FOUNDATION LESSON

WEEK 6 · LESSON 1

Percentage changes and compounded returns

A simple return divides a change by its prior positive value; consecutive returns compound by multiplication.

THE MENTAL MODEL

MENTAL MODEL

The mental model

A simple return divides a change by its prior positive value; consecutive returns compound by multiplication.

TWO OPPOSITE PRICE MOVES

CODE WALKTHROUGH W06-L01-S03

```
01  import pandas as pd
02  p=pd.Series([100.0,110.0,100.0])
03  r=p.pct_change(fill_method=None)
04  print("Returns:",r.round(6).tolist())
05  print("Cumulative:",round((1+r.iloc[1:]).prod(skipna=False)-1,6
↪   ))
```

WHAT TO TRACK

Two opposite price moves

PREDICT THE RESULT

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

TWO OPPOSITE PRICE MOVES · RESULT

CODE WALKTHROUGH W06-L01-S04

```
01  import pandas as pd
02  p=pd.Series([100.0,110.0,100.0])
03  r=p.pct_change(fill_method=None)
04  print("Returns:",r.round(6).tolist())
05  print("Cumulative:",round((1+r.iloc[1:]).prod(skipna=False)-1,6
↪   ))
```

WHAT TO TRACK

Two opposite price moves

VALIDATE THE RESULT

```
Returns: [nan, 0.1,
-0.090909]
Cumulative: 0.0
```

A MISSING INTERMEDIATE PRICE

CODE WALKTHROUGH W06-L01-S05

```
01  import pandas as pd
02  p=pd.Series([100.0,110.0,None,121.0,125.0])
03  prior=p.shift(1)
04  r=p.pct_change(fill_method=None).where((prior>0)&(p>0))
05  chain=(1+r.iloc[1:]).cumprod(skipna=False)-1
06  print("Returns:",r.round(4).tolist())
07  print("Strict chain:",chain.round(4).tolist())
08  print("Endpoint only:",p.iloc[-1]/p.iloc[0]-1)
```

WHAT TO TRACK

A missing intermediate price

PREDICT THE RESULT

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

A MISSING INTERMEDIATE PRICE · RESULT

CODE WALKTHROUGH W06-L01-S06

```
01  import pandas as pd
02  p=pd.Series([100.0,110.0,None,121.0,125.0])
03  prior=p.shift(1)
04  r=p.pct_change(fill_method=None).where((prior>0)&(p>0))
05  chain=(1+r.iloc[1:]).cumprod(skipna=False)-1
06  print("Returns:",r.round(4).tolist())
07  print("Strict chain:",chain.round(4).tolist())
08  print("Endpoint only:",p.iloc[-1]/p.iloc[0]-1)
```

WHAT TO TRACK

A missing intermediate price

VALIDATE THE RESULT

```
Returns: [nan, 0.1, nan,
nan, 0.0331]
Strict chain: [0.1, nan,
nan, nan]
Endpoint only: 0.25
```

ZERO AND NEGATIVE PRIOR VALUES

CODE WALKTHROUGH W06-L01-S07

```
01  import pandas as pd
02  p=pd.Series([0.0,10.0,-5.0,20.0])
03  prior=p.shift(1)
04  safe=p.where(p>0)/prior.where(prior>0)-1
05  print(safe.tolist())
```

WHAT TO TRACK

Zero and negative prior values

PREDICT THE RESULT

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

ZERO AND NEGATIVE PRIOR VALUES · RESULT

CODE WALKTHROUGH W06-L01-S08

```
01  import pandas as pd
02  p=pd.Series([0.0,10.0,-5.0,20.0])
03  prior=p.shift(1)
04  safe=p.where(p>0)/prior.where(prior>0)-1
05  print(safe.tolist())
```

WHAT TO TRACK

Zero and negative prior values

VALIDATE THE RESULT

```
[nan, nan, nan, nan]
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

For prices [50,55,60], calculate simple returns and a strict cumulative price-only return.

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

For [100,None,120,126], print guarded returns and a strict chain that remains unknown after the gap.

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 a ten percent rise followed by a ten percent fall return to the start? No. The second change uses a different denominator.

RECAP AND NEXT STEP

Recap and next step

State fraction versus percent, guard prior values, disable implicit filling, and compound consecutive returns under an explicit missing-data policy.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 6 · LESSON 2

Annualisation, margins and return on assets

Frequency and denominator determine the meaning of a ratio.

THE MENTAL MODEL

MENTAL MODEL

The mental model

Frequency and denominator determine the meaning of a ratio.

AN ANNUAL-EQUIVALENT RATE

CODE WALKTHROUGH W06-L02-S03

```
01 monthly_rate=0.01
02 annual_equivalent=(1+monthly_rate)**12-1
03 print("Monthly percent:",monthly_rate*100)
04 print("Annual-equivalent percent:",round(annual_equivalent*100,
↪  3))
05 print("Assumption: same monthly rate repeated; not a forecast")
```

WHAT TO TRACK

An annual-equivalent rate

PREDICT THE RESULT

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

AN ANNUAL-EQUIVALENT RATE · RESULT

CODE WALKTHROUGH W06-L02-S04

```
01 monthly_rate=0.01
02 annual_equivalent=(1+monthly_rate)**12-1
03 print("Monthly percent:",monthly_rate*100)
04 print("Annual-equivalent percent:",round(annual_equivalent*100,
↪ 3))
05 print("Assumption: same monthly rate repeated; not a forecast")
```

WHAT TO TRACK

An annual-equivalent rate

VALIDATE THE RESULT

```
Monthly percent: 1.0
Annual-equivalent percent:
12.683
Assumption: same monthly
rate repeated; not a
forecast
```

PROFITABILITY WITH EXPLICIT DENOMINATORS

CODE WALKTHROUGH W06-L02-S05

```
01  revenue=1000; cogs=600; operating_cost=200
02  net_profit=120; assets_begin=900; assets_end=1100
03  gross_profit=revenue-cogs
04  operating_profit=gross_profit-operating_cost
05  average_assets=(assets_begin+assets_end)/2
06  print("Gross margin:",gross_profit/revenue)
07  print("Operating margin:",operating_profit/revenue)
08  print("Net margin:",net_profit/revenue)
09  print("Annual ROA:",net_profit/average_assets)
```

WHAT TO TRACK

Profitability with explicit denominators

PREDICT THE RESULT

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

PROFITABILITY WITH EXPLICIT DENOMINATORS · RESULT

CODE WALKTHROUGH W06-L02-S06

```
revenue=1000; cogs=600; operating_cost=200
net_profit=120; assets_begin=900; assets_end=1100
gross_profit=revenue-cogs
operating_profit=gross_profit-operating_cost
average_assets=(assets_begin+assets_end)/2
print("Gross margin:",gross_profit/revenue)
print("Operating margin:",operating_profit/revenue)
print("Net margin:",net_profit/revenue)
print("Annual ROA:",net_profit/average_assets)
```

WHAT TO TRACK

Profitability with explicit denominators

VALIDATE THE RESULT

```
Gross margin: 0.4
Operating margin: 0.2
Net margin: 0.12
Annual ROA: 0.12
```

A ZERO-REVENUE BOUNDARY

CODE WALKTHROUGH W06-L02-S07

```
01  def ratio(numerator,denominator):
02      return None if denominator<=0 else numerator/denominator
03  print("Margin:",ratio(-20,0))
04  print("ROA:",ratio(10,100))
```

WHAT TO TRACK

A zero-revenue boundary

PREDICT THE RESULT

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

A ZERO-REVENUE BOUNDARY · RESULT

CODE WALKTHROUGH W06-L02-S08

```
01  def ratio(numerator,denominator):
02      return None if denominator<=0 else numerator/denominator
03  print("Margin:",ratio(-20,0))
04  print("ROA:",ratio(10,100))
```

WHAT TO TRACK

A zero-revenue boundary

VALIDATE THE RESULT

```
Margin: None
ROA: 0.1
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Compute annual-equivalent growth for a hypothetical 2% quarterly rate using four periods and state the assumption.

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

Annual net profit is 80; beginning/end assets are 600/1000. Calculate average-asset ROA and label its period.

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: Can quarterly profit divided by average quarter assets be labelled annual ROA without another step? No. Its numerator covers only that quarter.

RECAP AND NEXT STEP

Recap and next step

State frequency, profit period and denominator. Label annual-equivalent assumptions and use the documented average-asset convention for ROA.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 6 · LESSON 3

Liquidity and leverage ratios

Liquidity compares near-term resources with obligations; leverage compares defined debt with a stated capital base.

THE MENTAL MODEL

MENTAL MODEL

The mental model

Liquidity compares near-term resources with obligations; leverage compares defined debt with a stated capital base.

THREE LIQUIDITY VIEWS

CODE WALKTHROUGH W06-L03-S03

```
current_assets=300; inventory=100; cash=80
current_liabilities=200
print("Current ratio:",current_assets/current_liabilities)
print("Simplified quick ratio:",
      (current_assets-inventory)/current_liabilities)
print("Cash ratio:",cash/current_liabilities)
```

WHAT TO TRACK

Three liquidity views

PREDICT THE RESULT

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

THREE LIQUIDITY VIEWS · RESULT

CODE WALKTHROUGH W06-L03-S04

```
current_assets=300; inventory=100; cash=80
current_liabilities=200
print("Current ratio:",current_assets/current_liabilities)
print("Simplified quick ratio:",
      (current_assets-inventory)/current_liabilities)
print("Cash ratio:",cash/current_liabilities)
```

WHAT TO TRACK

Three liquidity views

VALIDATE THE RESULT

```
Current ratio: 1.5
Simplified quick ratio: 1.0
Cash ratio: 0.4
```

DEBT AND INTEREST COVERAGE

CODE WALKTHROUGH W06-L03-S05

```
01  interest_bearing_debt=240; equity=160; assets=500
02  annual_operating_profit=60; annual_interest=12
03  print("Debt/equity:",interest_bearing_debt/equity)
04  print("Debt/assets:",interest_bearing_debt/assets)
05  print("Annual interest coverage:",
06        annual_operating_profit/annual_interest)
```

WHAT TO TRACK

Debt and interest coverage

PREDICT THE RESULT

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

DEBT AND INTEREST COVERAGE · RESULT

CODE WALKTHROUGH W06-L03-S06

```
01  interest_bearing_debt=240; equity=160; assets=500
02  annual_operating_profit=60; annual_interest=12
03  print("Debt/equity:",interest_bearing_debt/equity)
04  print("Debt/assets:",interest_bearing_debt/assets)
05  print("Annual interest coverage:",
06        annual_operating_profit/annual_interest)
```

WHAT TO TRACK

Debt and interest coverage

VALIDATE THE RESULT

```
Debt/equity: 1.5
Debt/assets: 0.48
Annual interest coverage:
5.0
```

NONPOSITIVE EQUITY AND ZERO INTEREST

CODE WALKTHROUGH W06-L03-S07

```
01  def positive_ratio(n,d):
02      return None if d<=0 else n/d
03  print("Debt/equity:",positive_ratio(100,-20))
04  print("Interest coverage:",positive_ratio(50,0))
```

WHAT TO TRACK

Nonpositive equity and zero interest

PREDICT THE RESULT

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

NONPOSITIVE EQUITY AND ZERO INTEREST · RESULT

CODE WALKTHROUGH W06-L03-S08

```
01  def positive_ratio(n,d):
02      return None if d<=0 else n/d
03  print("Debt/equity:",positive_ratio(100,-20))
04  print("Interest coverage:",positive_ratio(50,0))
```

WHAT TO TRACK

Nonpositive equity and zero interest

VALIDATE THE RESULT

```
Debt/equity: None
Interest coverage: None
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

At one date, current assets are 180, inventory 30, liabilities 90. Calculate current and simplified quick ratios.

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

Debt is 80 and equity is zero. Print Undefined: nonpositive equity rather than dividing.

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: Is debt-to-assets automatically total liabilities divided by assets in this course? No. The numerator here is explicitly interest-bearing debt.

RECAP AND NEXT STEP

Recap and next step

Align dates and periods, define debt and quick assets, and guard denominator boundaries. These simplified ratios require contextual interpretation.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 6 · LESSON 4

Cash-flow measures and rolling windows

Cash flow measures money movements; a rolling window summarises a stated set of ordered observations.

THE MENTAL MODEL

MENTAL MODEL

The mental model

Cash flow measures money movements; a rolling window summarises a stated set of ordered observations.

A STATED CAPEX CONVENTION

CODE WALKTHROUGH W06-L04-S03

```
01  profit=80; operating_cash_flow=65; capex=20
02  if capex<0:
03      raise ValueError("CapEx must be a nonnegative outflow")
04  print("Profit:",profit)
05  print("Free cash flow:",operating_cash_flow-capex)
```

WHAT TO TRACK

A stated CapEx convention

PREDICT THE RESULT

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

A STATED CAPEX CONVENTION · RESULT

CODE WALKTHROUGH W06-L04-S04

```
profit=80; operating_cash_flow=65; capex=20
if capex<0:
    raise ValueError("CapEx must be a nonnegative outflow")
print("Profit:",profit)
print("Free cash flow:",operating_cash_flow-capex)
```

WHAT TO TRACK

A stated CapEx convention

VALIDATE THE RESULT

```
Profit: 80
Free cash flow: 45
```

GROUPED THREE-OBSERVATION MEANS

CODE WALKTHROUGH W06-L04-S05

```
01  import pandas as pd
02  d=pd.DataFrame({"entity":["A","A","A","B","B","B"],
03   "month":[1,2,3,1,2,3],"cfo":[10,20,30,100,200,300]})
04  d=d.sort_values(["entity","month"])
05  def three_observation_mean(values):
06      return values.rolling(3,min_periods=3).mean()
07  d["rolling_cfo"]=d.groupby("entity")["cfo"].transform(
08      three_observation_mean)
09  print(d.to_string(index=False))
```

WHAT TO TRACK

Grouped three-observation means

PREDICT THE RESULT

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

GROUPED THREE-OBSERVATION MEANS · RESULT

CODE WALKTHROUGH W06-L04-S06 1/2

```
01  import pandas as pd
02  d=pd.DataFrame({"entity":["A","A","A","B","B","B"],
03   "month":[1,2,3,1,2,3],"cfo":[10,20,30,100,200,300]})
04  d=d.sort_values(["entity","month"])
05  def three_observation_mean(values):
06      return values.rolling(3,min_periods=3).mean()
07  d["rolling_cfo"]=d.groupby("entity")["cfo"].transform(
08      three_observation_mean)
09  print(d.to_string(index=False))
```

WHAT TO TRACK

Grouped three-observation means

VALIDATE THE RESULT

```
entity  month  cfo
rolling_cfo
     A      1   10
NaN
     A      2   20
NaN
     A      3   30
20.0
     B      1  100
NaN
     B      2  200
NaN
```

GROUPED THREE-OBSERVATION MEANS · RESULT

CODE WALKTHROUGH W06-L04-S06 2/2

WHAT TO TRACK

Grouped three-observation means

VALIDATE THE RESULT

```
     B       3  300
200.0
```

THREE ROWS WITH A MISSING MONTH

CODE WALKTHROUGH W06-L04-S07

```
01  months=[1,3,4]
02  if any(b-a!=1 for a,b in zip(months,months[1:])):
03      raise ValueError("Monthly coverage is not consecutive")
```

WHAT TO TRACK

Three rows with a missing month

PREDICT THE RESULT

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

THREE ROWS WITH A MISSING MONTH · RESULT

CODE WALKTHROUGH W06-L04-S08

```
01 months=[1,3,4]
02 if any(b-a!=1 for a,b in zip(months,months[1:])):
03     raise ValueError("Monthly coverage is not consecutive")
```

WHAT TO TRACK

Three rows with a missing month

VALIDATE THE RESULT

```
ValueError: Monthly coverage
is not consecutive
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Operating cash flow is 90 and positive-outflow CapEx is 25. Compute FCF and explain the subtraction.

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

For [10,20,30,40], compute a three-observation rolling mean with min_periods=3.

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 rolling(3) guarantee a three-calendar-month window? No. It ordinarily counts three observations.

RECAP AND NEXT STEP

Recap and next step

Separate profit from cash flow, subtract nonnegative CapEx outflows, and check entity boundaries and calendar coverage before rolling.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 6 · LESSON 5

Comparable entities and deterministic scenarios

Scale, period and assumptions determine whether a comparison is meaningful.

THE MENTAL MODEL

MENTAL MODEL

The mental model

Scale, period and assumptions determine whether a comparison is meaningful.

PROFIT AND MARGIN RANKINGS

CODE WALKTHROUGH W06-L05-S03

```
01  import pandas as pd
02  d=pd.DataFrame({"entity":["A","B"],
03   "revenue":[1000,400],"profit":[120,80]})
04  d["margin"]=d["profit"]/d["revenue"]
05  print(d.to_string(index=False))
```

WHAT TO TRACK

Profit and margin rankings

PREDICT THE RESULT

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

PROFIT AND MARGIN RANKINGS · RESULT

CODE WALKTHROUGH W06-L05-S04

```
01  import pandas as pd
02  d=pd.DataFrame({"entity":["A","B"],
03   "revenue":[1000,400],"profit":[120,80]})
04  d["margin"]=d["profit"]/d["revenue"]
05  print(d.to_string(index=False))
```

WHAT TO TRACK

Profit and margin rankings

VALIDATE THE RESULT

```
entity  revenue  profit
margin
     A     1000     120
0.12
     B      400      80
0.20
```

A TRANSPARENT COST SCENARIO

CODE WALKTHROUGH W06-L05-S05

```
revenue=1000; baseline_cost=800
scenarios={"base":1.0,"cost_up_10pct":1.1,"cost_down_10pct":.9}
for name,factor in scenarios.items():
    profit=revenue-baseline_cost*factor
    print(name,"profit",round(profit,2),"margin",round(profit/revenue,3))
print("Revenue held fixed; deterministic scenarios, not forecasts")
```

WHAT TO TRACK

A transparent cost scenario

PREDICT THE RESULT

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

A TRANSPARENT COST SCENARIO · RESULT

CODE WALKTHROUGH W06-L05-S06

```
revenue=1000; baseline_cost=800
scenarios={"base":1.0,"cost_up_10pct":1.1,"cost_down_10pct":.9}
for name,factor in scenarios.items():
    profit=revenue-baseline_cost*factor
    print(name,"profit",round(profit,2),"margin",round(profit/revenue,3))
print("Revenue held fixed; deterministic scenarios, not forecasts")
```

WHAT TO TRACK

A transparent cost scenario

VALIDATE THE RESULT

```
base profit 200.0 margin 0.2
cost_up_10pct profit 120.0 margin 0.12
cost_down_10pct profit 280.0 margin 0.28
Revenue held fixed; deterministic scenarios, not forecasts
```

AN INVALID INDEX BASELINE

CODE WALKTHROUGH W06-L05-S07

```
01  values=[0,20,30]
02  if values[0]<=0:
03      raise ValueError("Index baseline must be positive")
04  print([v/values[0]*100 for v in values])
```

WHAT TO TRACK

An invalid index baseline

PREDICT THE RESULT

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

AN INVALID INDEX BASELINE · RESULT

CODE WALKTHROUGH W06-L05-S08

```
01  values=[0,20,30]
02  if values[0]<=0:
03      raise ValueError("Index baseline must be positive")
04  print([v/values[0]*100 for v in values])
```

WHAT TO TRACK

An invalid index baseline

VALIDATE THE RESULT

```
ValueError: Index baseline
must be positive
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Compare revenue/profit pairs 200/30 and 500/50. Print margins and identify the higher margin.

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

Revenue is 600 and cost 450. Apply a 1.2 cost factor without changing the baseline variable.

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: Is a scenario with a ten percent cost increase a forecast that costs will rise by ten percent? No. It is a stated hypothetical input.

RECAP AND NEXT STEP

Recap and next step

Compare like periods and definitions, distinguish amounts from margins, and describe scenarios through their explicit assumptions.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 6 · LESSON 6

A reusable analytical pipeline

Validate, compute and summarise through small reusable functions.

THE MENTAL MODEL

MENTAL MODEL

The mental model

Validate, compute and summarise through small reusable functions.

RUN THE SUPPLIED MODULE

CODE WALKTHROUGH W06-L06-S03

```
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  result=compute_metrics(d)
08  print(result[["profit","margin","fcf"]].to_string(index=False))
```

WHAT TO TRACK

Dependency: supplied learner/src/pipeline_tools.py

PREDICT THE RESULT

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

RUN THE SUPPLIED MODULE · RESULT

CODE WALKTHROUGH W06-L06-S04

```
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  result=compute_metrics(d)
08  print(result[["profit","margin","fcf"]].to_string(index=False))
```

WHAT TO TRACK

Dependency: supplied learner/src/pipeline_tools.py

VALIDATE THE RESULT

```
 profit   margin  fcf
   30.0 0.300000   20
   40.0 0.333333   25
```

A GROUP MARGIN FROM TOTALS

CODE WALKTHROUGH W06-L06-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":[5,10]})
base=summarise(compute_metrics(d))
scenario=summarise(compute_metrics(d,cost_factor=1.1))
print("Base:",base[["profit","margin"]].round(4).to_dict())
print("Scenario:",scenario[["profit","margin"]].round(4).to_dict())
```

WHAT TO TRACK

Same supplied module; cost multiplier only

PREDICT THE RESULT

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

A GROUP MARGIN FROM TOTALS · RESULT

CODE WALKTHROUGH W06-L06-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]})
base=summarise(compute_metrics(d))
scenario=summarise(compute_metrics(d,cost_factor=1.1))
print("Base:",base[["profit","margin"]].round(4).to_dict())
print("Scenario:",scenario[["profit","margin"]].round(4).to_dict())
```

WHAT TO TRACK

Same supplied module; cost multiplier only

VALIDATE THE RESULT

```
Base: {'profit': {'A': 70.0}, 'margin': {'A': 0.3182}}
Scenario: {'profit': {'A': 55.0}, 'margin': {'A': 0.25}}
```

REJECT A NEGATIVE CAPEX OUTFLOW

CODE WALKTHROUGH W06-L06-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.loc[0,"capex"]=-5
08  compute_metrics(d)
```

WHAT TO TRACK

Reject a negative CapEx outflow

PREDICT THE RESULT

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

REJECT A NEGATIVE CAPEX OUTFLOW · RESULT

CODE WALKTHROUGH W06-L06-S08

```
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]})
d.loc[0,"capex"]=-5
compute_metrics(d)
```

WHAT TO TRACK

Reject a negative CapEx outflow

VALIDATE THE RESULT

```
ValueError: Revenue, cost and CapEx must be nonnegative
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Using the supplied module, compute the two-month summary at cost_factor=1.2 and print profit.

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

Using the supplied module, set the first revenue to zero. Confirm its row margin is missing while profit still records costs.

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: Should group margin be the unweighted mean of row margins when revenue differs? No. Use total profit divided by total revenue for this question.

RECAP AND NEXT STEP

Recap and next step

Reuse one validated function chain for baseline and scenarios. This formative rehearsal supplies functions for the single required final capstone.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 6 · LESSON 7

A capstone question and portable project structure

A scoped question determines inputs, reporting grain, outputs and limitations.

THE MENTAL MODEL

MENTAL MODEL

The mental model

A scoped question determines inputs, reporting grain, outputs and limitations.

CREATE THE PROJECT FOLDERS

CODE WALKTHROUGH W06-L07-S03

```
01  from pathlib import Path
02  root=Path("capstone")
03  for folder in ["data/raw","data/processed","src","outputs","con
↪   fig"]:
04      (root/folder).mkdir(parents=True,exist_ok=True)
05  print((root/"data"/"raw").as_posix())
06  print((root/"outputs").exists())
```

WHAT TO TRACK

Create the project folders

PREDICT THE RESULT

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

CREATE THE PROJECT FOLDERS · RESULT

CODE WALKTHROUGH W06-L07-S04

```
01  from pathlib import Path
02  root=Path("capstone")
03  for folder in ["data/raw","data/processed","src","outputs","con
↪   fig"]:
04      (root/folder).mkdir(parents=True,exist_ok=True)
05  print((root/"data"/"raw").as_posix())
06  print((root/"outputs").exists())
```

WHAT TO TRACK

Create the project folders

VALIDATE THE RESULT

```
capstone/data/raw
True
```

A BOUNDED ANALYTICAL QUESTION

CODE WALKTHROUGH W06-L07-S05

```
01  question={"period":"2025 Q1","entities":["A","B"],
02   "measures":["revenue","profit","fcf"],"cost_factor":1.1}
03  print("Period:",question["period"])
04  print("Measures:",", ".join(question["measures"]))
05  print("Scenario changes cost only:",question["cost_factor"])
```

WHAT TO TRACK

A bounded analytical question

PREDICT THE RESULT

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

A BOUNDED ANALYTICAL QUESTION · RESULT

CODE WALKTHROUGH W06-L07-S06

```
01 question={"period":"2025 Q1","entities":["A","B"],
02  "measures":["revenue","profit","fcf"],"cost_factor":1.1}
03 print("Period:",question["period"])
04 print("Measures:",", ".join(question["measures"]))
05 print("Scenario changes cost only:",question["cost_factor"])
```

WHAT TO TRACK

A bounded analytical question

VALIDATE THE RESULT

```
Period: 2025 Q1
Measures: revenue, profit,
fcf
Scenario changes cost only:
1.1
```

MISSING REQUIRED INPUT

CODE WALKTHROUGH W06-L07-S07

```
01  from pathlib import Path
02  source=Path("data/raw/flows.csv")
03  if not source.is_file():
04      raise FileNotFoundError("Place flows.csv in data/raw before
↪    running")
```

WHAT TO TRACK

Missing required input

PREDICT THE RESULT

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

MISSING REQUIRED INPUT · RESULT

CODE WALKTHROUGH W06-L07-S08

```
01  from pathlib import Path
02  source=Path("data/raw/flows.csv")
03  if not source.is_file():
04      raise FileNotFoundError("Place flows.csv in data/raw before
↪    running")
```

WHAT TO TRACK

Missing required input

VALIDATE THE RESULT

```
FileNotFoundError: Place
flows.csv in data/raw before
running
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Create data/raw and outputs below a project folder named practice. Print both portable paths.

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

Write a scoped question dictionary for two branches during February with revenue and profit only.

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: Should rerunning the project delete raw data to start clean? No.
Regenerate outputs while preserving original inputs.

RECAP AND NEXT STEP

Recap and next step

Scope the capstone around available evidence. Separate raw files, source, configuration and outputs, and use portable paths.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 6 · LESSON 8

README documentation and data provenance

Provenance records origin and identity; the README explains how to reproduce and interpret the analysis.

THE MENTAL MODEL

MENTAL MODEL

The mental model

Provenance records origin and identity; the README explains how to reproduce and interpret the analysis.

WRITE EXPLICIT CONVENTIONS

CODE WALKTHROUGH W06-L08-S03

```
01  from pathlib import Path
02  text="""# Fictional quarterly business analysis
03  Run from the project root with the supplied dependencies.
04  Units: EUR. Period: 2025 Q1.
05  CapEx >= 0 denotes a cash outflow; FCF = CFO - CapEx.
06  ROA = quarter profit / mean(beginning assets, ending assets).
07  ROA is quarterly. Data are synthetic, not investment evidence.
08  """
09  Path("README.md").write_text(text,encoding="utf-8")
10  print(Path("README.md").read_text(encoding="utf-8"))
```

WHAT TO TRACK

Write explicit conventions

PREDICT THE RESULT

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

WRITE EXPLICIT CONVENTIONS · RESULT

CODE WALKTHROUGH W06-L08-S04 1/2

```
01 from pathlib import Path
02 text="""# Fictional quarterly business analysis
03 Run from the project root with the supplied dependencies.
04 Units: EUR. Period: 2025 Q1.
05 CapEx >= 0 denotes a cash outflow; FCF = CFO - CapEx.
06 ROA = quarter profit / mean(beginning assets, ending assets).
07 ROA is quarterly. Data are synthetic, not investment evidence.
08 """
09 Path("README.md").write_text(text,encoding="utf-8")
10 print(Path("README.md").read_text(encoding="utf-8"))
```

WHAT TO TRACK

Write explicit conventions

VALIDATE THE RESULT

```
# Fictional quarterly
business analysis
Run from the project root
with the supplied
dependencies.
Units: EUR. Period: 2025 Q1.
CapEx >= 0 denotes a cash
outflow; FCF = CFO - CapEx.
ROA = quarter profit /
mean(beginning assets,
ending assets).
ROA is quarterly. Data are
```

WRITE EXPLICIT CONVENTIONS · RESULT

CODE WALKTHROUGH W06-L08-S04 2/2

WHAT TO TRACK

Write explicit conventions

VALIDATE THE RESULT

synthetic, not investment evidence.

HASH AN ACTUAL INPUT FILE

CODE WALKTHROUGH W06-L08-S05

```
01  from pathlib import Path
02  import hashlib,json
03  p=Path("flows.csv")
04  p.write_text("entity,revenue\nA,100\n",encoding="utf-8")
05  record={"file":p.name,"source":"synthetic teaching data",
06          "sha256":hashlib.sha256(p.read_bytes()).hexdigest()}
07  Path("w06_data_provenance.json").write_text(
08      json.dumps(record,indent=2),encoding="utf-8")
09  print("Source:",record["source"])
10  print("SHA-256 length:",len(record["sha256"]))
```

WHAT TO TRACK

Hash an actual input file

PREDICT THE RESULT

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

HASH AN ACTUAL INPUT FILE · RESULT

CODE WALKTHROUGH W06-L08-S06

```
from pathlib import Path
import hashlib,json
p=Path("flows.csv")
p.write_text("entity,revenue\nA,100\n",encoding="utf-8")
record={"file":p.name,"source":"synthetic teaching data",
        "sha256":hashlib.sha256(p.read_bytes()).hexdigest()}
Path("w06_data_provenance.json").write_text(
    json.dumps(record,indent=2),encoding="utf-8")
print("Source:",record["source"])
print("SHA-256 length:",len(record["sha256"]))
```

WHAT TO TRACK

Hash an actual input file

VALIDATE THE RESULT

```
Source: synthetic teaching
data
SHA-256 length: 64
```

A CHANGED FILE HAS A CHANGED IDENTITY

CODE WALKTHROUGH W06-L08-S07

```
01  import hashlib
02  before=b"A,100\n"
03  after=b"A,101\n"
04  same=hashlib.sha256(before).digest()==hashlib.sha256(after).dig
↪   est()
05  print("Same bytes according to hash:",same)
```

WHAT TO TRACK

A changed file has a changed identity

PREDICT THE RESULT

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

A CHANGED FILE HAS A CHANGED IDENTITY · RESULT

CODE WALKTHROUGH W06-L08-S08

```
01  import hashlib
02  before=b"A,100\n"
03  after=b"A,101\n"
04  same=hashlib.sha256(before).digest()==hashlib.sha256(after).dig
↪   est()
05  print("Same bytes according to hash:",same)
```

WHAT TO TRACK

A changed file has a changed identity

VALIDATE THE RESULT

```
Same bytes according to
hash: False
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Write a README statement explaining positive-outflow CapEx and quarterly average-asset ROA.

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

Write a synthetic CSV, calculate its SHA-256, and store filename, source and hash in JSON.

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 a matching SHA-256 prove the financial values are correct? No. It checks byte identity, not business truth.

RECAP AND NEXT STEP

Recap and next step

The README carries run instructions and analytical conventions. Provenance records synthetic origin and hashes of actual input bytes.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 6 · LESSON 9

Configuration, secrets and reusable functions

Configuration controls reproducible choices; secrets grant access and must stay outside published files.

THE MENTAL MODEL

MENTAL MODEL

The mental model

Configuration controls reproducible choices; secrets grant access and must stay outside published files.

LOAD AND VALIDATE A PARAMETER

CODE WALKTHROUGH W06-L09-S03

```
01 import json,math
02 config=json.loads('{"cost_factor":1.1,"period":"2025-Q1"}')
03 factor=config["cost_factor"]
04 if isinstance(factor,bool) or not isinstance(factor,(int,float)
↪ ):
05     raise ValueError("cost_factor must be numeric")
06 if not math.isfinite(factor) or factor<0:
07     raise ValueError("cost_factor must be finite and nonnegativ
↪ e")
08 print("Period:",config["period"],"Cost factor:",factor)
```

WHAT TO TRACK

Load and validate a parameter

PREDICT THE RESULT

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

LOAD AND VALIDATE A PARAMETER · RESULT

CODE WALKTHROUGH W06-L09-S04

```
01 import json,math
02 config=json.loads('{"cost_factor":1.1,"period":"2025-Q1"}')
03 factor=config["cost_factor"]
04 if isinstance(factor,bool) or not isinstance(factor,(int,float)
↪ ):
05     raise ValueError("cost_factor must be numeric")
06 if not math.isfinite(factor) or factor<0:
07     raise ValueError("cost_factor must be finite and nonnegativ
↪ e")
08 print("Period:",config["period"],"Cost factor:",factor)
```

WHAT TO TRACK

Load and validate a parameter

VALIDATE THE RESULT

```
Period: 2025-Q1 Cost factor: 1.1
```

APPLY THE SUPPLIED REUSABLE PIPELINE

CODE WALKTHROUGH W06-L09-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":[5,10]})
config={"cost_factor":1.1}
r=compute_metrics(d,cost_factor=config["cost_factor"])
print(summarise(r)["profit"].to_dict())
```

WHAT TO TRACK

Dependency: supplied learner/src/pipeline_tools.py

PREDICT THE RESULT

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

APPLY THE SUPPLIED REUSABLE PIPELINE · RESULT

CODE WALKTHROUGH W06-L09-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]})
config={"cost_factor":1.1}
r=compute_metrics(d,cost_factor=config["cost_factor"])
print(summarise(r)["profit"].to_dict())
```

WHAT TO TRACK

Dependency: supplied learner/src/pipeline_tools.py

VALIDATE THE RESULT

```
{'A': 55.0}
```

AN ABSENT OPTIONAL CREDENTIAL

CODE WALKTHROUGH W06-L09-S07

```
01  import os
02  token=os.environ.get("AISYNAPSE_OPTIONAL_API_KEY")
03  if token:
04      print("Optional credential present; value is not displayed"
↪   )
05  else:
06      print("Offline mode: no credential required")
```

WHAT TO TRACK

An absent optional credential

PREDICT THE RESULT

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

AN ABSENT OPTIONAL CREDENTIAL · RESULT

CODE WALKTHROUGH W06-L09-S08

```
01  import os
02  token=os.environ.get("AISYNAPSE_OPTIONAL_API_KEY")
03  if token:
04      print("Optional credential present; value is not displayed"
↪   )
05  else:
06      print("Offline mode: no credential required")
```

WHAT TO TRACK

An absent optional credential

VALIDATE THE RESULT

```
Offline mode: no credential required
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Validate that a supplied cost factor of -1 is invalid and print a clear message without running the calculation.

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

Create a config JSON with cost_factor 0.9 and a separate .env.example containing only a placeholder variable name.

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 connecting a service justify copying its real key into an exercise file? No. Required exercises remain offline and use no credentials.

RECAP AND NEXT STEP

Recap and next step

Validate configuration before using it, reuse the same functions, and keep optional credentials outside notebooks and public archives.

UNDERSTAND APPLY VALIDATE EXPLAIN

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()))
```

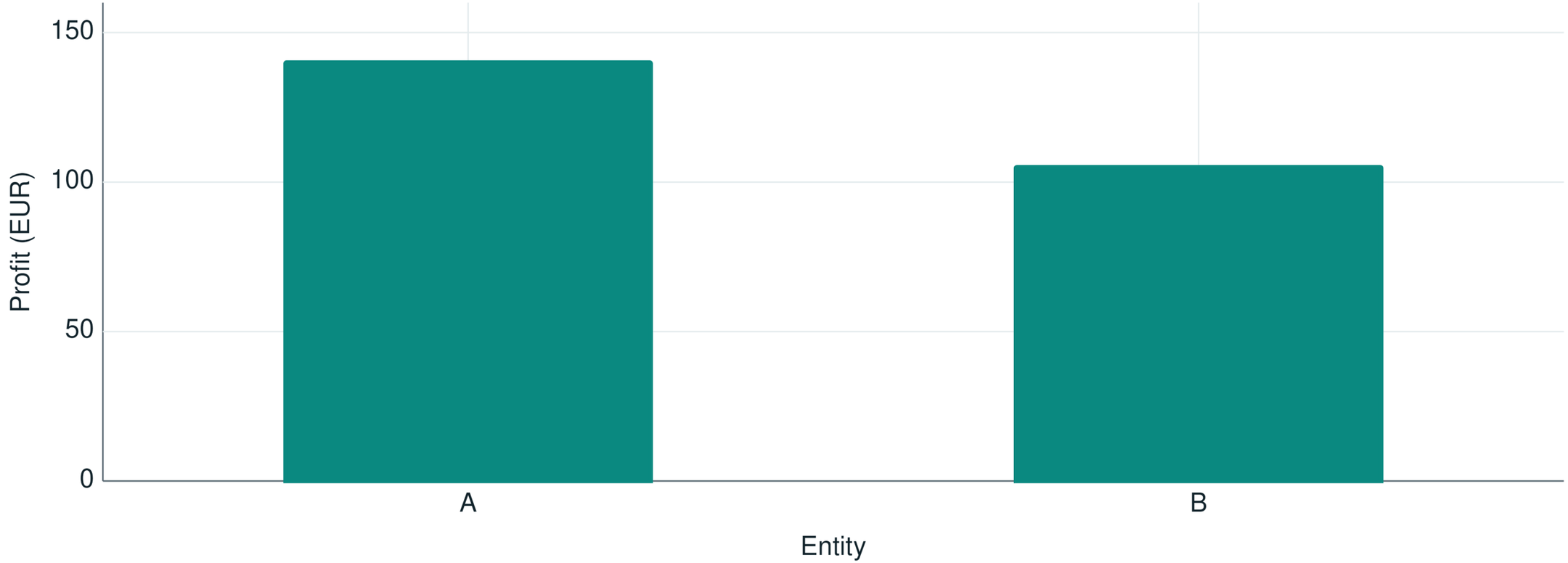
WHAT TO TRACK

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

FOUNDATION LESSON

WEEK 6 · LESSON 11

Findings, limitations and clean-run verification

A finding links a claim to evidence; a clean run rebuilds that evidence from the declared inputs.

THE MENTAL MODEL

MENTAL MODEL

The mental model

A finding links a claim to evidence; a clean run rebuilds that evidence from the declared inputs.

A SUPPORTED FINDING

CODE WALKTHROUGH W06-L11-S03

```
01  revenue=220;profit=70;fcf=45
02  print(f"2025 Jan–Feb, A: revenue EUR {revenue}; profit EUR {pro
↪   fit}.")
03  print(f"Margin {profit/revenue:.1%}; free cash flow EUR {fcf}."
↪   )
04  print("Synthetic two-month example; no causal or annual claim."
↪   )
```

WHAT TO TRACK

A supported finding

PREDICT THE RESULT

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

A SUPPORTED FINDING · RESULT

CODE WALKTHROUGH W06-L11-S04

```
01  revenue=220;profit=70;fcf=45
02  print(f"2025 Jan–Feb, A: revenue EUR {revenue}; profit EUR {pro
↪   fit}.")
03  print(f"Margin {profit/revenue:.1%}; free cash flow EUR {fcf}."
↪   )
04  print("Synthetic two-month example; no causal or annual claim."
↪   )
```

WHAT TO TRACK

A supported finding

VALIDATE THE RESULT

```
2025 Jan–Feb, A: revenue EUR
220; profit EUR 70.
Margin 31.8%; free cash flow
EUR 45.
Synthetic two-month example;
no causal or annual claim.
```

RECORD THE ANALYTICAL ENVIRONMENT

CODE WALKTHROUGH W06-L11-S05

```
01  import sys,pandas,numpy,matplotlib
02  print("Python:",sys.version.split()[0])
03  print("pandas:",pandas.__version__)
04  print("NumPy:",numpy.__version__)
05  print("Matplotlib:",matplotlib.__version__)
```

WHAT TO TRACK

Record the analytical environment

PREDICT THE RESULT

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

RECORD THE ANALYTICAL ENVIRONMENT · RESULT

CODE WALKTHROUGH W06-L11-S06

```
01  import sys,pandas,numpy,matplotlib
02  print("Python:",sys.version.split()[0])
03  print("pandas:",pandas.__version__)
04  print("NumPy:",numpy.__version__)
05  print("Matplotlib:",matplotlib.__version__)
```

WHAT TO TRACK

Record the analytical environment

VALIDATE THE RESULT

```
Python: 3.12.14
pandas: 2.2.3
NumPy: 2.3.5
Matplotlib: 3.10.8
```

HIDDEN NOTEBOOK STATE

CODE WALKTHROUGH W06-L11-S07

```
01  print(previous_result)
```

WHAT TO TRACK

Hidden notebook state

PREDICT THE RESULT

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

HIDDEN NOTEBOOK STATE · RESULT

CODE WALKTHROUGH W06-L11-S08

```
01  print(previous_result)
```

WHAT TO TRACK

Hidden notebook state

VALIDATE THE RESULT

```
NameError: name
'previous_result' is not
defined
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Write one finding for revenue 500, profit 75 and a three-month synthetic period. Include the margin and a limitation.

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

Write a standalone script that defines revenue and cost before calculating profit, then run it in a fresh process.

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 a notebook that works only after executing cells out of order satisfy reproducibility? No. It must run top to bottom from a fresh state.

RECAP AND NEXT STEP

Recap and next step

Tie every finding to verified measures, state limitations, and rerun from a clean environment with declared inputs and versions.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 6 · LESSON 12

Final capstone delivery and next steps

A professional handoff contains evidence, reproducible instructions and honest limits.

THE MENTAL MODEL

MENTAL MODEL

The mental model

A professional handoff contains evidence, reproducible instructions and honest limits.

A CONCRETE HANDOFF CHECKLIST

CODE WALKTHROUGH W06-L12-S03

```
required=["README.md","config/config.json","src/pipeline_tools.py",
 "data/raw/flows.csv","outputs/summary.csv",
 "outputs/findings.md","w06_data_provenance.json"]
for item in required:
    print(item)
print("One chosen variant; one compulsory capstone")
```

WHAT TO TRACK

A concrete handoff checklist

PREDICT THE RESULT

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

A CONCRETE HANDOFF CHECKLIST · RESULT

CODE WALKTHROUGH W06-L12-S04

```
required=["README.md","config/config.json","src/pipeline_tools.
py",
 "data/raw/flows.csv","outputs/summary.csv",
 "outputs/findings.md","w06_data_provenance.json"]
for item in required:
    print(item)
print("One chosen variant; one compulsory capstone")
```

WHAT TO TRACK

A concrete handoff checklist

VALIDATE THE RESULT

```
README.md
config/config.json
src/pipeline_tools.py
data/raw/flows.csv
outputs/summary.csv
outputs/findings.md
w06_data_provenance.json
One chosen variant; one
compulsory capstone
```

A PUBLIC-SAFE ARCHIVE

CODE WALKTHROUGH W06-L12-S05

```
01 from pathlib import Path
02 import zipfile
03 root=Path("submission");root.mkdir()
04 (root/"README.md").write_text("Synthetic offline capstone",enco
↪ ding="utf-8")
05 allowed=[root/"README.md"]
06 with zipfile.ZipFile("submission.zip","w") as z:
07     for p in allowed:
08         z.write(p,p.relative_to(root))
09 with zipfile.ZipFile("submission.zip") as z:
10     print(z.namelist())
```

WHAT TO TRACK

A public-safe archive

PREDICT THE RESULT

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

A PUBLIC-SAFE ARCHIVE · RESULT

CODE WALKTHROUGH W06-L12-S06

```
from pathlib import Path
import zipfile
root=Path("submission");root.mkdir()
(root/"README.md").write_text("Synthetic offline capstone",enco
ding="utf-8")
allowed=[root/"README.md"]
with zipfile.ZipFile("submission.zip","w") as z:
    for p in allowed:
        z.write(p,p.relative_to(root))
with zipfile.ZipFile("submission.zip") as z:
    print(z.namelist())
```

WHAT TO TRACK

A public-safe archive

VALIDATE THE RESULT

```
['README.md']
```

AN OMITTED REQUIRED OUTPUT

CODE WALKTHROUGH W06-L12-S07

```
from pathlib import Path
required=[Path("outputs/summary.csv"),Path("outputs/findings.md
")]
missing=[p.as_posix() for p in required if not p.is_file()]
if missing:
    raise ValueError("Missing deliverables: "+", ".join(missing
))
```

WHAT TO TRACK

An omitted required output

PREDICT THE RESULT

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

AN OMITTED REQUIRED OUTPUT · RESULT

CODE WALKTHROUGH W06-L12-S08

```
01  from pathlib import Path
02  required=[Path("outputs/summary.csv"),Path("outputs/findings.md
↪   ")]
03  missing=[p.as_posix() for p in required if not p.is_file()]
04  if missing:
05      raise ValueError("Missing deliverables: "+", ".join(missing
↪   ))
```

WHAT TO TRACK

An omitted required output

VALIDATE THE RESULT

```
ValueError: Missing
deliverables: outputs/summar
y.csv, outputs/findings.md
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Create a final checklist that explicitly includes a labelled chart, README, configuration, raw inputs, source and findings.

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

Write a two-sentence handoff note identifying synthetic data and a deterministic cost scenario without making an investment recommendation.

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: Is the Week 6 formative pipeline a second compulsory project? No. It prepares the reusable functions for the single required final capstone.

RECAP AND NEXT STEP

Recap and next step

Submit one complete offline capstone with reproducible evidence and explicit limits.
Keep practising Python, validating data and explaining results.

UNDERSTAND APPLY VALIDATE EXPLAIN