مكتب

Lab Activity

Week 9: Introduction to `for` Loops

Duration: 50 minutes — Work in pairs!

Name: _____ **Partner:** _____

Lab Goals

By the end of this lab, you will be able to:

- Write simple for loops using `range()`
- Repeat actions a specific number of times
- Use loop variables in your programs
- Create simple patterns with loops

Getting Started

1. Extract the given `week9_lab.zip` file to your desktop. This folder contains all the files you need for this lab.

1 Exercise 1: Your First Loop! (6 minutes)

Let's start by comparing how we used to repeat things vs. using a loop. Type this code exactly as shown and run it.

Exercise 1: From repetition to loops

Open `week9_ex1.py` and type the following code:

```
for i in range(3):
    print(f"Line {i}: Hi!")
```

Run it and see what happens.

Understanding What Just Happened:
When you ran the code, Python did this behind the scenes:

```
for i in range(3):  <-- Creates list of numbers [0, 1, 2]
    First time:  i = 0, prints "Line 0: Hi!"
    Second time: i = 1, prints "Line 1: Hi!"
    Third time:  i = 2, prints "Line 2: Hi!"
    Fourth time: i would be 3, but 3 is not in [0,1,2], so STOP
```

Answer the following questions:

- **Before running:** If you change `range(3)` to `range(5)`, what will be the first line number printed? _____

What will be the last line number? _____

- Now run it. Were you correct (yes/no)? _____
- Change `range(5)` to `range(0)` - What happens? _____
- Try `range(1)` - How many times does it print? _____
  What line number? _____
- **Pattern Discovery:** If you use `range(n)` *(where `n` can be any +ve integer)*, the loop runs _____ times, and `i` goes from _____ to _____
- Change the message to print your name 7 times. What should you put in `range()`? _____
- **Challenge:** Make it print "Line 1:", "Line 2:", "Line 3:" instead of starting at 0. *Hint: You'll need to modify what's inside the print statement*

Tip: The number in `range()` tells Python how many times to repeat!

Checkpoint: Show your teacher the output before moving on!

2 Exercise 2: Understanding range() (7 minutes)

For this exercise you are not allowed to run the code on a computer. First think about it and then fill in the blanks to make these loops work correctly.

Make it count from 0 to 4:

```
print("Count to 4:")
for number in range(_____):
    print(number)
print()
```

Make it print "Clap!" exactly 7 times

```
print("Seven claps:")
for _____ in range(_____):
    print("Clap!")
print()
```

Make it show 5 star emojis

```
print("Five stars:")
for _____ in _____(_____):
    print("*", end=" ")  # end=" " keeps them on same line
print()
```

Count down from 10 (tricky!)

```
print("\nCountdown:")
for i in range(10):
    print(_____)
print("Blast off!")
```

After you've filled your answers above, now you can test your code on a computer.

How many of the above code snippets dis you get right? _____

Checkpoint: All four loops working correctly? Excellent!

Pair Programming: Switch who types every 5 minutes!

3 Exercise 3: Recognize Repetition - When to Use Loops (8 minutes)

Open `week9_ex3.py` and Type the following code:

Step 1: Start with this simple pattern:

```
for i in range(8):
    print("#", end=" ")
print()
```

Recall: The `end=" "` part keeps the symbols on the same line.

Step 2: Try these pattern experiments (run after each change):

1. Change the # symbol to a *. How many stars appear?

   Result: _____

2. Make it print 12 symbols instead of 8. What did you change?

   Changed: _____

3. Can you make it alternate between two symbols?

```
for i in range(10):
    if i % 2 == 0:
        print("*", end=" ")
    else:
        print("o", end=" ")
```

   Before running, can you predict the pattern? _____

   Replace current code in the file `week9_ex3.py` with the one given above. Now run the code. Was your prediction correct? (yes/no) _____

   What does `i % 2` do? _____

Step 3: Create a Staircase Effect

```
for i in range(5):
    spaces = " " * i  # Create spacing
    print(spaces + "#")
```

First predict the output without running the program:

```
----------------------
----------------------
----------------------
----------------------
----------------------
```

Now run it. Did it match your prediction? (yes/no) _____

4 Exercise 4: Times Tables Helper (10 minutes)

Open week9_ex4.py. It already has a useful program for practicing multiplication:

```
1  # Multiplication table generator
2  print("=== Times Tables Practice ===")
3
4  # Ask which table to practice
5  number = int(input("Which times table? (2-12): "))
6
7  print(f"\n{number} Times Table:")
8  print("-" * 20)  # Line of dashes
9
10 # Generate the table
11 for i in range(1, 11):  # Why 11?
12     result = number * i
13     print(f"{number} x {i} = {result}")
14
15 # Add a practice question
16 print("\nQuick test:")
17 test_num = 5
18 answer = int(input(f"What is {number} x {test_num}? "))
19
20 if answer == number * test_num:
21     print("Correct!")
22 else:
23     print(f"Not quite. {number} x {test_num} = {number * test_num}")
```

Test your program:

- Run it with number = 7
- Check that it shows 7 x 1 through 7 x 10
- Why there are two numbers in `range(1, 11)` in line 11? _____

Modify the program:

- Make it show up to 12 times instead of 10
- Instead of multiplying the number by 5 *(line 18)*, ask the user for a number between 1 and 12
- Add an encouraging message after each correct answer

Tip: `range(1, 11)` starts at 1 and goes up to (but not including) 11!

5 Exercise 7: Debug Using Loop Tracing (5 minutes)

These programs have bugs. Use tracing to find and fix them!

Bug 1: Wrong Output

```
1 # Should print: 5, 4, 3, 2, 1
2 # But it prints: 0, 1, 2, 3, 4
3 for i in range(5):
4     print(i)
```

Trace to find the bug:

i	We want	Formula needed for that
0	5	5 - 0 = 5
1	4	5 - 1 = 4
2	_____	_____
3	_____	_____
4	_____	_____

Your fix:

```
for i in range(5):
    print(__________)  # Use the formula from your trace!
```

Bug 2: Off by One

```
# Should show numbers 1 through 10
# But stops at 9!
for num in range(10):
    print(num)
```

Fix: Change `range(10)` to `range(_____)`

Checkpoint: Both bugs fixed? Excellent debugging! Show it to your teacher!

6 Exercise 5: Mini-Challenge - Digital Clock Display (10 minutes)

Create a simple countdown timer:

1. Ask the user how many seconds to count down
2. Count down from that number to 0
3. Print each number on the screen
4. Make a celebration when it reaches 0!

Example Output:

```
=== Countdown Timer ===
How many seconds? 5

Starting countdown...
5...
4...
3...
2...
1...
TIME'S UP!
```

Starter Help:

Open week9_ex5.py and start with this code:

```
print("=== Countdown Timer ===")
seconds = int(input("How many seconds? "))

print("\nStarting countdown...")

# Your loop here to count down
for i in range(seconds):
    # Figure out what number to show
    # Hint: If seconds=5 and i=0, show 5
    # If seconds=5 and i=1, show 4
    current = _______
    print(f"{current}...")

# Add your celebration here!
```

Bonus challenges:

- Count down only even numbers from any starting point
- Make every 5th second special: "10... 9... 8... 7... 6... [5 - HALFWAY!] 4... 3..."
- Create a "warning zone" - numbers below 4 print differently: "7... 6... 5... 4... WARNING: 3! WARNING: 2! WARNING: 1!"

Checkpoint: Timer counting down correctly? Show your teacher!

7 Lab Summary

Checkpoint: Final checkpoint - make sure you have:

- ☐ Completed all exercises with your partner
- ☐ Traced loops using tables in Exercise 2
- ☐ Filled in code using range() correctly
- ☐ Identified when to use loops in Exercise 4
- ☐ Created visual patterns with loops
- ☐ Built a times table generator
- ☐ Fixed bugs using trace tables
- ☐ Shown all checkpoints to your teacher

Reflection (2 minutes)

Rate your understanding of today's concepts:

Concept	Need Help	Getting It	Got It!
for loops	○	○	○
Tracing loop execution	○	○	○
When to use loops	○	○	○
Debugging with traces	○	○	○

Compare your first loop to your last. What improved?

_____

Which tracing technique helped you most: tables, memory boxes, or step-by-step?

_____

Excellent work with loops!