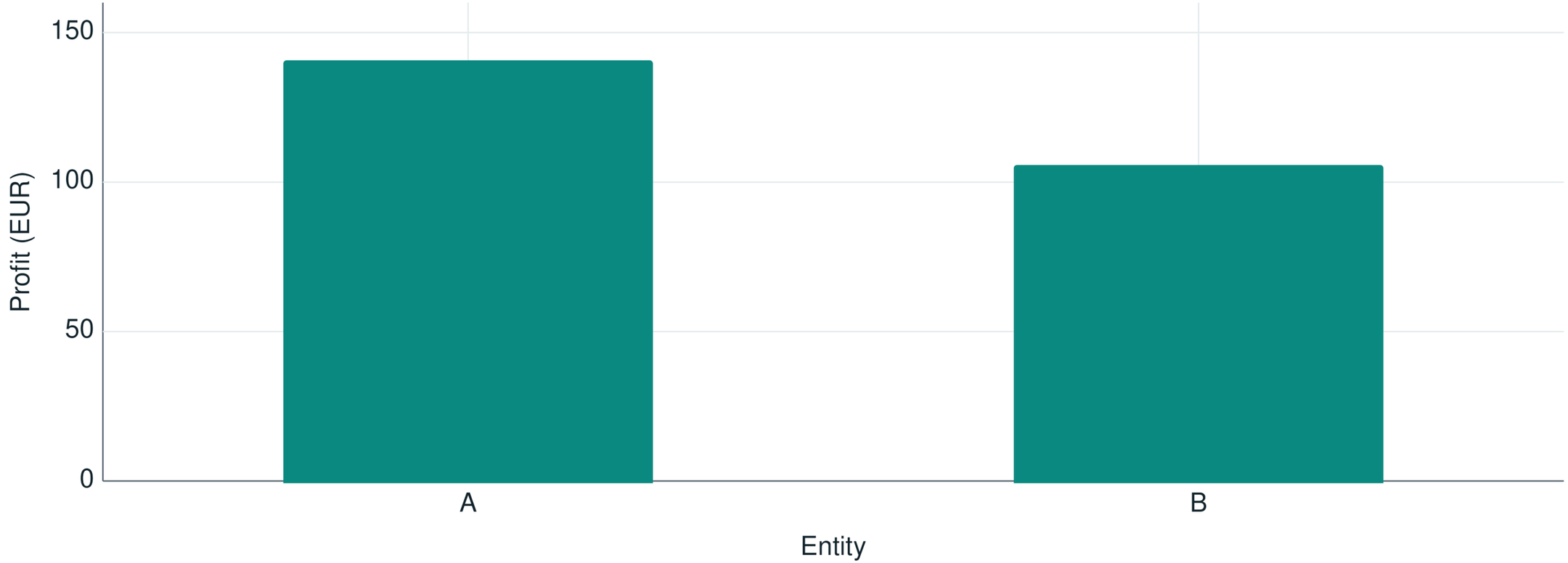
WHAT TO TRACK

2025 Q1 capstone variant A; A=140 EUR, B=105 EUR

VALIDATE THE RESULT

```
Quarter profit EUR: 245
```

VISIBLE CHART RESULT



GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Check CFO=50, CapEx=12, FCF=38 using math.isclose and a clear failure message if it does not reconcile.

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.

INDEPENDENT PRACTICE

LEARNER PRACTICE

Independent practice

Use the supplied module and confirm that changing the first cost to -1 raises ValueError.

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.

CHECK YOUR UNDERSTANDING

MENTAL MODEL

Check your understanding

Diagnostic: Does assert alone provide permanent production-input validation? No. Optimised Python can disable assertions; use explicit guarded errors for input rules.

RECAP AND NEXT STEP

Recap and next step

Validate grain, coverage and amounts, then reconcile calculated identities with explicit tolerances. Tables and charts must use the same verified results.

UNDERSTAND APPLY VALIDATE EXPLAIN