FOUNDATION LESSON

WEEK 4 · LESSON 5

Calculate with arrays and broadcasting

Arithmetic is elementwise; shapes must be compatible

THE IDEA BEHIND THE OPERATION

MENTAL MODEL

The idea behind the operation

Arithmetic is elementwise; shapes must be compatible

REPETITION AND ELEMENTWISE ARITHMETIC

CODE WALKTHROUGH W04-L05-S03

```
01  import numpy as np
02  values = [10, 20, 30]
03  array = np.array(values)
04  print("List:", values * 2)
05  print("Array:", array * 2)
06  print("Add:", array + 5)
```

WHAT TO TRACK

The same operator has type-specific behaviour.

PREDICT THE RESULT

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

REPETITION AND ELEMENTWISE ARITHMETIC — RESULT

CODE WALKTHROUGH W04-L05-S04

```
01  import numpy as np
02  values = [10, 20, 30]
03  array = np.array(values)
04  print("List:", values * 2)
05  print("Array:", array * 2)
06  print("Add:", array + 5)
```

WHAT TO TRACK

The same operator has type-specific behaviour.

VALIDATE THE RESULT

```
List: [10, 20, 30, 10, 20,
30]
Array: [20 40 60]
Add: [15 25 35]
```

PRODUCT PRICES ACROSS TWO STORES

CODE WALKTHROUGH W04-L05-S05

```
01  import numpy as np
02  quantity = np.array([[2, 3, 1], [4, 0, 2]])
03  price = np.array([10.0, 20.0, 5.0])
04  revenue = quantity * price
05  print("Shapes:", quantity.shape, price.shape)
06  print("Revenue:")
07  print(revenue)
08  print("Result shape:", revenue.shape)
```

WHAT TO TRACK

Rows: stores. Final-axis product order agrees.

PREDICT THE RESULT

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

PRODUCT PRICES ACROSS TWO STORES — RESULT

CODE WALKTHROUGH W04-L05-S06

```
01  import numpy as np
02  quantity = np.array([[2, 3, 1], [4, 0, 2]])
03  price = np.array([10.0, 20.0, 5.0])
04  revenue = quantity * price
05  print("Shapes:", quantity.shape, price.shape)
06  print("Revenue:")
07  print(revenue)
08  print("Result shape:", revenue.shape)
```

WHAT TO TRACK

Rows: stores. Final-axis product order agrees.

VALIDATE THE RESULT

```
Shapes: (2, 3) (3,)
Revenue:
[[20. 60.  5.]
 [40.  0. 10.]]
Result shape: (2, 3)
```

REJECT ZERO BEFORE DIVISION

CODE WALKTHROUGH W04-L05-S07

```
01  import numpy as np
02  revenue = np.array([100.0, 50.0])
03  units = np.array([5, 0])
04  if np.any(units == 0):
05      raise ValueError("unit price requires positive quantities")
06  print(revenue / units)
```

WHAT TO TRACK

This task requires strictly positive quantities.

PREDICT THE RESULT

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

REJECT ZERO BEFORE DIVISION — RESULT

CODE WALKTHROUGH W04-L05-S08

```
01  import numpy as np
02  revenue = np.array([100.0, 50.0])
03  units = np.array([5, 0])
04  if np.any(units == 0):
05      raise ValueError("unit price requires positive quantities")
06  print(revenue / units)
```

WHAT TO TRACK

This task requires strictly positive quantities.

VALIDATE THE RESULT

```
ValueError: unit price
requires positive quantities
```

GUIDED MODIFICATION

LEARNER PRACTICE

Guided modification

Compute revenues for quantities [2, 0, 5] and prices [12.0, 8.0, 3.0]. Apply a fictional 10 percent discount to each revenue. Print both arrays; zero quantity remains valid.

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 APPLICATION

LEARNER PRACTICE

Independent application

Try multiplying a two-by-three table by a two-entry vector. Catch ValueError and explain why the rightmost dimensions conflict. Then repair the business task using three product prices.

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

Array arithmetic works element by element. Predict broadcasting from the rightmost dimensions and verify business alignment separately. Guard invalid denominators before division. Next, we reduce arrays into totals and summaries along a stated axis.

UNDERSTAND APPLY VALIDATE EXPLAIN