COSC 1306 - Prog for Non-Majors

Decision Structures

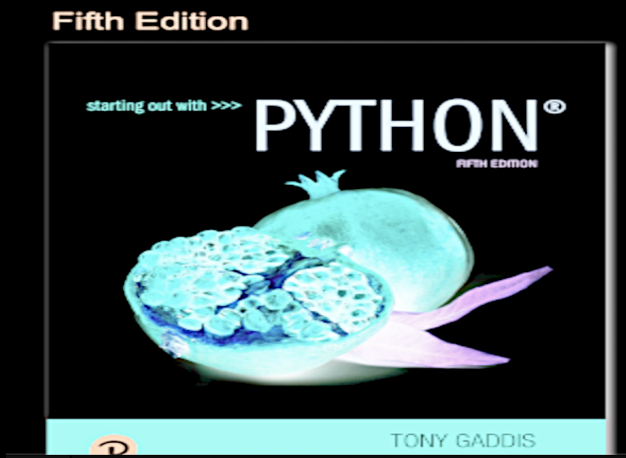
Dr. Mohan

McMurry University

September 26, 2023

Lesson Topics Overview (1 of 2)

- The if Statement.
- The if-else Statement
- Comparing Strings.
- Nested Decision Structures and the if-elif-else Statement.
- Logical Operators.
- Boolean Variables.



Chapter 3

Decision Structures and Boolean Logic

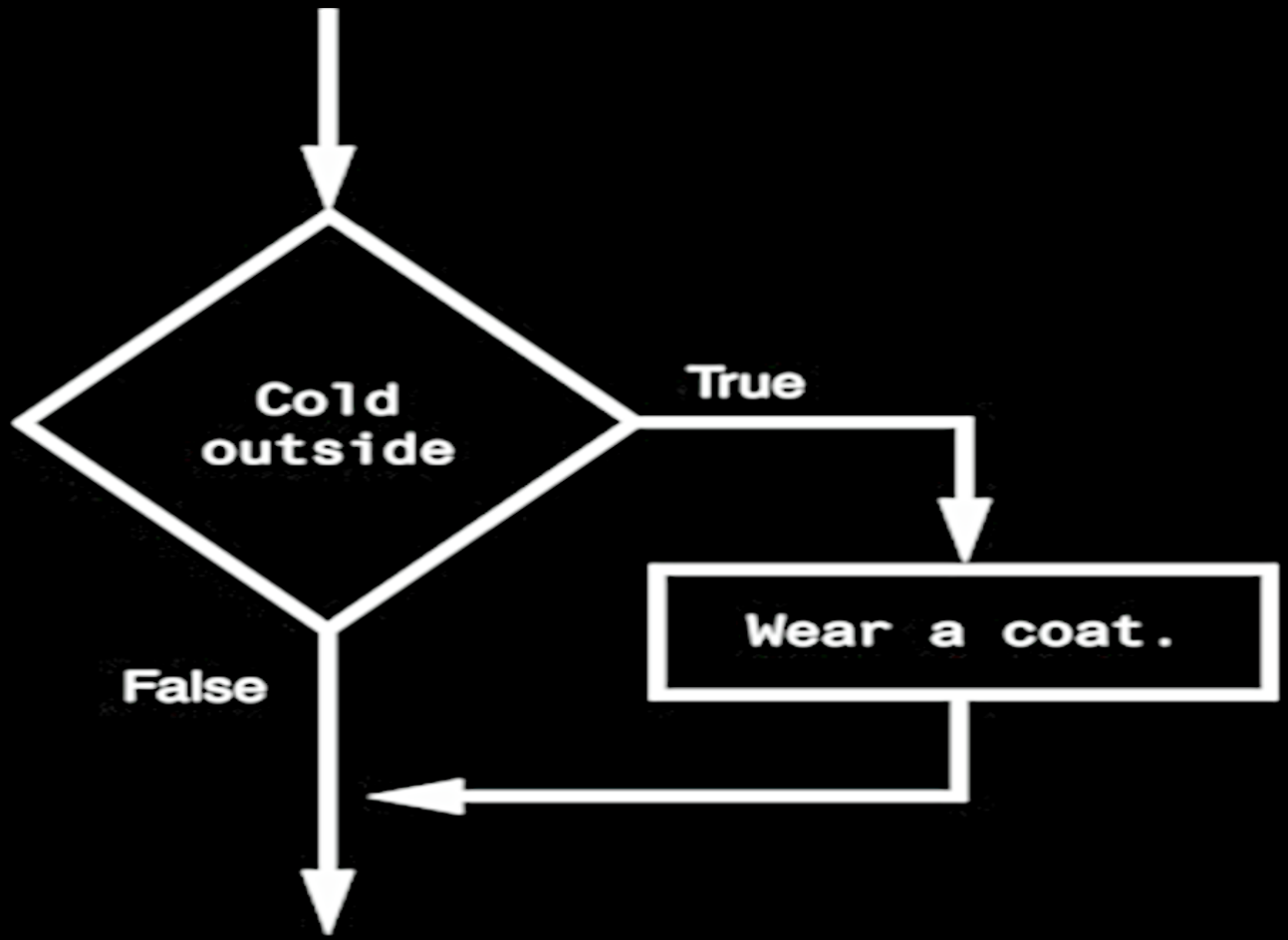
The if Statement (1 of 4)

- **Control structure:** logical design that controls order in which set of statements execute.
- Sequence structure: set of statements that execute in the order they appear.
- Decision structure: specific action(s) performed only if a condition exists. Also known as selection structure.

The if Statement (2 of 4)

- In flowchart, diamond represents true/false condition that must be tested.
- Actions can be conditionally executed and performed only when a condition is true.
- **Single alternative decision structure:** provides only one alternative path of execution. If condition is not true, exit the structure.

The if Statement (3 of 4)



The if Statement (4 of 4)

- Python syntax:
  - **if condition:**
    - **Statement**
    - **Statement**
- First line known as the if clause that includes the keyword if followed by condition.
- The condition can be true or false
- When the if statement executes, the condition is tested, and if it is true the block statements are executed. otherwise, block statements are skipped.

Boolean Expressions and Relational Operators (1 of 5)

- **Boolean expression:** expression tested by if statement to determine if it is true or false.

  Example: a $>$ b.

  true if a is greater than b;

  false otherwise.
- Relational operator: determines whether a specific relationship exists between two values.

  Example: greater than ($>$).

Boolean Expressions and Relational Operators (2 of 5)

- $>=$ and $<=$ operators test more than one relationship. It is enough for one of the relationships to exist for the expression to be true.
- $==$ operator determines whether the two operands are equal to one another. Do not confuse with assignment operator (=).
- $!=$ operator determines whether the two operands are not equal.

Boolean Expressions and Relational Operators (3 of 5)

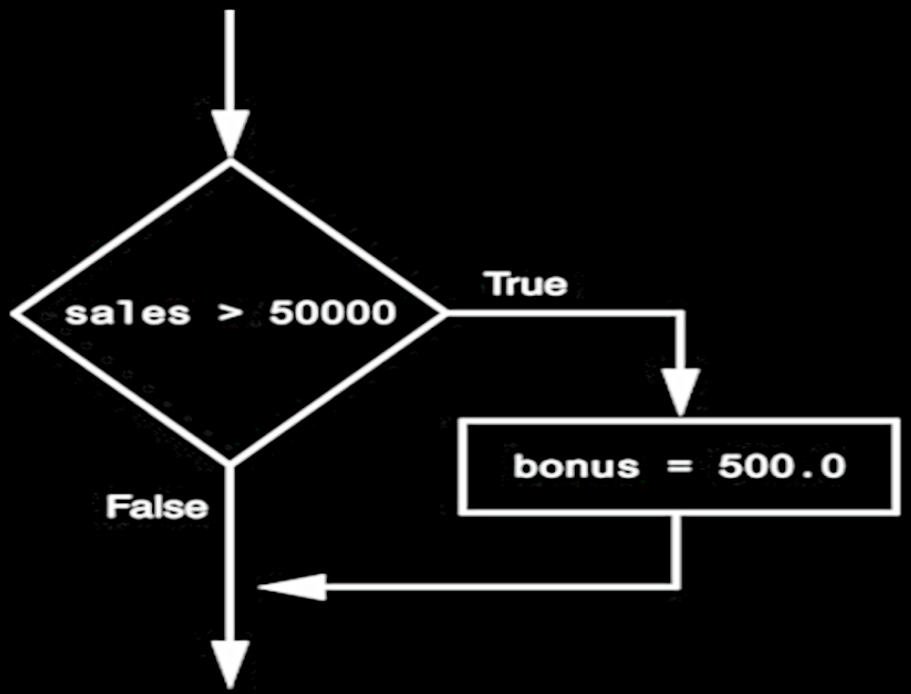
Table 3-2 Boolean expressions using relational operators

Expression	Meaning
`x > y`	Is `x` greater than `y`?
`x < y`	Is `x` less than `y`?
`x >= y`	Is `x` greater than or equal to `y`?
`x <= y`	Is `x` less than or equal to `y`?
`x == y`	Is `x` equal to `y`?
`x != y`	Is `x` not equal to `y`?

Boolean Expressions and Relational Operators (4 of 5)

- Using a Boolean expression with the $>$ relational operator.

Figure 3-3 Example decision structure



Boolean Expressions and Relational Operators (5 of 5)

- Any relational operator can be used in a decision block

  Example: if balance == 0

  Example: if payment != balance

- It is possible to have a block inside another block.

  Example: if statement inside a function

  Statements in inner block must be indented with respect to the outer block.

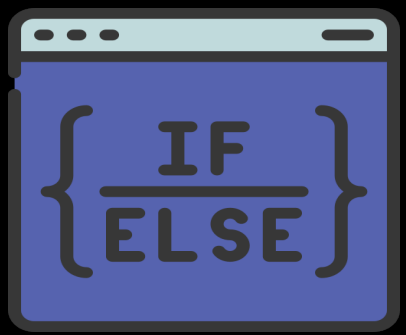
The if-else Statement (1 of 3)

- **Dual alternative decision structure:**

  Two possible paths of execution:
  - One is taken if the condition is true, and the other if the condition is false.
  - Syntax:

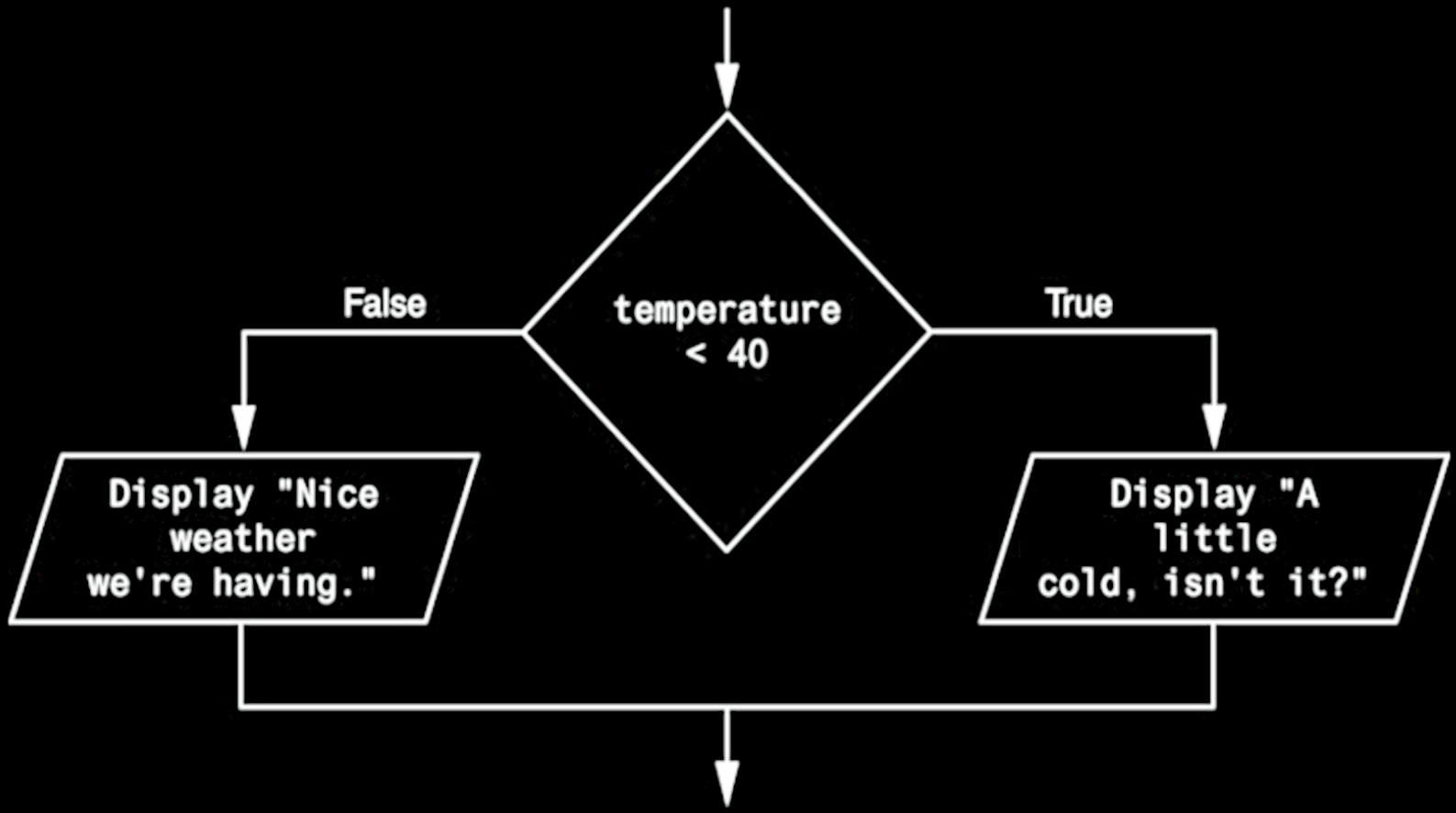
```
if condition:
    statements
else:
    other statements
```

  - if clause and else clause must be aligned.
  - Statements must be consistently indented.



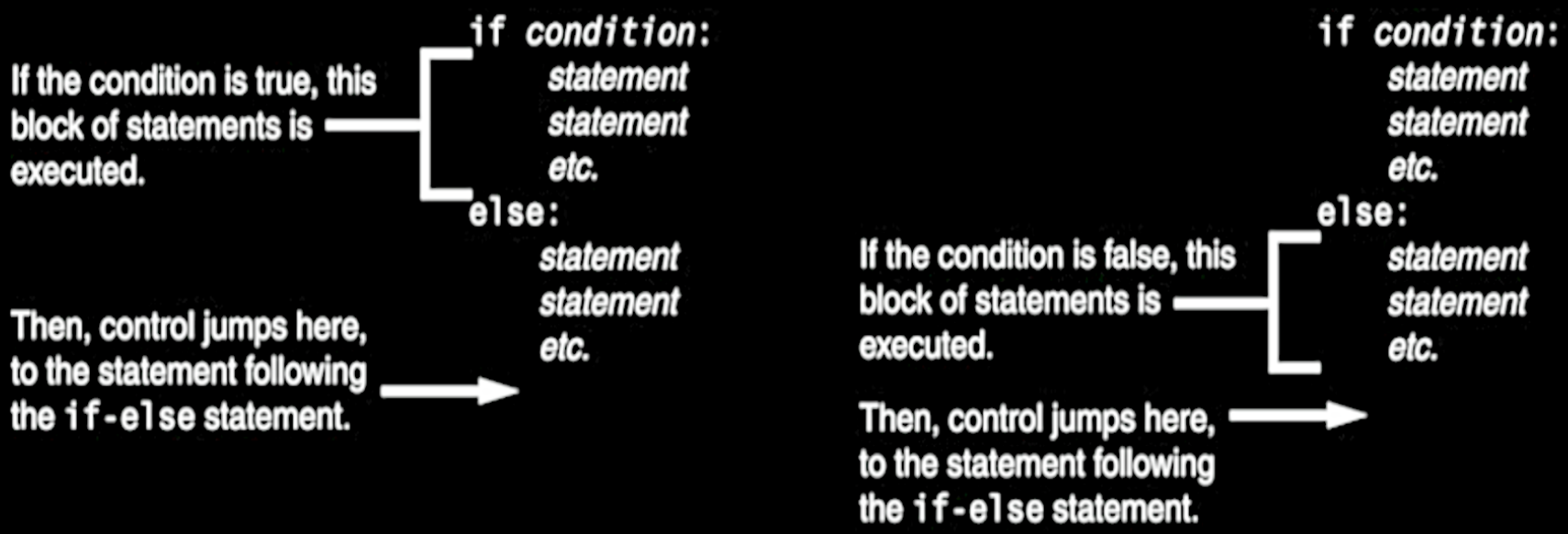
The if-else Statement (2 of 3)

Figure 3-5 A dual alternative decision structure:



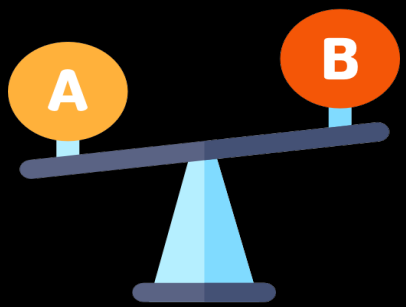
The if-else Statement (3 of 3)

Figure 3-6 Conditional execution in an if-else statement:



Comparing Strings (1 of 2)

- Strings can be compared using the $==$ and $!=$ operators
- String comparisons are case sensitive.
- Strings can be compared using $>$, $<$, $>=$, and $<=$ operators.
  - Compared character by character based on the ASCII values for each character.
  - If shorter word is substring of longer word, longer word is greater than shorter word.



Comparing Strings (2 of 2)

Figure 3-9 Comparing each character in a string:

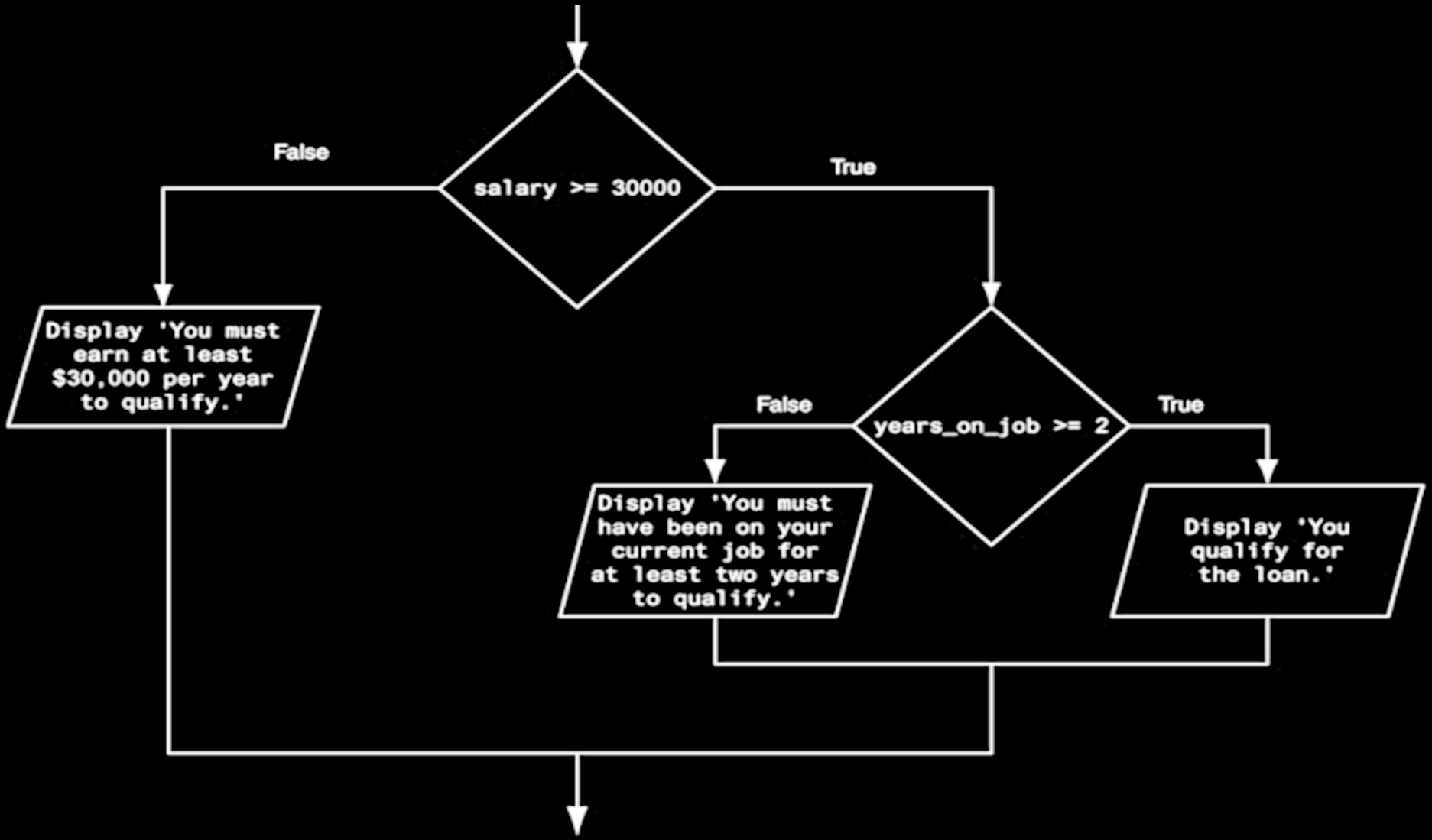
Nested Decision Structures and the if-elif-else Statement (1 of 3)

- A decision structure can be nested inside another decision structure and commonly needed in programs.

  Example:
  - Determine if someone qualifies for a loan, they must meet two conditions:
    - Must earn at least $30,000 per year.
    - Must have been employed for at least two years.
  - Check first condition, and if it is true, check second condition.

Nested Decision Structures and the if-elif-else Statement (2 of 3)

Figure 3-12 A nested decision structure



Nested Decision Structures and the if-elif-else Statement (3 of 3)

- Important to use proper indentation in a nested decision structure.
  - Important for Python interpreter.
  - Makes code more readable for programmer.
  - Rules for writing nested if statements:
    - else clause should align with matching if clause.
    - Statements in each block must be consistently indented.

The if-elif-else Statement (1 of 3)

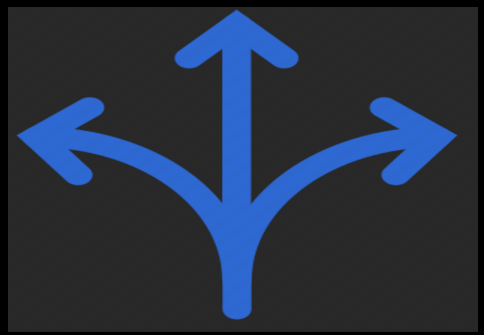
- if-elif-else statement: special version of a decision structure.
- Makes logic of nested decision structures simpler to write.
- Can include multiple elif statements.
- Syntax:

```
if condition_1:
    statement(s)
elif condition_2:
    statement(s)
elif condition_3:
    statement(s)
else
    statement(s)
```

Insert as many `elif` clauses as necessary.

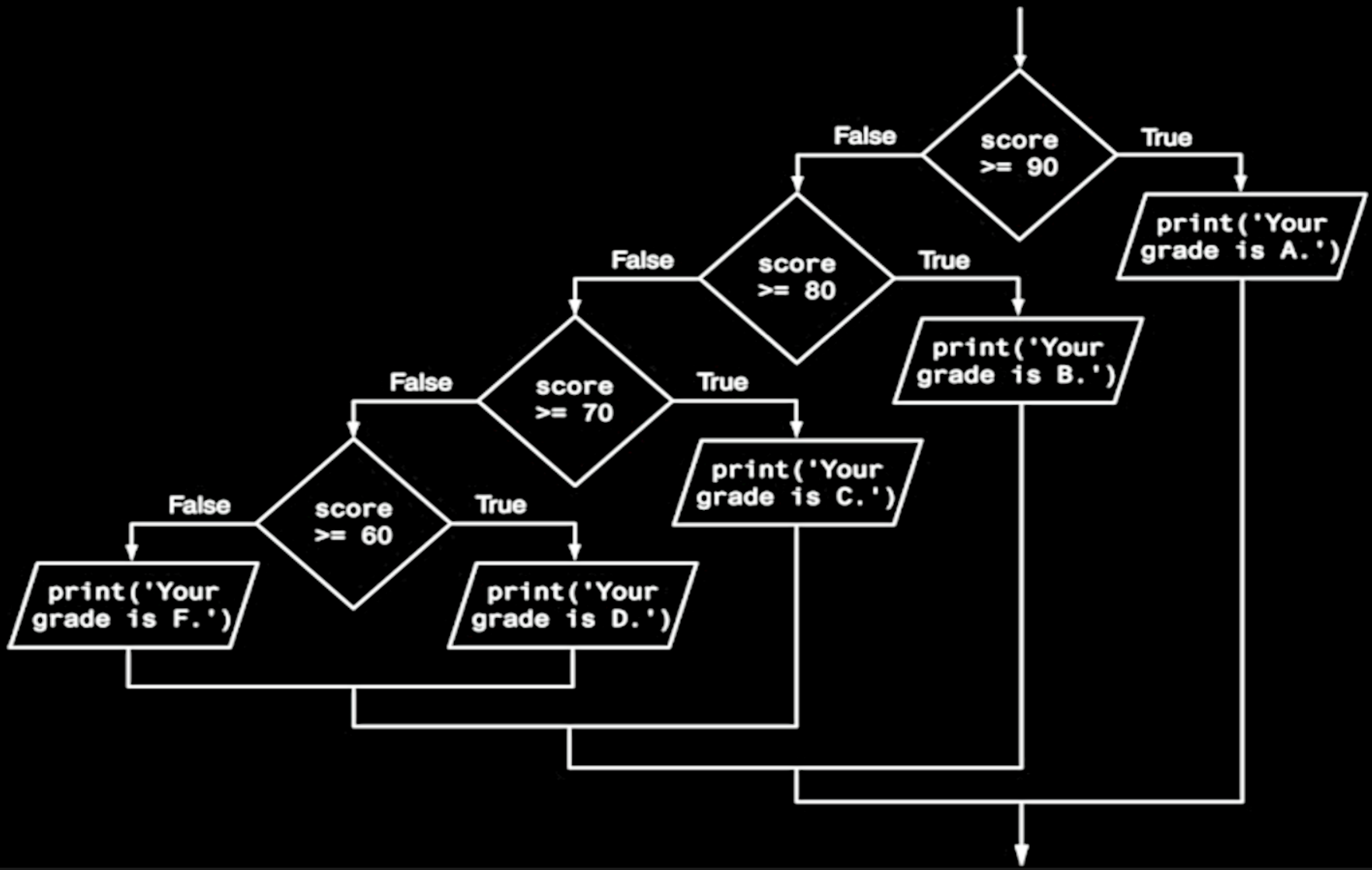
The if-elif-else Statement (2 of 3)

- Alignment used with if-elif-else statement:
  - if, elif, and else clauses are all aligned.
  - Conditionally executed blocks are consistently indented.
- if-elif-else statement is never required, but logic easier to follow:
  - Can be accomplished by nested if-else.
  - Code can become complex, and indentation can cause problematic long lines



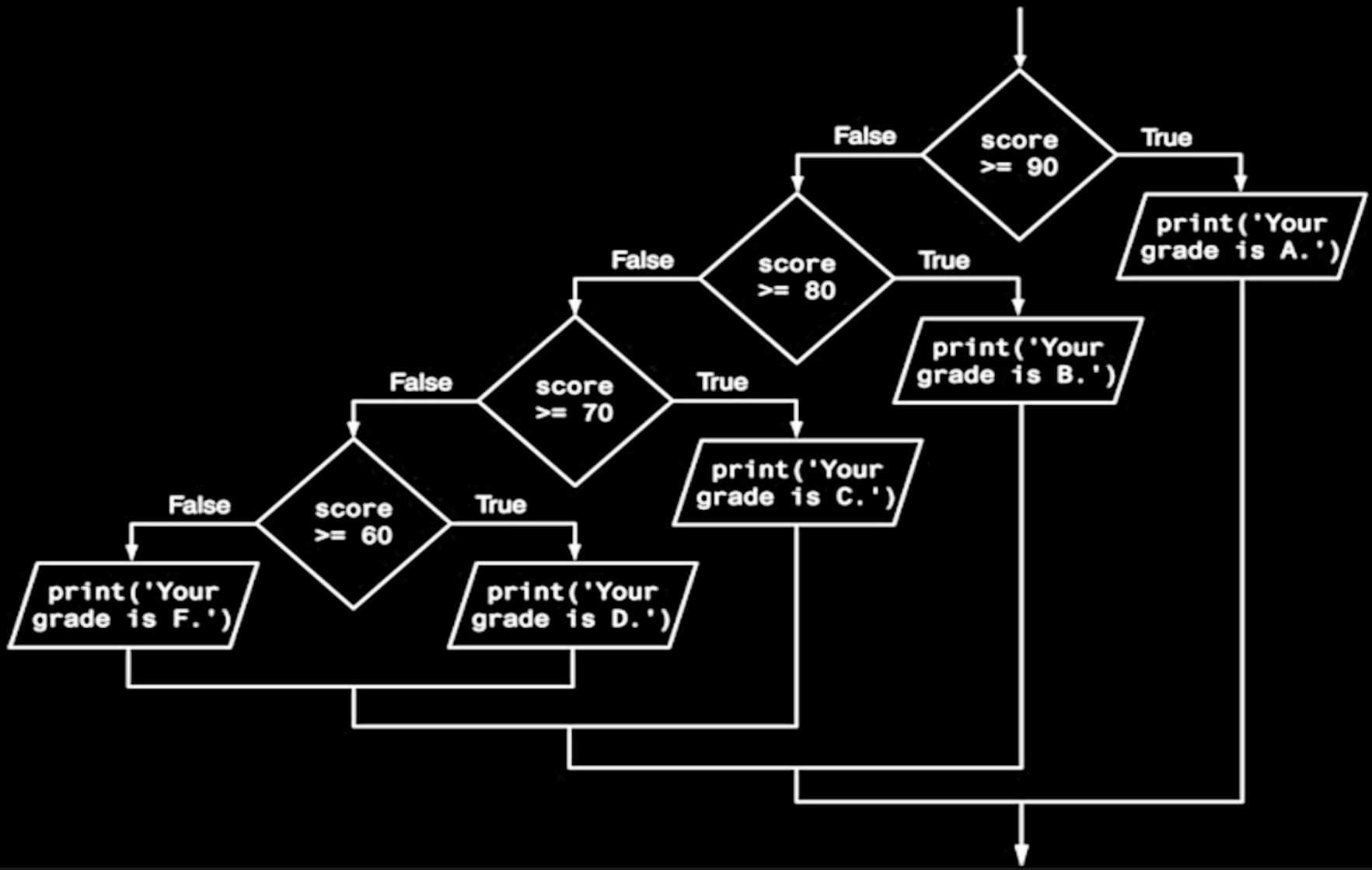
The if-elif-else Statement (3 of 3)

Figure 3-15 Nested decision structure to determine a grade:



The if-elif-else Statement (3 of 3)

Figure 3-15 Nested decision structure to determine a grade:



Logical Operators

- Logical operators: operators that can be used to create complex Boolean expressions.
- **and** operator & **or** operator: binary operators, connect two Boolean expressions into a compound Boolean expression.
- **not** operator: unary operator, reverses the truth of its Boolean operand.

The and Operator

- Takes two Boolean expressions as operands.
- Creates compound Boolean expression that is true only when both sub expressions are true.
- Can be used to simplify nested decision structures.
- Truth table for the and operator:

I_1	I_2	O
false	false	false
false	true	false
true	false	false
true	true	true

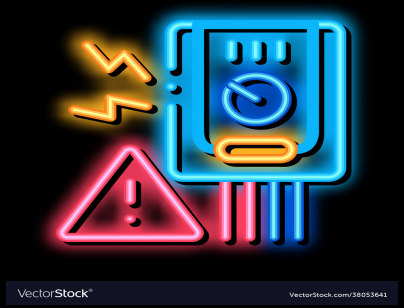
The or Operator

- Takes two Boolean expressions as operands.
- Creates compound Boolean expression that is true when either of the sub expressions is true.
- Can be used to simplify nested decision structures.
- Truth table for the or operator:

I_1	I_2	O
false	false	false
false	true	true
true	false	true
true	true	true

Short-Circuit Evaluation

- **Short circuit evaluation:** deciding the value of a compound Boolean expression after evaluating only one sub expression.
- Performed by the **or** and **and** operators.
  - For **or** operator: If left operand is true, compound expression is true. Otherwise, evaluate right operand.
  - For **and** operator: If left operand is false, compound expression is false. Otherwise, evaluate right operand.



The not Operator

- Takes one Boolean expressions as operand and reverses its logical value.
- Sometimes it may be necessary to place parentheses around an expression to clarify to what you are applying the not operator.
- Truth table for the not operator:

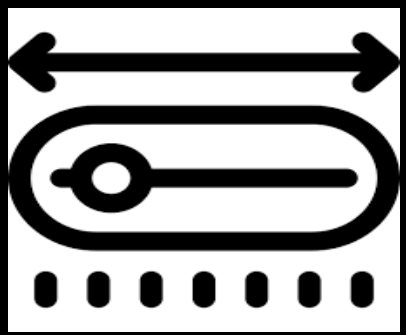
I	O
false	true
true	false

Checking Numeric Ranges with Logical Operators

- To determine whether a numeric value is **within** a specific range of values, use and

  Example: x >= 10 and x <= 20
- To determine whether a numeric value is **outside** of a specific range of values, use or

  Example: x < 10 or x > 20



Boolean Variables

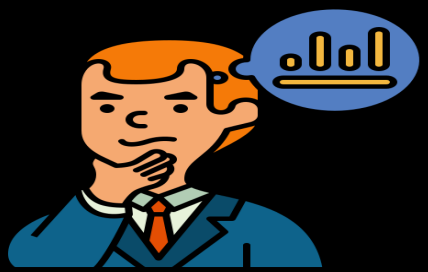
- **Boolean variable:** references one of two values, True or False.
- Represented by bool data type.
- Commonly used as flags.

  Flag: variable that signals when some condition exists in a program.
- Flag set to False $\rightarrow$ condition does not exist.
- Flag set to True $\rightarrow$ condition exists.

Lesson Summary

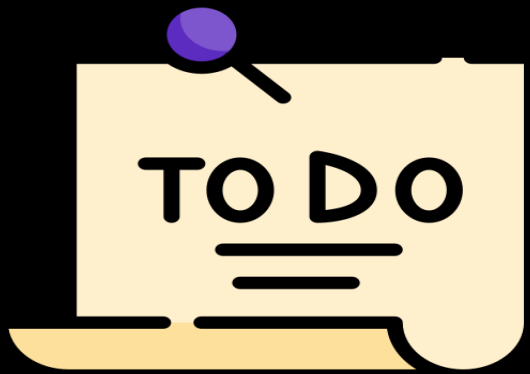
This chapter covered:

- Decision structures, including:
  - Single alternative decision structures.
  - Dual alternative decision structures.
  - Nested decision structures.
- Relational operators and logical operators as used in creating Boolean expressions:
  - String comparison as used in creating Boolean expressions.
  - Boolean variables.
  - Determining the state of the turtle in Turtle Graphics.



Things to do

- Complete Activity for Week-5.
- Read Textbook Chapter-3.



Questions?

Please ask your Questions to clarify!