

Programming Fundamentals

Lab 02

Booleans & Logic

Comp 102 — Forman Christian University
Spring 2026

Estimated Time: ~**3 hours** — Based on Lectures 3 & 4

How to Use This Lab

This lab is a **walkthrough tutorial**—it is designed to guide you through learning, not to test you. Work through each section at your own pace:

- **Predict** — Write down what you think will happen *before* touching the computer.
- **Type** — Enter the code in Thonny’s **Code Editor** (top pane) and press **Run** (or **F5**).
- **Verify** — Compare your prediction with the actual result. If they differ, figure out *why*.
- Exercises are graded by difficulty:
 - ★ **Easy** — Immediate practice. Follow the pattern you just saw.
 - ★★ **Medium** — Requires some thinking. Combines concepts.
 - ★★★ **Hard** — Stretch goal. It is OK to need help!
- **Ask for help** whenever you are stuck for more than 5 minutes.

1 Review: Strings & I/O (~15 min)

Ref: Lecture 3 — Strings and I/O slides

💡 Shell vs Code Editor

Use the **Shell** (>>>) for short, one-line experiments. When the code is more than 3–4 lines (especially code using `input()`), type it in Thonny's **code editor** (top pane) and press **Run** (or **F5**).

Exercise 1

★ Easy

Predict the output of each `print` statement, then verify:

```
1 >>> x = "Best"
2 >>> y = 3
3 >>> z = "goals"
4 >>> print(x, y, z)
```

Your prediction: _____

*Hint: When `print` receives multiple arguments separated by commas, it puts **spaces** between them.*

Exercise 2

★ Easy

Predict the output, then verify:

```
1 >>> x = "Best"
2 >>> y = 3
3 >>> z = "goals"
4 >>> print(x + str(y) + z)
```

Your prediction: _____

Why is `str(y)` needed here?

Exercise 3

★★ Medium

Predict first! Type this in the **code editor** and run it. When prompted, enter 7. What does each line print?

```
1 w = input("Enter: ")
2 print(w * 2)
3 print(int(w) * 2)
```

Line 2 prints: _____ Line 3 prints: _____

Why are they different?

Exercise 4

★★ Medium

Type this in the **code editor** and run it. Identify the error(s) and fix them:

```
1 x = 42
2 y = 2.5
3 z = "age"
```

```

4 msg1 = "x is " + x
5 msg2 = "y is " + y
6 msg3 = "z is " + z

```

Which line(s) cause an error and why?

Exercise 5

★★ Medium

Predict what each line prints, then verify:

```

1 dist = 3
2 print(f"Half distance: {dist/2} km")
3 print(f"Distance squared: {dist**2} km")

```

Line 2 prints: _____

Line 3 prints: _____

2 The Boolean Type (~15 min)

Ref: Lecture 4 — The Boolean Type slides

Light switch	on → True
	off → False
Door	open → True
	closed → False
Attendance	present → True
	absent → False

Everything is either True or False — that's a Boolean!

Exercise 6

★ Easy

Type each line in the Shell. Predict the output of each **before** you type it.

```

1 >>> True
2 >>> type(True)
3 >>> type(False)
4 >>> print(type(False))

```

Your predictions: _____

What do you notice about the capitalization of True and False?

⚠ Case Matters!

Python is **case-sensitive**. Only `True` and `False` (with a capital letter) are valid booleans. Writing `true`, `FALSE`, or `TRUE` will give you a `NameError`.

Exercise 7

★★ Medium

Careful! Which of the following are **valid** in Python? Write **Valid** or **Error** next to each:

- (a) `x = True` _____
- (b) `x = true` _____
- (c) `x = FALSE` _____
- (d) `x = False` _____
- (e) `x = "True"` _____

For part (e): What is the **type** of `x`? Is it a `bool` or a `str`?

Exercise 8

★★ Medium

Predict the output, then verify:

```
1 >>> a = True
2 >>> b = "True"
3 >>> print(type(a))
4 >>> print(type(b))
5 >>> print(a == b)
```

Your predictions:

`type(a)`: _____

`type(b)`: _____

`a == b`: _____

Exercise 9

★ Easy

Predict the output:

```
1 >>> exercise = True
2 >>> coding = False
3 >>> print(exercise)
4 >>> print(type(coding))
```

Your predictions: _____

Booleans can be stored in variables, just like `int` or `str`.

3 Comparison Operators (~25 min)

Ref: Lecture 4 — Comparison Operators slides

Assignment vs Comparison

Assignment

x = 5

Stores the value 5 into the variable x

Comparison

x == 5

Checks if x equals 5; returns True or False

💡 The Six Comparison Operators

$i < j$ → less than
 $i <= j$ → less than or equal to
 $i > j$ → greater than
 $i >= j$ → greater than or equal to
 $i == j$ → equality test (**not** =)
 $i != j$ → inequality test (not equal)

Each comparison returns either True or False.

Exercise 10

★ Easy

Predict the result of each comparison, then verify:

```

1 >>> 8 == 8
2 >>> 8 == 3
3 >>> 8 != 3
4 >>> 8 != 8
  
```

Your predictions: _____

Exercise 11

★ Easy

Predict the result, then verify:

```

1 >>> 4 < 9
2 >>> 9 < 4
3 >>> 9 > 4
  
```

Your predictions: _____

Exercise 12

★ Easy

Watch out for <= and >=! Predict the result, then verify:

```

1 >>> 7 <= 7
2 >>> 7 >= 8
3 >>> 7 <= 6
  
```

Your predictions: _____

Why is `7 <= 7` True?

Exercise 13

★ Easy

You can compare **expressions**, not just simple values. Predict the result, then verify:

```
1 >>> (3 * 4) == 12
2 >>> len("python") > 4
```

Your predictions: _____

Each side is evaluated first, then compared.

Exercise 14

★★ Medium

Types and comparisons. Predict the result, then verify:

```
1 >>> 5 == 5.0
2 >>> type(5) == type(5.0)
3 >>> "5" == 5
4 >>> len("hello") == 5
```

Your predictions: _____

Why is `5 == 5.0` True but `type(5) == type(5.0)` False?

Exercise 15

★★ Medium

Now let `m = 12` and `n = 5`. Predict the result:

```
1 >>> m = 12
2 >>> n = 5
3 >>> m >= 12
4 >>> m == (n * 2)
5 >>> (m // n) == 2
6 >>> abs(n - m) > 5
```

Your predictions: _____

Exercise 16

★★ Medium

Predict the value **and type** of each:

```
1 >>> 6 + 2
2 >>> 6 + 2 == 8
3 >>> type(6 + 2)
4 >>> type(6 + 2 == 8)
```

Your predictions: _____

Notice how adding `== 8` changes the type from `int` to `bool`.

Exercise 17

★★★ Hard

Tricky! Predict the result, then verify:

```
1 >>> 1.1 + 2.2 == 3.3
2 >>> 1.1 + 2.2
```

Your predictions: _____
 Why does the first line return an unexpected result?

4 Logical Operators: not, and, or (~25 min)

Ref: Lecture 4 — Logical Operators slides

not		and			or		
A	not A	A	B	A and B	A	B	A or B
True	False	True	True	True	True	True	True
False	True	True	False	False	True	False	True
		False	True	False	False	True	True
		False	False	False	False	False	False

and is strict — both must be True *or is generous* — at least one True

Exercise 18

★ Easy

Predict the result of each, then verify:

```
1 >>> not True
2 >>> not False
```

Your predictions: _____

Exercise 19

★ Easy

Predict the result of each, then verify:

```
1 >>> True and True
2 >>> True and False
3 >>> False and True
4 >>> False and False
```

Your predictions: _____

Exercise 20

★ Easy

Predict the result of each, then verify:

```
1 >>> True or True
2 >>> True or False
3 >>> False or True
4 >>> False or False
```

Your predictions: _____

Exercise 21

★★ Medium

Trace through the `not` operator step by step:

```
1 | >>> not (8 > 2)
```

Step 1: Evaluate inside the parentheses: `8 > 2 = _____`Step 2: Apply `not`: `not ___ = _____`

Exercise 22

★★ Medium

Trace through `and` step by step:

```
1 | >>> (7 > 4) and (1 > 6)
```

Step 1: Evaluate left side: `7 > 4 = _____`Step 2: Evaluate right side: `1 > 6 = _____`Step 3: Combine: `___ and ___ = _____`

Exercise 23

★★ Medium

Trace through `or` step by step:

```
1 | >>> (8 < 3) or (4 == 4)
```

Step 1: Evaluate left side: `8 < 3 = _____`Step 2: Evaluate right side: `4 == 4 = _____`Step 3: Combine: `___ or ___ = _____`

Exercise 24

★★ Medium

A library lets you borrow a book if you have a **valid card** **and** you have **no overdue books**. Predict the output:

```
1 | valid_card = True
2 | overdue_books = 3
3 | can_borrow = valid_card and (overdue_books == 0)
4 | print(can_borrow)
```

Your prediction: _____

Now change `overdue_books = 3` to `overdue_books = 0`. What does `can_borrow` become?

This is how programs make real decisions using boolean logic.

Exercise 25

★★★ Hard

not binds first! The `not` operator applies to the **next value only**, before `and/or`. Trace through carefully:

```
1 | >>> not True and False
2 | >>> not False or True
```

For the first expression:

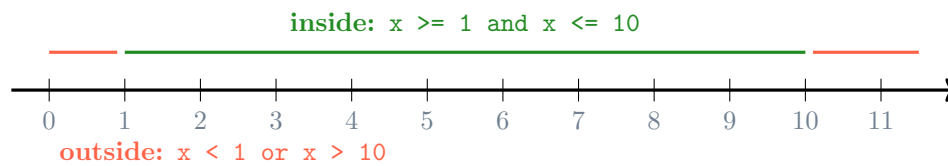
Step 1: `not True = _____`Step 2: `___ and False = _____`

For the second expression:

Step 1: `not False = _____`Step 2: `___ or True = _____`

5 Compound Boolean Expressions (~25 min)

Ref: Lecture 4 — Compound Boolean Expressions slides



Exercise 26

★ Easy

Predict the result, then verify:

```
1 >>> x = 5
2 >>> x >= 1 and x <= 10
```

Your prediction: _____
Is x inside the range 1 to 10?

Exercise 27

★ Easy

Predict the result, then verify:

```
1 >>> x = 15
2 >>> x < 1 or x > 10
```

Your prediction: _____
Is x outside the range 1 to 10?

Exercise 28

★★ Medium

Translate each English phrase into a Python boolean expression. Let `score = 85` and `hours = 10`.

- (a) “score is between 60 and 100”
Your expression: _____
- (b) “hours is above 8 or score is above 90”
Your expression: _____
- (c) “score is NOT below 50”
Your expression: _____

Type each expression in the Shell to verify.

Exercise 29

★★ Medium

Predict the output of this program, then verify:

```
1 study_hours = 6
2 sleep_hours = 9
3 print(study_hours > sleep_hours)
4 morning_person = True
```

```

5 | has_coffee = False
6 | ready = morning_person and has_coffee
7 | print(ready)

```

Line 3 prints: _____ Line 6 prints: _____

Exercise 30

★★ Medium

Tricky range mistake! Predict the result for each, with $x = 15$:

```

1 | >>> x = 15
2 | >>> x < 1 and x > 10
3 | >>> x < 1 or x > 10

```

Your predictions: _____

Why is the first expression **always** False regardless of x ?

*Hint: Can x be less than 1 **and** greater than 10 at the same time?*

Exercise 31

★★ Medium

Fix the bug! A student wrote this code to check if `age` is that of a teenager (between 13 and 19). But it has a logic error:

```

1 | age = 13
2 | is_teen = age > 13 and age < 19
3 | print(is_teen)

```

What does it print? _____

The student expected True (13 is a teenager!). What should line 2 be changed to?

Your fix: _____

Hint: Think about $>$ vs $>=$.

6 Operator Precedence (~15 min)

Ref: Lecture 4 — Operator Precedence slides

💡 Order of Operations

Python evaluates operators in this order (highest priority first):

Priority	Operators	Example
1 (highest)	**	2 ** 3
2	* / // %	6 / 2
3	+ -	3 + 4
4	< <= > >= == !=	x > 5
5	not	not True
6	and	a and b
7 (lowest)	or	a or b

Remember: arithmetic → comparison → not → and → or

Exercise 32

★ Easy

Add parentheses to show the order of evaluation, then predict the result:

```
1 | >>> 5 > 3 and 2 < 4
```

With parentheses: _____

Result: _____

Exercise 33

★ Easy

Add parentheses to show the order, then predict:

```
1 | >>> not 3 > 5
```

With parentheses: _____

Result: _____

Exercise 34

★★ Medium

Trace through this expression step by step:

```
1 | >>> 4 + 1 > 3 and not True
```

Step 1 (arithmetic): $4 + 1 =$ _____Step 2 (comparison): $___ > 3 =$ _____Step 3 (not): $\text{not True} =$ _____Step 4 (and): $___ \text{ and } ___ =$ _____

Exercise 35

★★ Medium

Careful! `and` binds tighter than `or`. Predict the result:

```
1 | >>> False or True and True
```

Step 1 (and first): $\text{True and True} =$ _____Step 2 (or): $\text{False or } ___ =$ _____*What if it were (False or True) and True instead?*

Exercise 36

★★ Medium

Add parentheses, then evaluate step by step:

```
1 | >>> 7 > 2 and 3 * 2 == 6
2 | >>> not 6 < 1 or 8 > 9
```

First expression result: _____

Second expression result: _____

Show your work:
 **When in Doubt, Use Parentheses!**

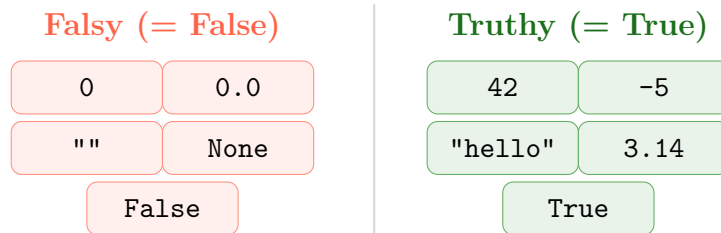
Parentheses make your code **readable** and remove ambiguity. Even if they're not technically needed, they help:

not True or False
(not True) or False

← confusing
← crystal clear

7 Truthy, Falsy & String Comparison (~15 min)

Ref: Lecture 4 — Truthy and Falsy / Comparing Strings slides



Use `bool()` to check: `bool(0) → False`, `bool(42) → True`

Exercise 37

★ Easy

Predict the result, then verify:

```
1 >>> bool(99)
2 >>> bool(0)
3 >>> bool(-3)
4 >>> bool(0.0)
5 >>> bool(None)
6 >>> bool("")
```

Your predictions: _____

Rule: zero, 0.0, empty string, None, and False are Falsy. Everything else is Truthy.

Exercise 38

★★ Medium

Careful with strings! Predict the result, then verify:

```
1 >>> bool("python")
2 >>> bool("")
3 >>> bool("None")
4 >>> bool(" ")
```

Your predictions: _____

Why is `bool("None")` not False? Why is `bool(" ")` not False?

Exercise 39

★★ Medium

Truthy in action! Predict the output of this program, then verify:

```
1 name = input("Enter your name: ")
2 if name:
3     print(f"Hello, {name}!")
```

```

4 else:
5     print("You didn't enter a name.")

```

Run it twice: once entering Ali, and once pressing Enter without typing anything.

With Ali: _____

With empty input: _____

An empty string is Falsy, so it acts like False in an if condition.

Exercise 40

★ Easy

Predict the result, then verify:

```

1 >>> "cat" == "cat"
2 >>> "cat" == "Cat"
3 >>> "dog" != "cat"

```

Your predictions: _____

String comparison is case-sensitive!

Exercise 41

★★ Medium

Strings are compared character by character (like a dictionary). Predict the result, then verify:

```

1 >>> "mango" < "orange"
2 >>> "c" < "d"
3 >>> "B" < "b"
4 >>> "Y" < "b"

```

Your predictions: _____

*Uppercase letters always come **before** lowercase: "A" < "Z" < "a" < "z".*

8 Introduction to if/else (~20 min)

Preview — Coming up in Lecture 5

💡 How if/else Works

Now that you know how to write boolean expressions, you can use them to make **decisions** in your programs. The `if` statement runs a block of code **only when** a condition is `True`:

```
1 if condition:
2     # runs when condition is True
3 else:
4     # runs when condition is False
```

Important: The indented lines (4 spaces) are the **body**. Python uses indentation to know which lines belong to the `if` or `else`.

Exercise 42

★ Easy

Type this program into the code editor and run it. Try it **twice**: once entering 25, and once entering 10.

```
1 age = int(input("Enter your age: "))
2 if age >= 18:
3     print("You are an adult")
4 else:
5     print("You are a minor")
```

When you enter 25, it prints: _____

When you enter 10, it prints: _____

Only ONE branch runs — never both.

Exercise 43

★ Easy

Predict the output, then type it in to verify:

```
1 x = 15
2 if x > 20:
3     print("big")
4 else:
5     print("small")
```

Your prediction: _____

Now change `x = 15` to `x = 30` and predict again: _____

Exercise 44

★ Easy

`if` can be used **without** `else`. Predict the output:

```
1 temp = 40
2 if temp > 35:
3     print("It is hot today!")
4 print("Have a nice day")
```

Your prediction: _____

Now change `temp = 40` to `temp = 20`. What prints now?

*The last `print` is **not** indented, so it runs **always**.*

Exercise 45

★ Easy

Predict the output, then verify:

```

1 logged_in = False
2 if not logged_in:
3     print("Please log in")
4 else:
5     print("Welcome back")

```

Your prediction: _____

Now change `logged_in = False` to `logged_in = True` and predict again:

The `not` operator flips the condition.

Exercise 46

★ Easy

Predict the output, then verify:

```

1 color = "red"
2 if color == "blue":
3     print("Sky")
4 else:
5     print("Not sky")

```

Your prediction: _____

Now change `color = "red"` to `color = "blue"` and predict again: _____

Exercise 47

★ Easy

Predict the output, then verify:

```

1 a = 4
2 b = 9
3 if a + b > 10:
4     print("More than ten")
5 else:
6     print("Ten or less")

```

Your prediction: _____

What is `a + b`? _____ Is it greater than 10? _____

Exercise 48

★ Easy

Predict the output:

```

1 raining = True
2 if raining:
3     print("Take an umbrella")
4 else:
5     print("Enjoy the sun")

```

Your prediction: _____

Now change `raining = True` to `raining = False` and predict again:

A boolean variable can be used directly as the condition.

Exercise 49

★ Easy

Predict the output, then verify:

```
1 marks = 85
2 if marks >= 50:
3     print("You passed!")
4     print("Well done!")
5 print("Goodbye")
```

Your prediction:

Now change `marks = 85` to `marks = 30`. Which lines still print?

*Both indented lines belong to the `if` body. "Goodbye" is not indented, so it runs **always**.*

9 Capstone Challenges (~20 min)

These problems combine everything you have learned. Write your solutions in Thonny's code editor.

Exercise 50

★★ Medium

Movie Ticket Discount. Write a program that:

1. Asks the user for their age
2. Prints whether they get a child discount (age 12 or below)
3. Prints whether they get a senior discount (age 60 or above)

Example (user enters 8):

```
1 Child discount: True
2 Senior discount: False
```

Example (user enters 65):

```
1 Child discount: False
2 Senior discount: True
```

Hint: Use two separate comparison expressions.

Write your code:

Exercise 51

★★ Medium

Password Length Check. Write a program that:

1. Asks the user for a password
2. Prints whether the password is at least 8 characters long
3. Prints whether the password is between 8 and 20 characters (inclusive)

Example (user enters hello123):

```
1 Long enough: True
2 Valid length: True
```

Write your code:

Exercise 52

★★ Medium

Grade Check. Write a program that:

1. Asks the user for their marks (out of 100)
2. Prints whether their marks are passing (50 or above)
3. Prints whether their marks qualify for distinction (75 or above)

Example (user enters 82):

```
1 Passing: True
2 Distinction: True
```

Write your code:

Exercise 53

★★★ Hard

Leap Year. A year is a leap year if:

- It is divisible by 4, **AND**
- It is NOT divisible by 100, **OR** it is divisible by 400

Write a program that:

1. Asks the user for a year
2. Computes a boolean expression for whether it's a leap year
3. Prints the result

Example (user enters 2024):

```
1 2024 is a leap year: True
```

Example (user enters 1900):

```
1 1900 is a leap year: False
```

Hint: Use % (modulo). A number is divisible by n if $x \% n == 0$.

Write your code:

Exercise 54

★★★ Hard

Triangle Validator. Three side lengths form a valid triangle if and only if the sum of any two sides is greater than the third side.

Write a program that:

1. Asks the user for three side lengths
2. Prints whether they form a valid triangle

Example (user enters 3, 4, 5):

```
1 Valid triangle: True
```

Example (user enters 1, 2, 10):

```
1 Valid triangle: False
```

Hint: You need three conditions combined with `and`.

Write your code:

Self-Assessment Checklist

Before you leave, check off what you can do:

- I can use `print()` with commas and concatenation, and I know the difference
- I know that `input()` always returns a `str` and I must cast it for arithmetic
- I can use f-strings to format output with expressions
- I know that `bool` has exactly two values: `True` and `False` (case-sensitive)
- I can use all six comparison operators: `==`, `!=`, `<`, `>`, `<=`, `>=`
- I know the difference between `=` (assignment) and `==` (comparison)
- I can use `not`, `and`, and `or` and I know their truth tables
- I can write compound boolean expressions for range checks
- I know the full operator precedence: arithmetic \rightarrow comparison \rightarrow `not` \rightarrow `and` \rightarrow `or`
- I know which values are Truthy and which are Falsy
- I can compare strings (case-sensitive, lexicographic order)
- I can write a basic `if/else` statement and I understand that indentation defines the body

- When in doubt, I use **parentheses** to make my expressions clear