FOUNDATION LESSON

WEEK 3 · LESSON 3

Return useful results and understand local names

Capture results; distinguish local names from shared objects.

EXPLAIN THE RESULT CHANNEL

MENTAL MODEL

Explain the result channel

Print displays information.
Return supplies a reusable result.
No explicit return supplies None.

MINIMAL COMPARISON AND PREDICTION

CODE WALKTHROUGH W03-L03-S03

```
01  def display_profit(revenue, cost):
02      print(revenue - cost)
03
04
05  def calculate_profit(revenue, cost):
06      return revenue - cost
07
08
09  shown = display_profit(1200, 800)
10  saved = calculate_profit(1200, 800)
11
12  print("Display result:", shown)
13  print("Saved result:", saved)
14  print("After reserve:", saved - 100)
```

WHAT TO TRACK

Capture each call’s result.
Distinguish display from return.

PREDICT THE RESULT

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

RESULT AND REASONING

CODE WALKTHROUGH W03-L03-S04

```
01  def display_profit(revenue, cost):
02      print(revenue - cost)
03
04
05  def calculate_profit(revenue, cost):
06      return revenue - cost
07
08
09  shown = display_profit(1200, 800)
10  saved = calculate_profit(1200, 800)
11
12  print("Display result:", shown)
13  print("Saved result:", saved)
14  print("After reserve:", saved - 100)
```

WHAT TO TRACK

The displayed 400 is not assigned to shown.
The returned 400 is assigned to

VALIDATE THE RESULT

```
400
Display result: None
Saved result: 400
After reserve: 300
```

CHAPTER BRIDGE: LOCAL WORKING NAMES

MENTAL MODEL

Chapter bridge: local working names

Local names belong to a call.
Names and objects are different.
Arguments can refer to shared objects.

BUSINESS APPLICATION: A RETURNED ACCUMULATOR

CODE WALKTHROUGH W03-L03-S06

```
01  def total_positive(amounts):
02      total = 0
03
04      for amount in amounts:
05          if amount > 0:
06              total += amount
07
08      return total
09
10
11  total = 999
12  accepted_total = total_positive([120, -20, 2500, 0])
13
14  print("Accepted total:", accepted_total)
15  print("Caller total:", total)
16  print("Empty batch:", total_positive([]))
```

WHAT TO TRACK

Include positive amounts only.
Empty batch total: 0.
Read without mutating.

PREDICT THE RESULT

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

LOCAL-SCOPE RESULT AND LIFETIME

CODE WALKTHROUGH W03-L03-S07

```
01  def total_positive(amounts):
02      total = 0
03
04      for amount in amounts:
05          if amount > 0:
06              total += amount
07
08      return total
09
10
11  total = 999
12  accepted_total = total_positive([120, -20, 2500, 0])
13
14  print("Accepted total:", accepted_total)
15  print("Caller total:", total)
16  print("Empty batch:", total_positive([]))
```

WHAT TO TRACK

The local total and caller total differ.
Each call starts its own

VALIDATE THE RESULT

```
Accepted total: 2620
Caller total: 999
Empty batch: 0
```

MUTATION BOUNDARY AND PREDICTION

CODE WALKTHROUGH W03-L03-S08

```
01  def adjust_draft(amount, amounts):
02      amount = 0
03      amounts.append(0)
04      print("Inside:", amount, amounts)
05
06
07  amount = 100
08  amounts = [10, 20]
09
10  adjust_draft(amount, amounts)
11
12  print("Outside:", amount, amounts)
```

WHAT TO TRACK

Rebinding a local name differs from mutation.

PREDICT THE RESULT

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

EXPLAIN THE DIFFERENT EFFECTS

CODE WALKTHROUGH W03-L03-S09

```
01  def adjust_draft(amount, amounts):
02      amount = 0
03      amounts.append(0)
04      print("Inside:", amount, amounts)
05
06
07  amount = 100
08  amounts = [10, 20]
09
10  adjust_draft(amount, amounts)
11
12  print("Outside:", amount, amounts)
```

WHAT TO TRACK

The caller’s integer binding stays 100.
The shared list gains a zero.

VALIDATE THE RESULT

```
Inside: 0 [10, 20, 0]
Outside: 100 [10, 20, 0]
```

MAKE A SEPARATE FLAT-LIST DRAFT

CODE WALKTHROUGH W03-L03-S10

```
01  def draft_with_marker(amounts):
02      draft = amounts.copy()
03      draft.append(0)
04      return draft
05
06
07  original = [10, 20]
08  draft = draft_with_marker(original)
09
10  print("Original:", original)
11  print("Draft:", draft)
```

WHAT TO TRACK

Copy the flat outer list before append.
Return the draft to the caller.

VALIDATE THE RESULT

```
Original: [10, 20]
Draft: [10, 20, 0]
```

GUIDED DEBUGGING

CODE WALKTHROUGH W03-L03-S11

```
01  def classify_amount(amount):
02      if amount <= 0:
03          return "invalid"
04
05      if amount >= 2000:
06          return "review"
07
08
09  print(classify_amount(2500))
10  print(classify_amount(100))
```

WHAT TO TRACK

What happens when a path reaches the end?
Add the missing ordinary-case

VALIDATE THE RESULT

```
review
None
```

RECAP AND ENDING

Recap and next step

Print displays; return supplies a result.
Local rebinding leaves caller bindings alone.
Shared mutations can remain visible.

UNDERSTAND APPLY VALIDATE EXPLAIN