Session 3.2: Advanced Loops & While

Nested Loops & Condition-Based Iteration • Week 3, Day 2

Today's Journey

- Part 1: Nested Loops - Loops Inside Loops!
- Part 2: Workshop - Process 2D Data
- Break!
- Part 3: While Loops & Loop Control
- Part 4: Workshop - Advanced Patterns
- Part 5: Quiz & Week 3 Wrap-up

Part 1: Nested Loops

Loops inside loops!

Last Session Review 🔄

Session 3.1 you learned:

- ✓ For loops (automation!)
- ✓ range() function
- ✓ enumerate() for position
- ✓ Accumulator pattern
- ✓ Git branching

Today: Take loops to next level!

Nested loops = loop inside another loop

- Process 2D data (tables, matrices)
- Generate combinations
- Complex patterns

This unlocks multidimensional data! 🚀

What Are Nested Loops? (Simple Explanation)

Nested loop = loop inside another loop

Real life examples:

- For each student (outer loop)
- Check each homework problem (inner loop)
- For each row in spreadsheet (outer)
- Process each column (inner)
- For each email recipient (outer)
- Send each attachment (inner)

Outer loop runs once

Inner loop runs FULLY each time

Think: rows and columns!

Nested Loop Syntax

Format:

```
for OUTER_ITEM in OUTER_COLLECTION:
    for INNER_ITEM in INNER_COLLECTION:
        # code here runs for each combination
```

Key points:

- Two levels of indentation
- Inner loop completes fully each time
- Total iterations = outer × inner

Example: 3 outer × 4 inner = 12 total iterations

Mind-bending at first, but powerful! 💡

Your First Nested Loop (1/3)

```
# Multiplication table
for i in range(1, 4):  # Outer: 1, 2, 3
    for j in range(1, 4):  # Inner: 1, 2, 3
        print(f'{i} × {j} = {i*j}')


# Output:
# 1 × 1 = 1
# 1 × 2 = 2
# 1 × 3 = 3
# 2 × 1 = 2
# 2 × 2 = 4
# 2 × 3 = 6
# 3 × 1 = 3
# 3 × 2 = 6
# 3 × 3 = 9
# Inner loop runs 3 times for EACH outer
```

Your First Nested Loop (2/3)

```
# Multiplication table
for row in range(1, 6):                 # outer loop: rows 1 to 5
    for col in range(1, 6):             # inner loop: columns 1 to 5
        product = row * col
        print(f"{product:3d}", end=" ")    # print with spacing
    print()   # new line after each row

# Output:
1   2   3   4   5
2   4   6   8  10
3   6   9  12  15
4   8  12  16  20
5  10  15  20  25

# Output (a nice 5×5 table):
```

Nested Loops: Creating Patterns (3/3

```
# Right triangle pattern
for i in range(1, 6):           # Row number
    for j in range(i):          # Stars in this row
        print('*', end='')      # end='' prevents newline
    print()                     # New line after each row

# Output:
# *         ← row 1: 1 star
# **        ← row 2: 2 stars
# ***       ← row 3: 3 stars
# ****      ← row 4: 4 stars
# *****     ← row 5: 5 stars

# Visual patterns:
# Row 1: inner runs 1 time
# Row 2: inner runs 2 times
# Row 3: inner runs 3 times
```

Nested Loops for 2D Data

2D data = table/matrix (rows and columns)

Examples:

- Spreadsheet (rows × columns)
- Image pixels (width × height)
- Game board (grid)
- Student grades (students × subjects)

Pattern:

```
for row in data:
    for value in row:
        process(value)
```

Essential for data science! 📊

Processing 2D Lists (Matrices)

```
# Student grades: [student][subject]
grades = [
    [95, 87, 92],  # Alice: Math, English, Science
    [88, 90, 85],  # Bob
    [92, 88, 94]   # Carol
]

# Calculate average for each student
for student_num, student_grades in enumerate(grades, 1):
    total = 0
    for grade in student_grades:
        total += grade
    avg = total / len(student_grades)
    print(f'Student {student_num} average: {avg:.1f}')
# Output:
# Student 1 average: 91.3
# Student 2 average: 87.7
# Student 3 average: 91.3
```

Nested Loops with Lists

```
# Generate all combinations
colors = ['red', 'blue', 'green']
sizes = ['small', 'medium', 'large']
products = []
for color in colors:
    for size in sizes:
        product = f'{size} {color} shirt'
        products.append(product)
print(products)
# ['small red shirt', 'medium red shirt', 'large red shirt',
#  'small blue shirt', 'medium blue shirt', 'large blue shirt',
#  'small green shirt', 'medium green shirt', 'large green
shirt']

# 3 colors × 3 sizes = 9 products
# Automatic combinations!
```

When to Use Nested Loops

✅ USE nested loops for:

- Processing 2D data (tables, matrices)
- Generating combinations
- Creating visual patterns
- Comparing all pairs

⚠️ BE CAREFUL:

- Can be slow with large data
- 1000 × 1000 = 1,000,000 iterations!
- Sometimes comprehensions are better

But for learning: perfect! 🎯

Nested Loops in ML

Machine Learning uses nested loops:

Training epochs and batches:

```
for epoch in range(num_epochs):
    for batch in dataset:
        train_on_batch(batch)
```

Image processing (pixels):

```
for row in image:
    for pixel in row:
        process_pixel(pixel)
```

Grid search (hyperparameter tuning):

```
for learning_rate in rates:
    for batch_size in sizes:
        train_model(lr, bs)
```

You'll see this constantly!

Part 2: Workshop

Process 2D Data!

💻 Exercise 1: Multiplication Table Generator

Create file: mult_table_generator.py

Build complete multiplication table:

1. Use nested loops: range(1, 11) for both

2. Print formatted table:

   1×1=1 1×2=2 1×3=3 ... 1×10=10

   2×1=2 2×2=4 2×3=6 ... 2×10=20

   ...

   10×1=10 10×2=20 ... 10×10=100

3. Align columns nicely

4. Use f-strings for formatting

Bonus: Add row/column headers

This creates a full times table!

💻 Exercise 2: Grade Matrix Processor

Create file: grade_matrix.py

Given grade matrix:

```
grades = [[95,87,92,88], [88,90,85,92],
          [92,88,94,90], [78,85,80,88]]
```

Calculate and print:

1. Average for each student (row)
2. Average for each subject (column)
3. Overall class average
4. Highest grade in entire matrix
5. How many A's total (>= 90)

Use nested loops!

This is real data analysis! 📊

💻 Exercise 3: Pattern Master

Create file: pattern_master.py

Generate these patterns with nested loops:

Pattern 1: Square

```
*****
*****
*****
*****
*****
```

Pattern 2: Countdown

```
5 4 3 2 1
5 4 3 2
5 4 3
5 4
5
```

Pattern 3: Number grid (1-25 in 5×5)

Challenge your pattern-building skills!

Break

Nested loops mastered!

Part 3: While Loops & Control

Condition-based repetition

Introducing While Loops

for loop = repeat for each item

while loop = repeat while condition is True

Difference:

- for: know how many times
- while: repeat until condition changes

Use while when:

- Don't know iterations in advance
- Waiting for user input
- Checking a condition
- Model convergence (ML!)

Different tool, same goal: repetition! 🔁

While Loop Syntax

Format:

```
while CONDITION:
    # code to repeat
    # must change condition eventually!
```

Parts:

- while = keyword
- CONDITION = expression (True/False)
- : = colon
- Indented code = what repeats

⚠️ WARNING: Must make condition False eventually!

Otherwise: infinite loop (program stuck forever!)

Your First While Loop

```
# Count to 5 with while
count = 0
while count < 5:
    print(count)
    count += 1  # MUST update condition!

# Output:
# 0
# 1
# 2
# 3
# 4

# If you forget 'count += 1':
# Infinite loop! (Ctrl+C to stop)
# Always update the condition variable!
```

While vs For - Same Result

```
# Count to 5 with for loop
for i in range(5):
    print(i)

# Same with while loop
i = 0
while i < 5:
    print(i)
    i += 1

# Both output: 0, 1, 2, 3, 4
# For loop: cleaner when you know count
# While loop: better for conditions

# Choose based on situation!
```

When to Use While Loops

✅ USE while loops when:

- Waiting for user input
- Checking convergence (ML)
- Processing until condition met
- Reading file until end
- Game loops (run until quit)

✅ USE for loops when:

- Iterating over collection
- Know number of iterations
- Processing each item

Both are powerful tools! 🔧

While Loop: User Input Example

```
# Keep asking until valid input
age = -1  # Invalid start value
while age < 0 or age > 120:
    age = int(input('Enter your age (0-120): '))
    if age < 0 or age > 120:
        print('Invalid! Try again.')
print(f'Valid age: {age}')
# Loop continues until valid age entered
# Don't know how many tries it takes!
# Perfect use case for while!

# This is input validation!
```

Infinite Loops - Dangerous! ⚠️

Infinite loop = never stops running

Bad example:

- count = 0
- while count < 10:

  print(count)

  # Forgot to update count!

This runs FOREVER! Computer freezes!

How to stop:

- Press Ctrl+C (kills program)
- Close terminal window

ALWAYS make condition eventually False!

Update your condition variable!

break Statement - Exit Loop Early

break = immediately exit loop

Use when:

- Found what you're looking for
- Error condition met
- User wants to quit

Works in both for and while loops

Example:

```
while True:
    answer = input('Continue? (y/n): ')
    if answer == 'n':
        break  # Exit loop!
```

Powerful control tool! 🛑

Using break in Loops

```
# Find first number divisible by 7 and 11
for num in range(1, 1000):
    if num % 7 == 0 and num % 11 == 0:
        print(f'Found it: {num}')
        break  # Stop searching!
# Output: Found it: 77
# Doesn't check 78-999 (break stopped loop)

# Without break, would check all 999 numbers
# With break: stops when found
# More efficient!

# Use break to exit early when done!
```

continue Statement - Skip Rest of Iteration

continue = skip to next iteration

Difference from break:

- break: exit loop completely
- continue: skip current, continue loop

Use when:

- Skip invalid data
- Ignore certain values
- Filter while looping

Example:

```
for num in range(10):
    if num % 2 == 0:
        continue  # Skip evens
    print(num)  # Only odds printed
```

Using continue in Loops

```
# Process only positive numbers
numbers = [5, -3, 8, -1, 12, -7, 15, 0, 20]
for num in numbers:
    if num <= 0:
        continue  # Skip non-positive

    # This code only runs for positive numbers
    squared = num ** 2
    print(f'{num} squared = {squared}')

# Output:
# 5 squared = 25
# 8 squared = 64
# 12 squared = 144
# 15 squared = 225
# 20 squared = 400
# Negative numbers skipped!
```

While Loops in ML

Machine Learning uses while loops:

Training until convergence:

```
error = float('inf')
while error > threshold:
    train_step()
    error = calculate_error()
```

Early stopping:

```
patience = 0
while patience < max_patience:
    if no_improvement():
        patience += 1
```

You'll write loops like this in ML! 🎓

Part 4: Workshop

Master While Loops!

💻 Exercise 4: Number Guesser Game

Create file: number_guesser.py

Build a guessing game:

1. Computer picks random number 1-100

```
import random

secret = random.randint(1, 100)
```

2. User guesses (while loop)

3. Give hints: 'Too high' or 'Too low'

4. Count number of guesses

5. Continue until correct

6. Print: 'Correct! You took N guesses'

Use while loop (don't know how many guesses!)

Use break when correct

This is a real game! 🎮

Part 5: Quiz & Wrap-up

Week 3 Complete!

Session 3.2 Quiz

🎉 Week 3 COMPLETE! You're Automating!

This week you mastered:

- ✓ For loops (process collections)
- ✓ range() and enumerate()
- ✓ Nested loops (2D data!)
- ✓ While loops (condition-based)
- ✓ break and continue (control flow)
- ✓ Git branching (professional workflow)

You can now:

- Automate any repetitive task
- Process tables and matrices
- Build complex patterns
- Control program flow
- MAJOR progress! 🌟

Homework - Week 3 Mastery

1. Complete the quiz** (10 questions)

2. Project: Data Table Processor

- Build: table_processor.py
  - Create 5×5 matrix (nested lists)
  - Fill with random numbers 1-100
  - Calculate row averages
  - Calculate column averages
  - Find max value and its position
  - Use nested loops!

3. Git Practice

- Work on feature branch
- Commit incrementally
- Merge to main, push to GitHub

4. Prepare: Read about if/else (Week 4 topic)

Looking Ahead: Week 4 - Decisions!

Next week: Conditional Statements

Session 4.1: If/Elif/Else

- Making decisions in code
- Boolean logic
- Comparison operators
- Complex conditions

Session 4.2: Conditionals + Loops

- Filtering data
- Data validation
- Smart processing

Programs that think! 🧠

Week 3 Reflection 💭

Amazing growth this week:

Tuesday:

- 'How do I repeat code?'

Today:

- 'I write nested loops and while loops!'

You've learned:

- Automation with loops
- Processing 2D data
- Flow control (break, continue)
- Professional Git workflow

Next week: Decision-making in code

You're halfway to ML-ready! 🎓

Weekend practice, see you Monday! 🚀