مكتب

Lab Activity

Week 11: Smart Loops

Duration: 50 minutes — Work in pairs!

Name: _____ **Partner:** _____

Lab Goals

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

- Put if statements inside for loops to filter numbers
- Use break to stop loops early when you find what you need
- Combine loops and conditions to make smart programs
- Count specific items that match certain conditions

Getting Started

1. Extract the given `week10_lab.zip` file to your desktop. This folder contains all the files you need for this lab.

1 Exercise 1: Let's Filter Numbers (5 minutes)

Open `week11_ex1.py` and type the code below to print only odd numbers from 1 to 10.

```
# Exercise 1: Print only odd numbers
print("=== Odd Number Filter ===")
print("Numbers from 1 to 10:")
print("Odd numbers only: ", end="")

for num in range(1, 11):
    if num % 2 == 1:
        print(num, end=" ")

print()  # New line at the end
print("\nAll odd numbers printed!")
```

Now modify the code:

- Change the if condition to print only even numbers (hint: `% 2 == 0`)
- Change the range to go from 1 to 20
- Add a counter to count how many odd numbers you found

Checkpoint: Show your teacher the output before moving on!

2 Exercise 2: Multiple Counter (5 minutes)

Open week11_ex2.py and fill in the blanks to count multiples of 5 between 1 and 30.

```
# Exercise 2: Count multiples of 5
print("=== Counting Multiples of 5 ===")
count = 0

print("Checking numbers from 1 to 30:")

for num in range(1, 31):
    if num % 5 == 0:
        print(f"{num} is a multiple of 5!")
        _______ = _______ + 1

print(f"\nTotal multiples of 5 found: {_______}")
print(f"They are: ", end="")

# Print them again in one line
for num in range(1, 31):
    if _______ % _______ == _______:
        print(num, end=" ")
print()
```

Test your program:

- Should find 6 multiples of 5 between 1 and 30
- They should be: 5, 10, 15, 20, 25, 30

Checkpoint: Counter working correctly? Great job! Show your teacher!

Pair Programming: Switch who types every 5 minutes!

3 Exercise 3: Experiment with break (10 minutes)

Let's explore how break helps us stop loops early.
Step 1: Open week11_ex3.py which has the following code already written:

```
# Exercise 3: Find first square number over 50
print("=== Find First Large Square ===")
print("Looking for first square number greater than 50...")

found = False
found_number = 0
found_root = 0

for n in range(1, 20):
    square = n * n
    print(f"{n} squared = {square}")

    if square > 50:
        print(f"Found it! {square} is greater than 50")
        found = True
        found_number = square
        found_root = n
        break
```

```
    else:
        print("  Too small, keep looking...")

if found:
    print(f"\nThe answer is {found_root} squared = {found_number}")
else:
    print("\nNo square found in range")
```

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

1. What happens if you remove the break statement?
   Result: _____
2. What happens if you change the condition to `> 100`?
   Result: _____
3. Can you modify it to find the first square less than 50 (starting from 20 going down)?
   Hint: Use `range(20, 0, -1)`

Tip: The break statement saves time by stopping as soon as we find what we're looking for!

4 Exercise 4: Fix the Lucky Number Finder (10 minutes)

Open week11_ex4.py, find and fix all issues in the code:

```
print("=== Lucky Number Finder ===")
for num in range(1, 20)
    print(f"Checking {num}...")
    if num = 7:
        print("Found lucky number 7!")
    else:
        print("Not lucky, continue...")

print("Done searching!)
```

Checkpoint: Fixed code runs without errors and stops at 7? Show your teacher!

5 Exercise 5: Complete the Divisor Finder (5 minutes)

Open week11_ex5.py and complete the code to find all divisors of a number.

```
# Exercise 5: Find all divisors
number = int(input("Enter a number to find its divisors: "))
divisor_count = _______  # Initialize counter

print(f"\nFinding all divisors of {number}:")
print("Divisors: ", end="")

for possible_divisor in range(1, number + 1):
    if number % _______ == 0:
        print(possible_divisor, end=" ")
        _______ = _______ + _______

print(f"\n\n{number} has {_______} divisors")

# Check if it's prime (only 2 divisors: 1 and itself)
```

```python
if divisor_count == 2:
    print(f"{number} is a prime number!")
elif divisor_count == 1:
    print(f"{number} is neither prime nor composite!")
else:
    print(f"{number} is a composite number!")
```

Test with these numbers:

- 12 should have 6 divisors: 1, 2, 3, 4, 6, 12
- 7 should have 2 divisors: 1, 7 (it's prime!)
- 1 should have 1 divisor: 1

Checkpoint: Divisor finder working correctly? Nice work!

6 Exercise 6: Mini-Challenge - Number Guessing Helper (10 minutes)

Create a program that helps Player 2 guess Player 1's number efficiently:

1. Player 1 picks a secret number between 1 and 50
2. The program checks numbers systematically
3. For each guess, Player 1 types 'h' (too high), 'l' (too low), or 'c' (correct)
4. Use break when the correct number is found
5. Count how many guesses it took

Example Output:

```
=== Number Guessing Helper ===
Player 1: Think of a number between 1 and 50
Player 2: I'll help you guess it!

Press Enter when ready...

Is your number 25? Enter h/l/c: h
Is your number 12? Enter h/l/c: l
Is your number 18? Enter h/l/c: c

Found it! Your number is 18
It took 3 guesses!
```

Open `week11_ex6.py` and write your code in the file.

Pair Programming: Switch who types every 5 minutes!

Bonus: Can you think of a better guessing strategy than just trying every number?

Checkpoint: Guessing helper working with break? Excellent!

7 Lab Summary

Checkpoint: Final checkpoint - make sure you have:

- ☐ Completed all exercises with your partner
- ☐ Used if statements inside for loops successfully
- ☐ Used break to exit a loop early
- ☐ Counted items that match conditions
- ☐ Shown all checkpoints to your teacher

Reflection (2 minutes)

Rate your understanding of today's concepts:

Concept	Need Help	Getting It	Got It!
Filtering numbers with loops	○	○	○
Using break to exit early	○	○	○
Counting matching items	○	○	○
Combining loops & conditions	○	○	○

One smart loop I created today: _____

The break statement is useful when: _____

Great work making smart loops today!