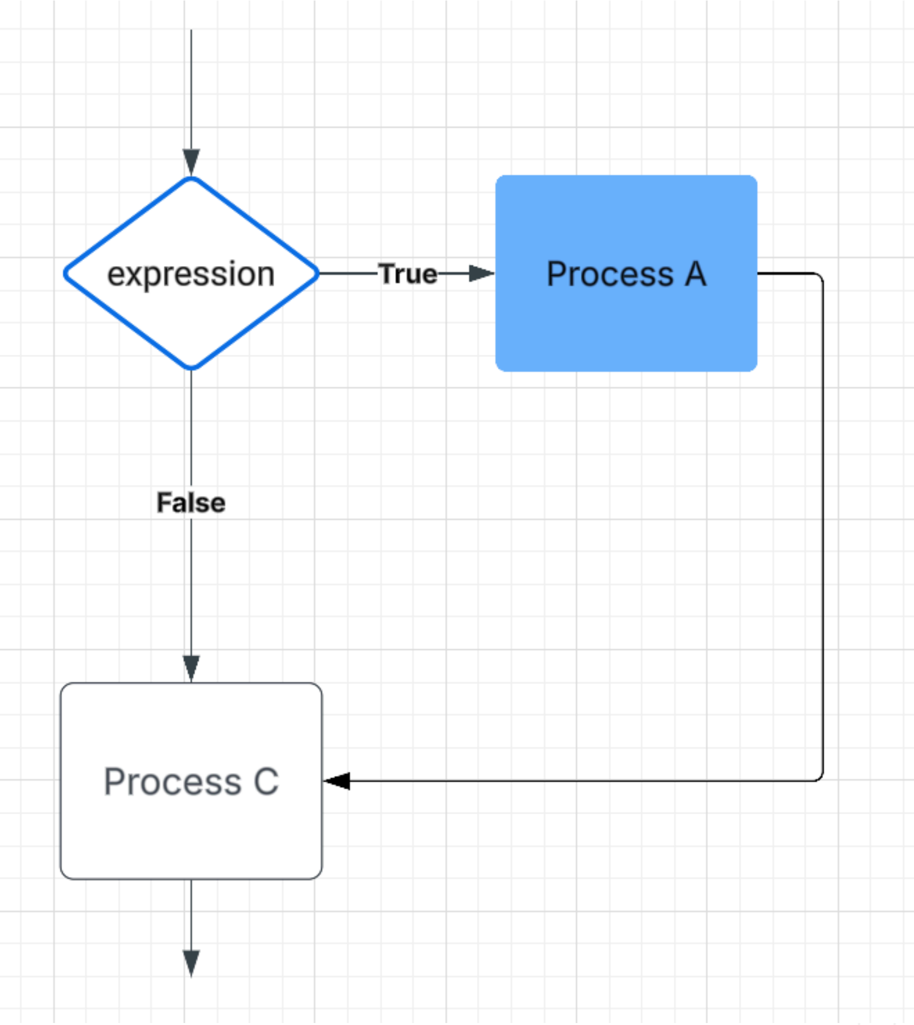
Ch4: Branching

by Lina Liu

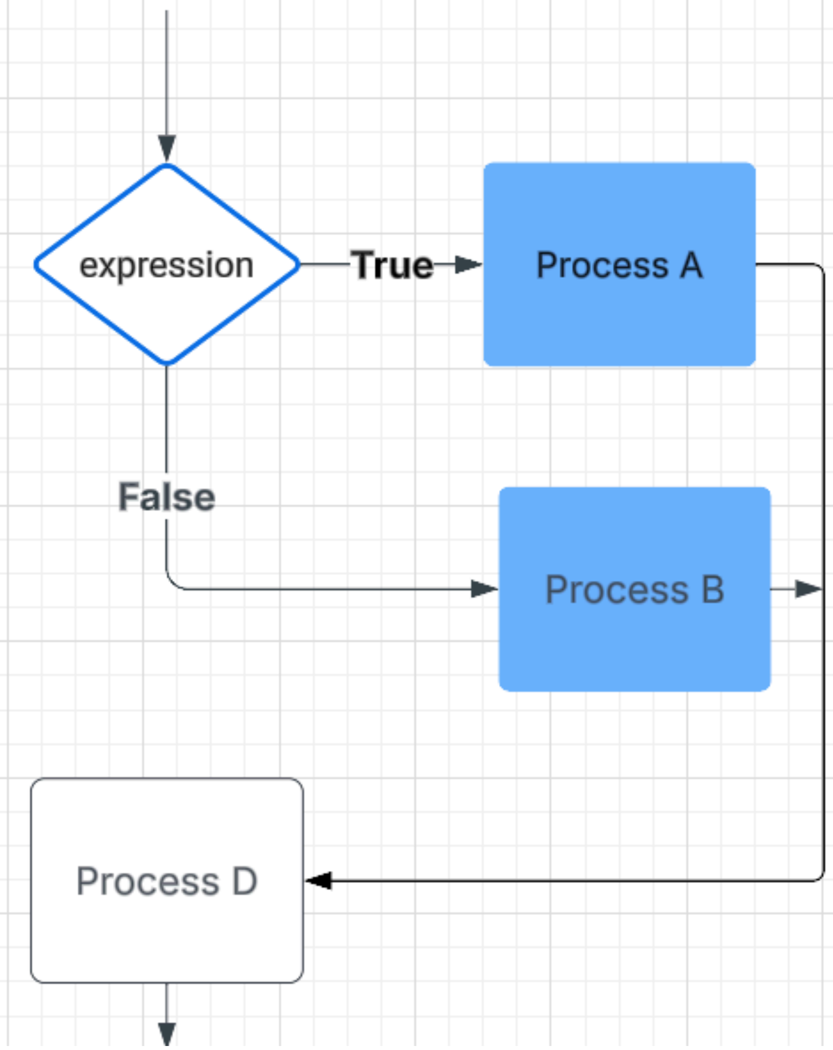
Review of Key Topics

- If-else branches (general)
- Detecting equal values with branches
- Detecting ranges with branches (general)
- Detecting ranges with branches
- Detecting ranges using logical operators
- Detecting ranges with gaps
- Detecting multiple features with branches
- Comparing data types and common errors
- Membership and identity operators
- Order of evaluation
- Code blocks and indentation
- Conditional expressions

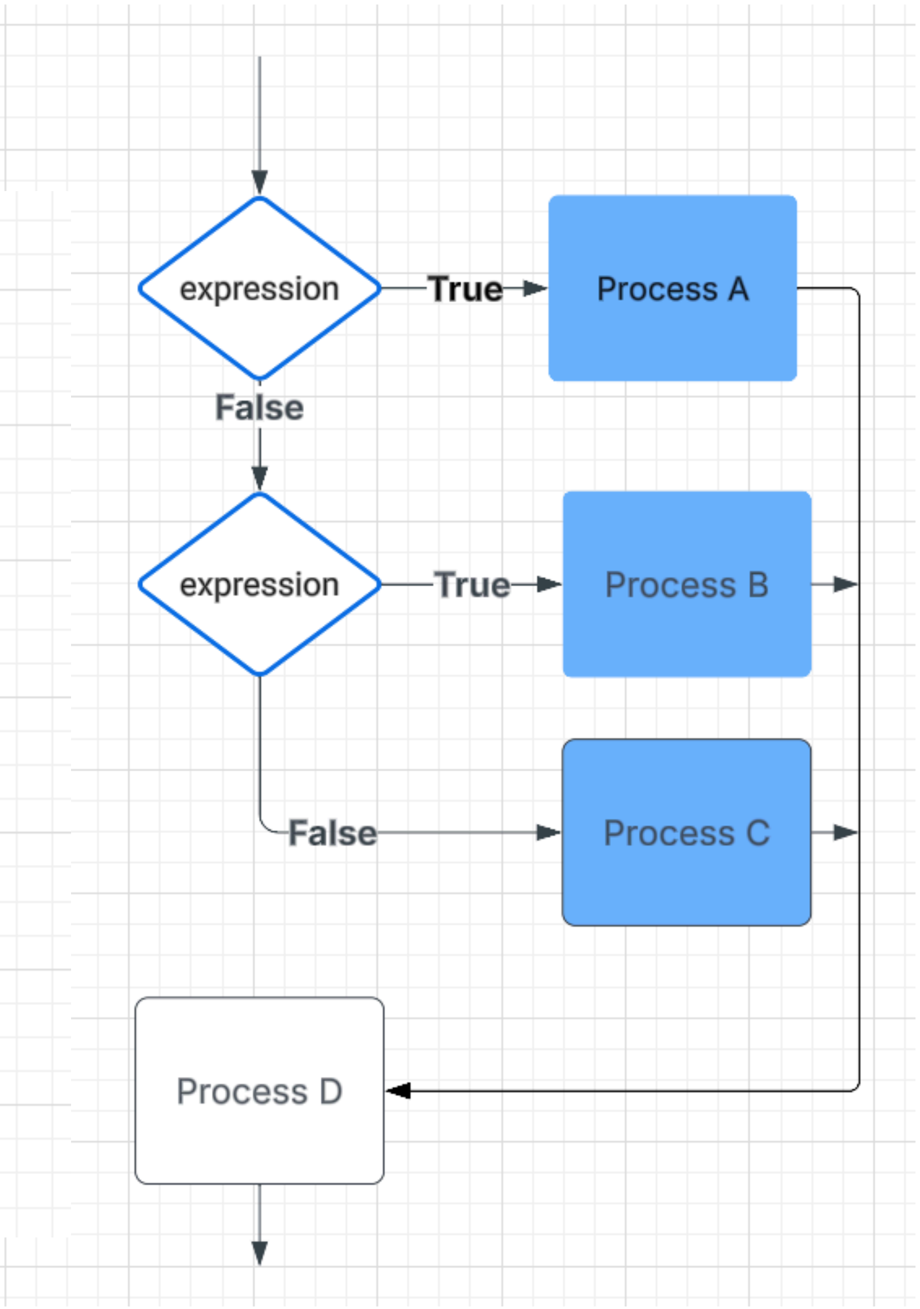
If-else branches (general)



Branch basics (If)

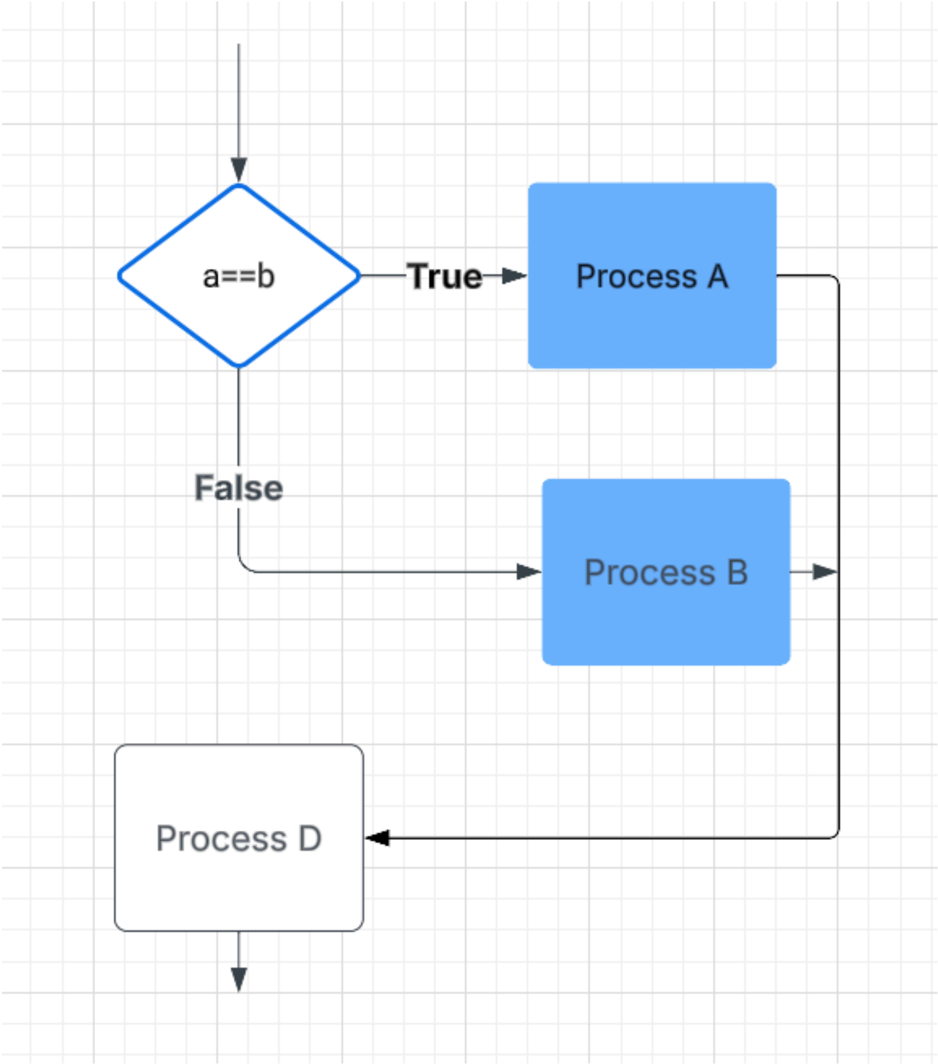


If-else branches

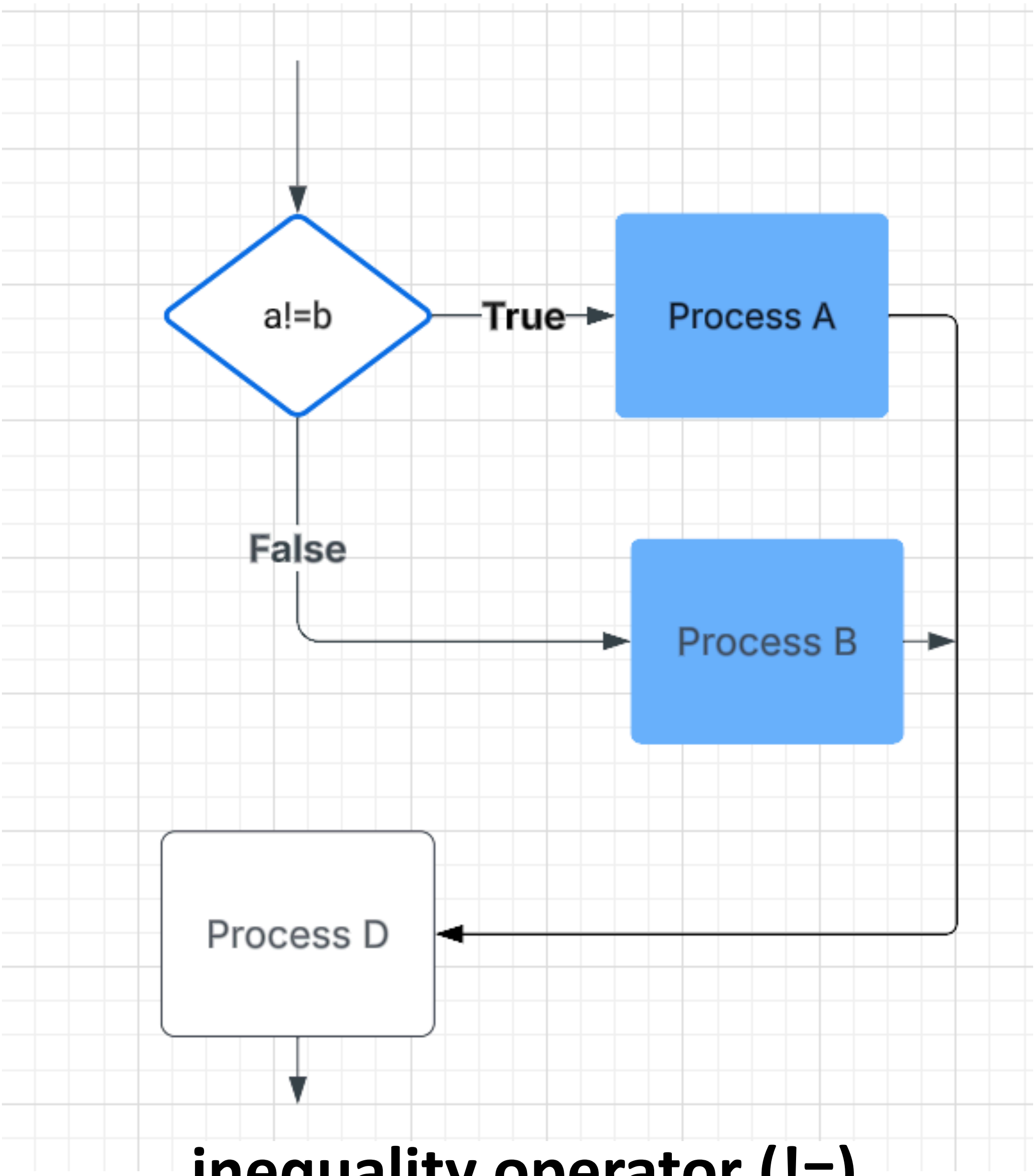


If-elseif-else branches

Detecting equal values with branches

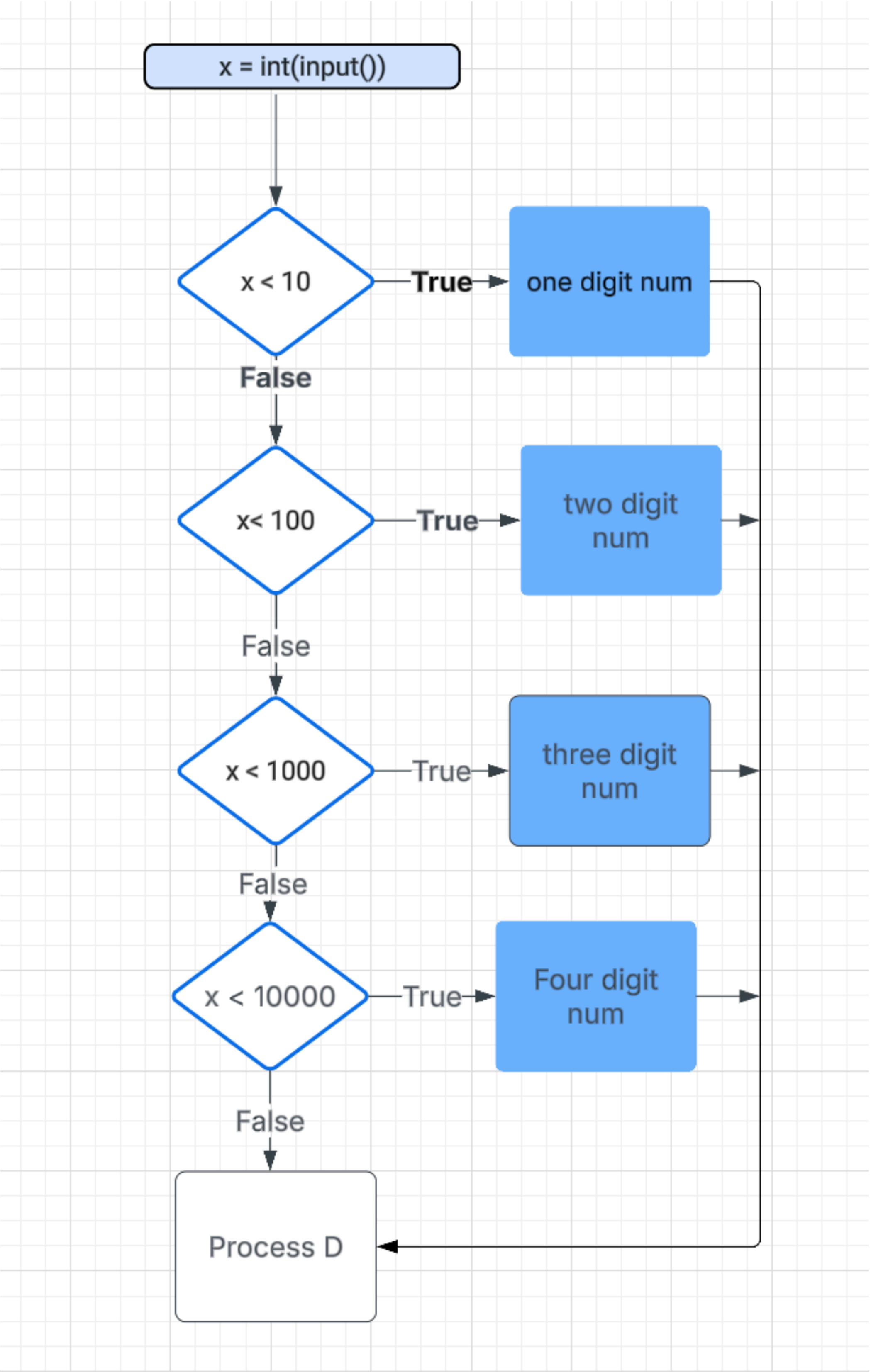


equality operator (==) with branches

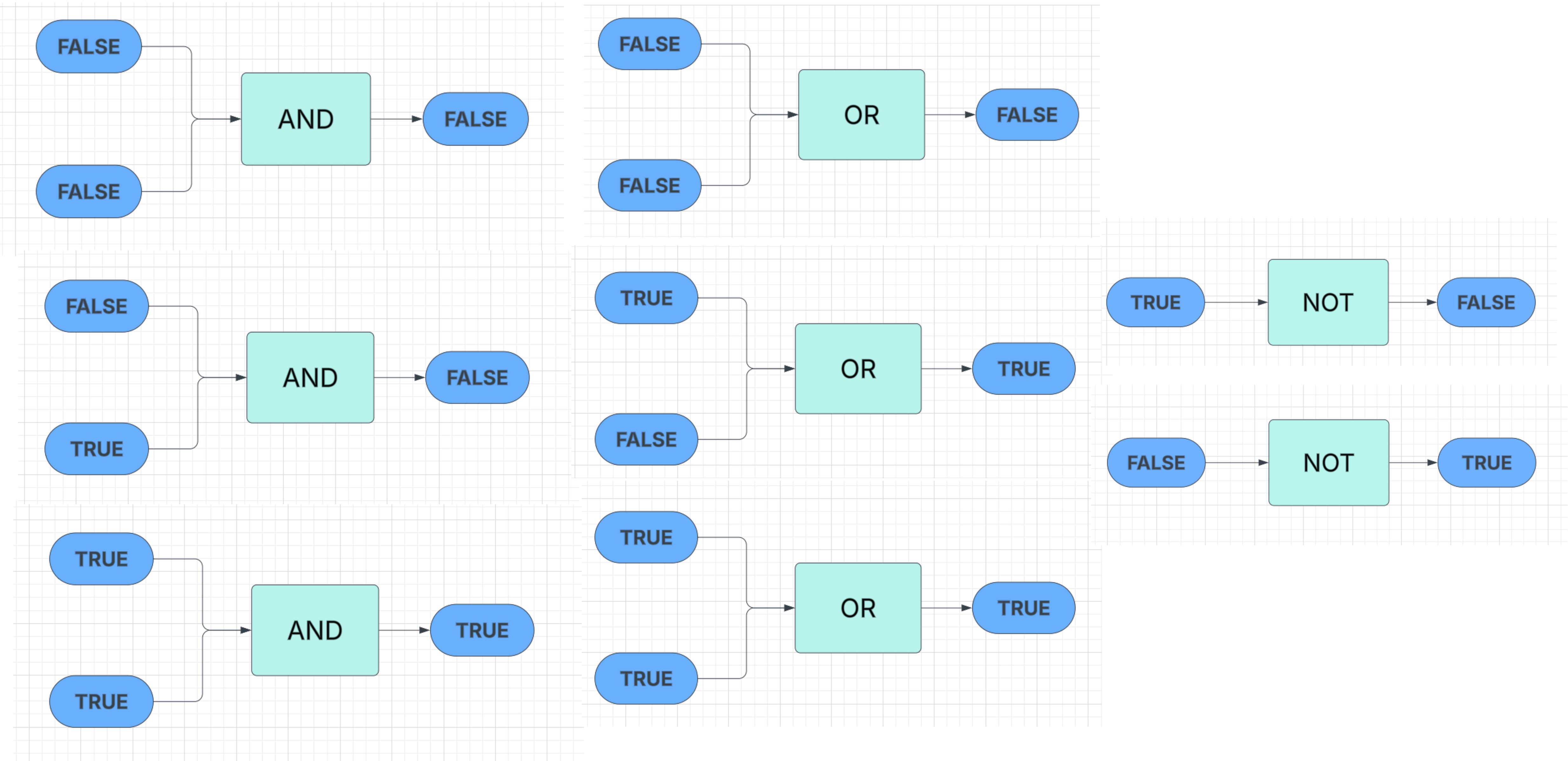


inequality operator (!=) with branches

Detecting ranges with branches (general)



Detecting ranges using logical operators

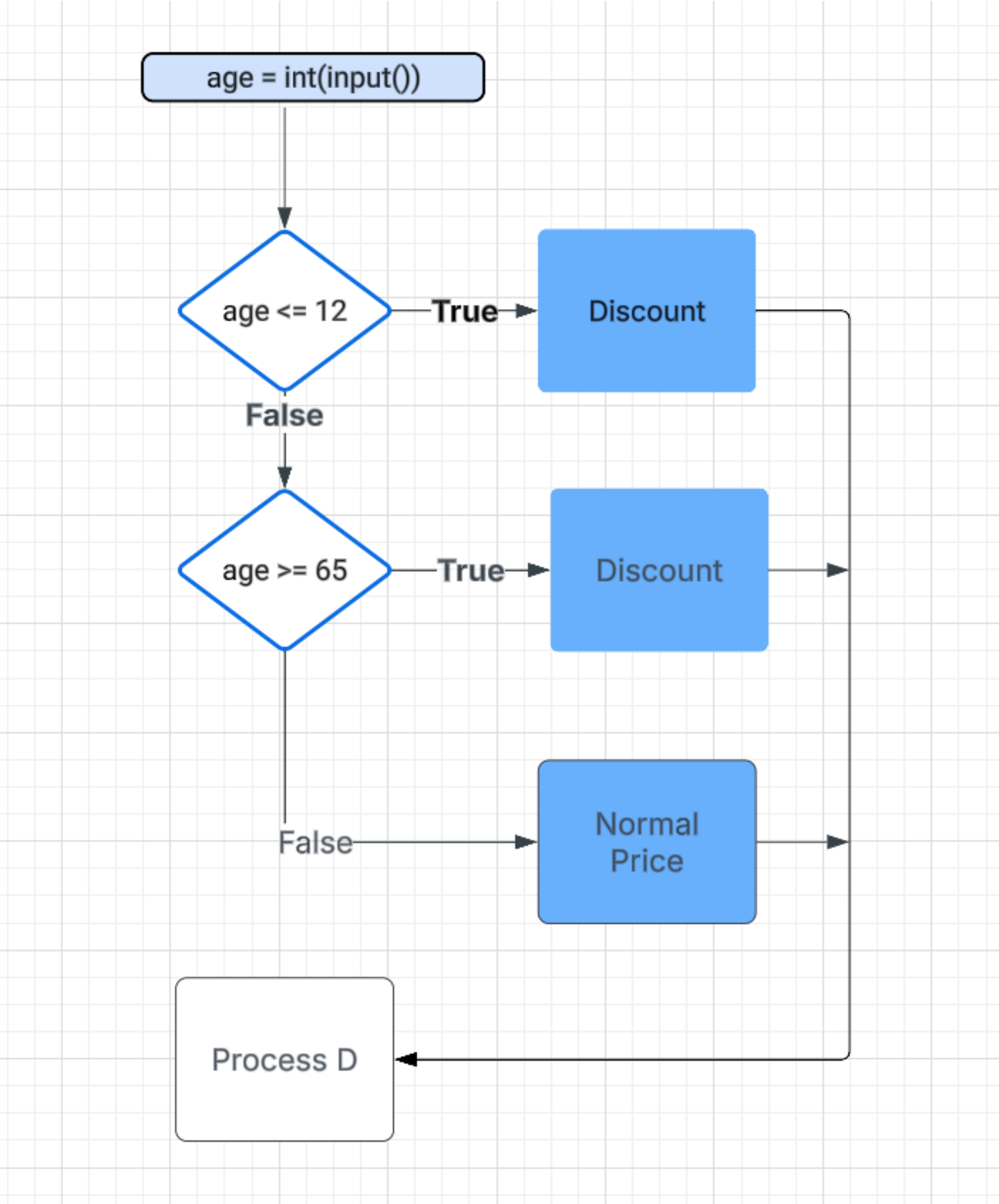


Logical AND: True when both of its operands are True.

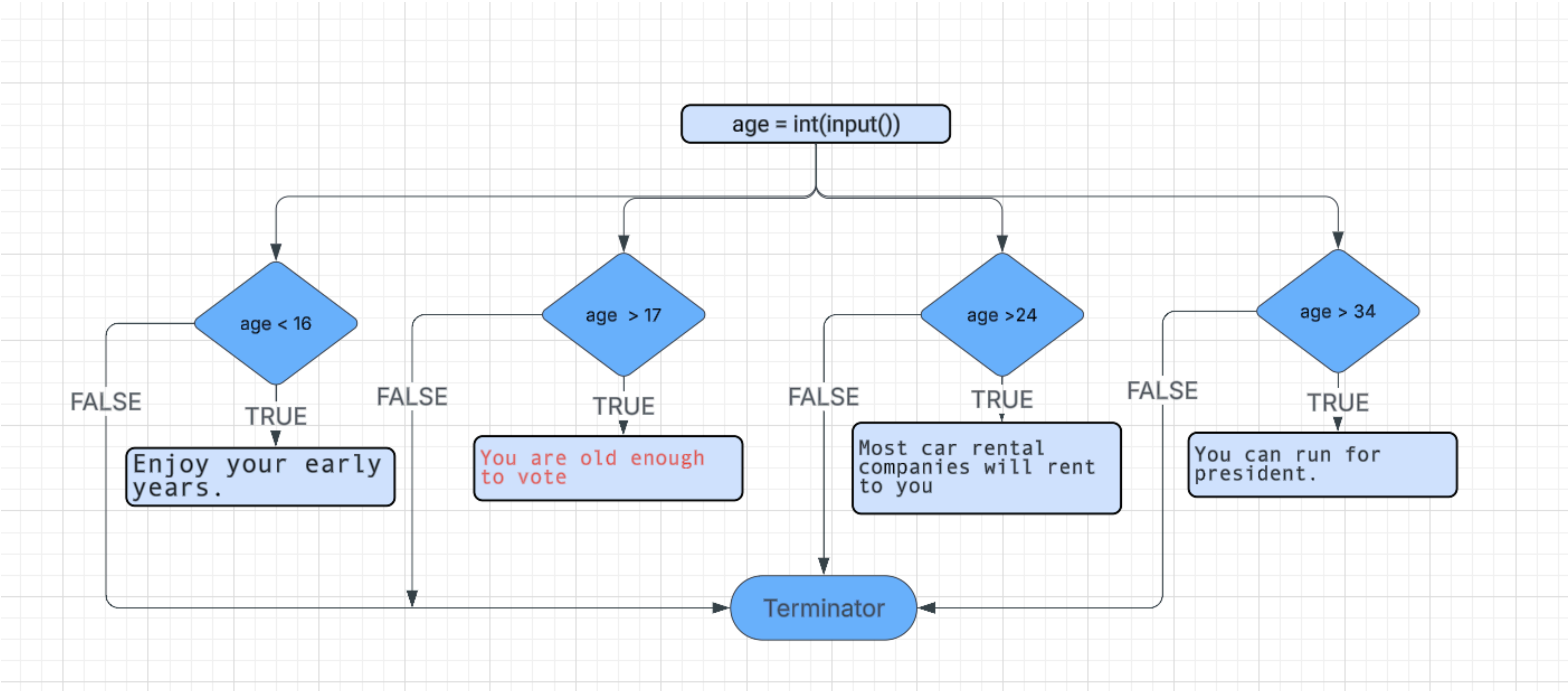
Logical OR: True when at least one of its two operands are True.

Logical NOT: True when its one operand is False, and vice versa.

Detecting ranges with gaps



Detecting multiple features with branches



Comparing data types and common errors

`my_int == 42` ✓

`my_float == 3.14` ⛔

unexpected results

`my_string == 'Hello'` ✓

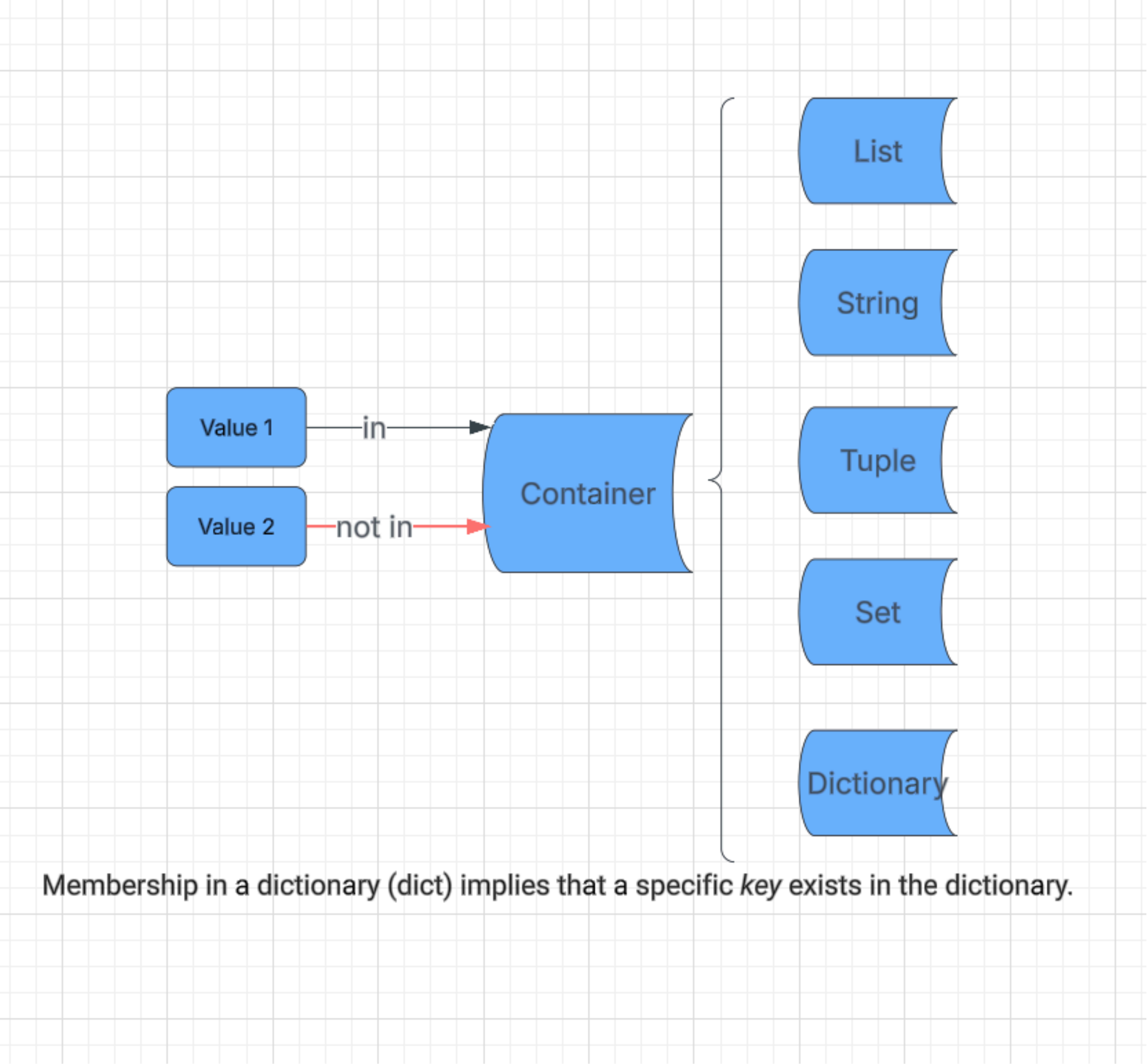
`{'Henrik': '$25'} == {'Daniel': '$25'}` FALSE

`(1,2,3) > (0,2,3)` TRUE

`[1, 2, 3] >= ['1', '2', '3']` ⛔

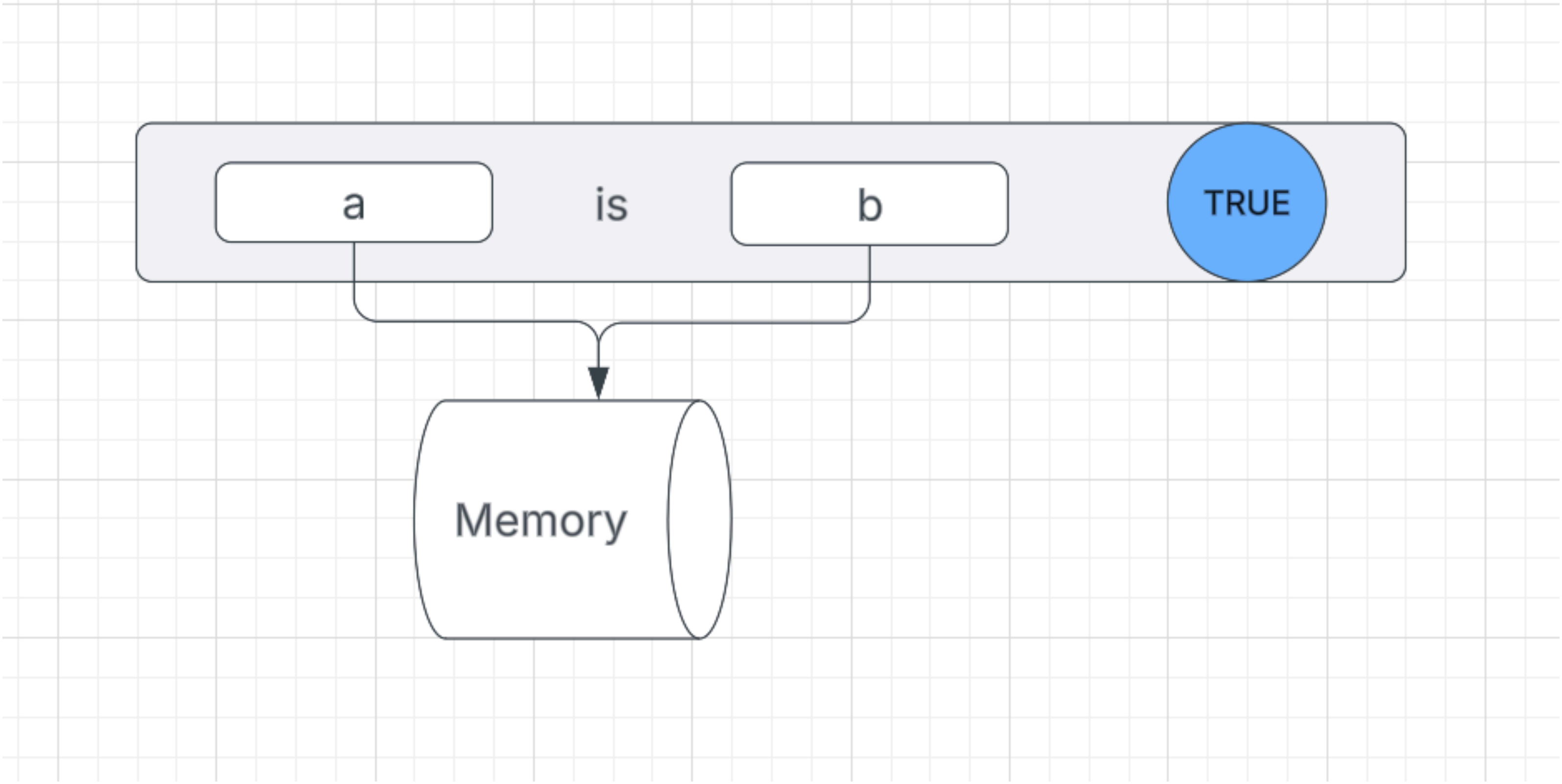
ERROR

Membership operators: in/not in



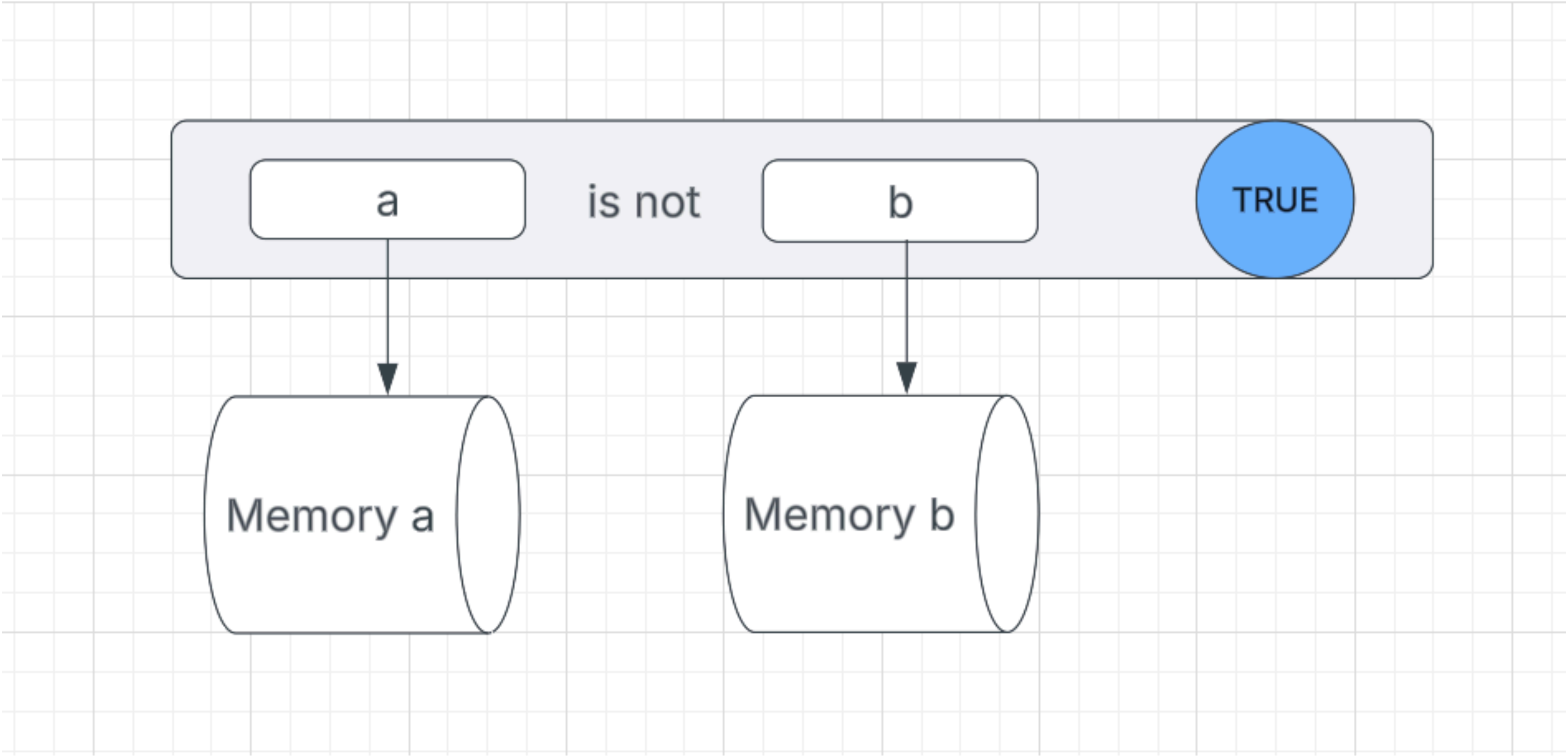
Membership in a dictionary (dict) implies that a specific *key* exists in the dictionary.

Identity operators: is/is not



a is b TRUE

Memory

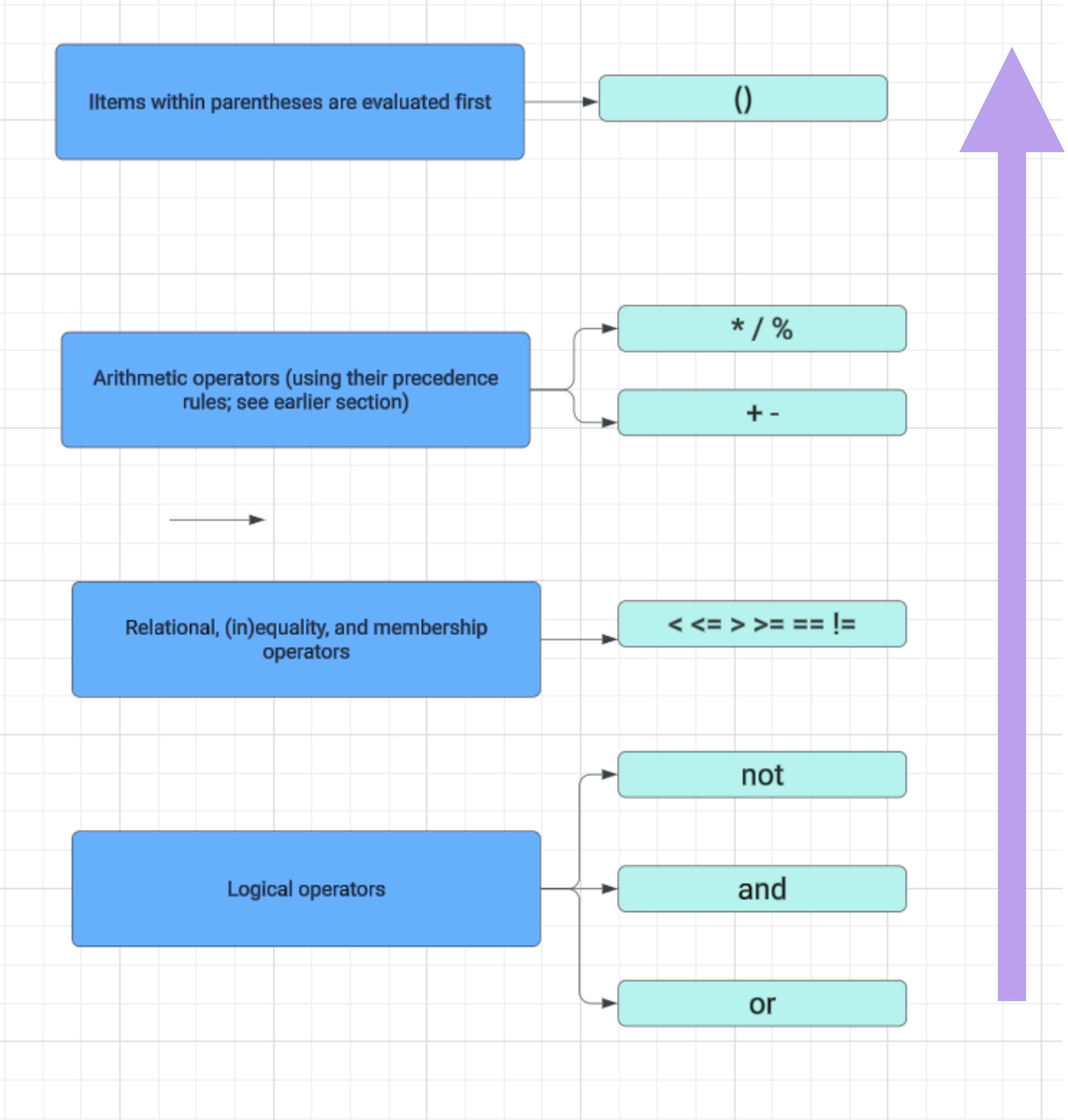


a is not b TRUE

Memory a

Memory b

Order of evaluation



IItems within parentheses are evaluated first

()

Arithmetic operators (using their precedence rules; see earlier section)

* / %

+ -

Relational, (in)equality, and membership operators

< <= > >= == !=

Logical operators

not

and

or

Code blocks and indentation

Code Blocks Defined by Indentation:

- Python uses indentation (spaces) to group statements into code blocks.
- The level of indentation determines which statements belong to the same block.

Initial Code Block:

- The main, top-level code block is not indented.

Colon-Ended Statements:

- Statements like if, else, for, while, def, and class that end with a colon (:) introduce a new, indented code block.

Increased Indentation:

- A new code block must be more indented than the preceding block.

Consistent Indentation:

- The amount of indentation is arbitrary, but it must be consistent within a code block.

Recommended Indentation:

- The standard recommendation is to use four spaces per indentation level.

Tabs vs. Spaces (Crucial):

- **Never** mix tabs and spaces for indentation.
- Python treats tabs differently from spaces, leading to IndentationError if mixed.
- Use spaces exclusively for indentation.
- Set your text editor to replace tabs with spaces.

IndentationError:

- Mixing tabs and spaces will cause an IndentationError.

Conditional expressions

```
if condition:
   my_var = expr1
else:
   my_var = expr2
```

↕

```
my_var = expr1 if (condition) else  expr2
```