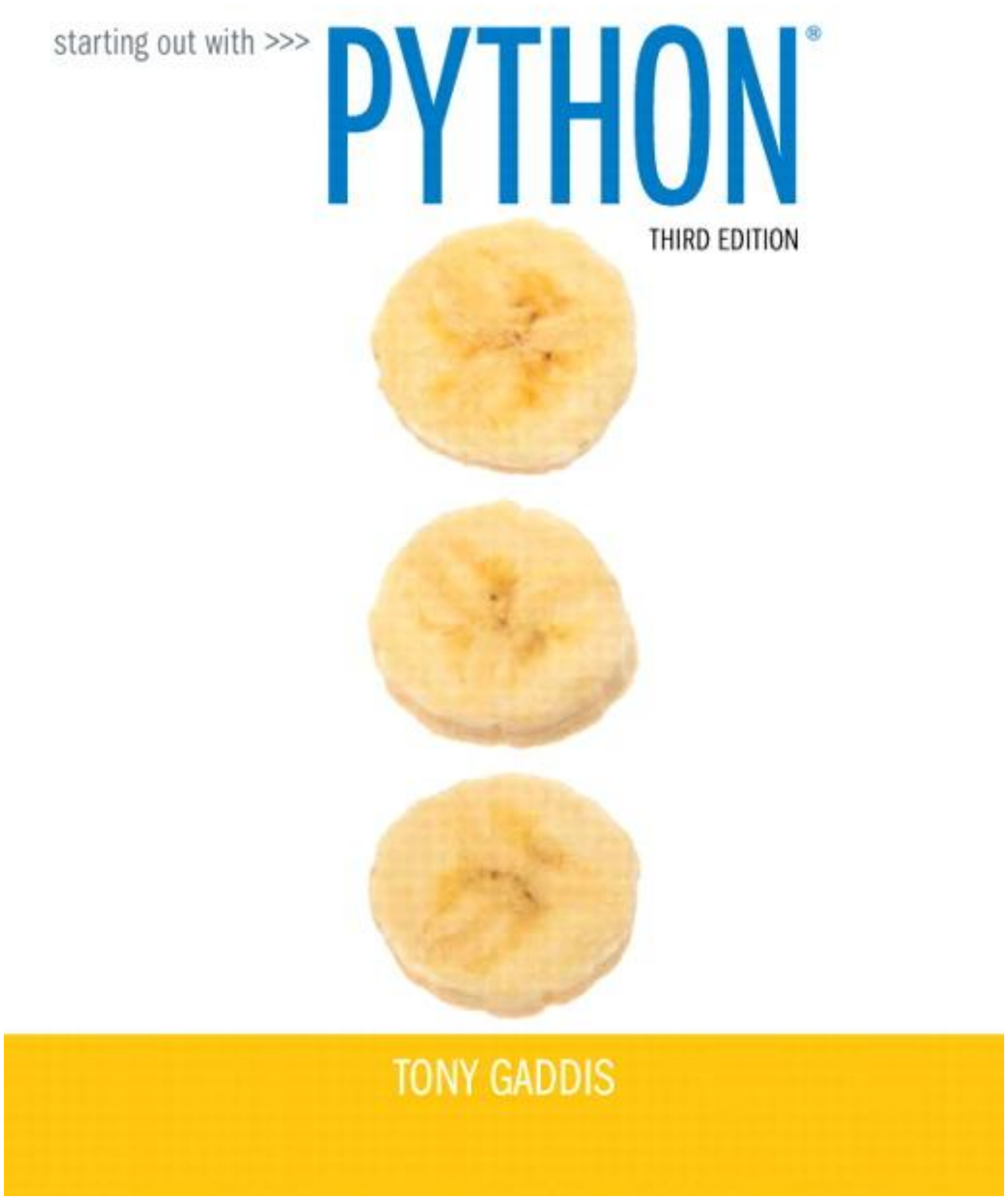
CHAPTER 3

Decision Structures and Boolean Logic



Topics

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

Scope of Variables

- **Local variable: variable that is assigned a value inside a function**
- **Scope known as lifespan of a variable refers to a code block within which a variable exists and can be used or referenced**
  - Variable only exists within the block it was declared
  - Outside of that block a variable does not exist and cannot be used
    - function block

Scope of Variables

```
def anotherFunction():
   someNum = 7  # only exists inside function
   thisNum = 3  # only exists inside function

def someFunction():
   someNum = 0  # have the same name but not same memory location
   print(someNume) # prints 0
   print(thisNum)  # ERROR - variable not defined

def main():
   someFunction()
   anotherFunction()

   print(someNume) # ERROR - variable not defined
   print(thisNum)  # ERROR - variable not defined

main()
```

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` → condition does not exist
    - Flag set to `True` → condition exists

Boolean Algebra

- **Boolean algebra: operations on Boolean variables, `True` or `False`, in 3 combinations AND, OR, NOT**
- **AND**
  - statement1 and statement2 must be true to continue
- **OR**
  - statement1 or statement2 must be true to continue
- **NOT**
  - Negate the statement
    - if statement is true - becomes false
    - if statement is false - becomes true

Boolean Algebra

	AND	
Statement 1	Statement 2	Result
True	True	True
True	False	False
False	True	False
False	False	False

	OR	
Statement 1	Statement 2	Result
True	True	True
True	False	True
False	True	True
False	False	False

NOT	
Statement 1	Result
True	False
False	True

Boolean Algebra

False OR False =

True AND True =

False AND True =

NOT False =

True OR True =

True AND False =

NOT True =

True OR False =

False AND False =

False OR True =

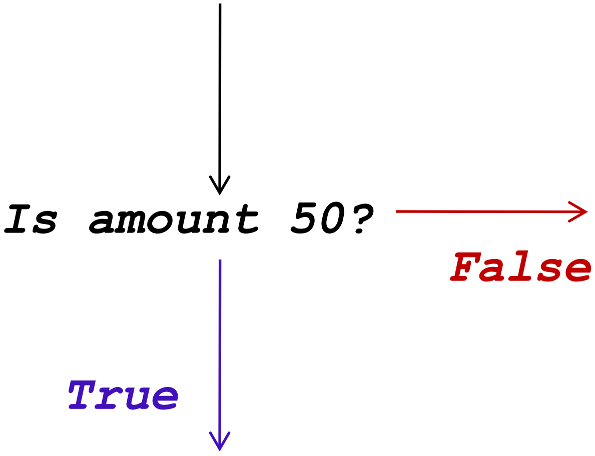
The `if` Statement

- **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
  - Allow a computer to make choices based on a condition

The `if` Statement (cont'd.)

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

The `if` Statement (cont'd.)

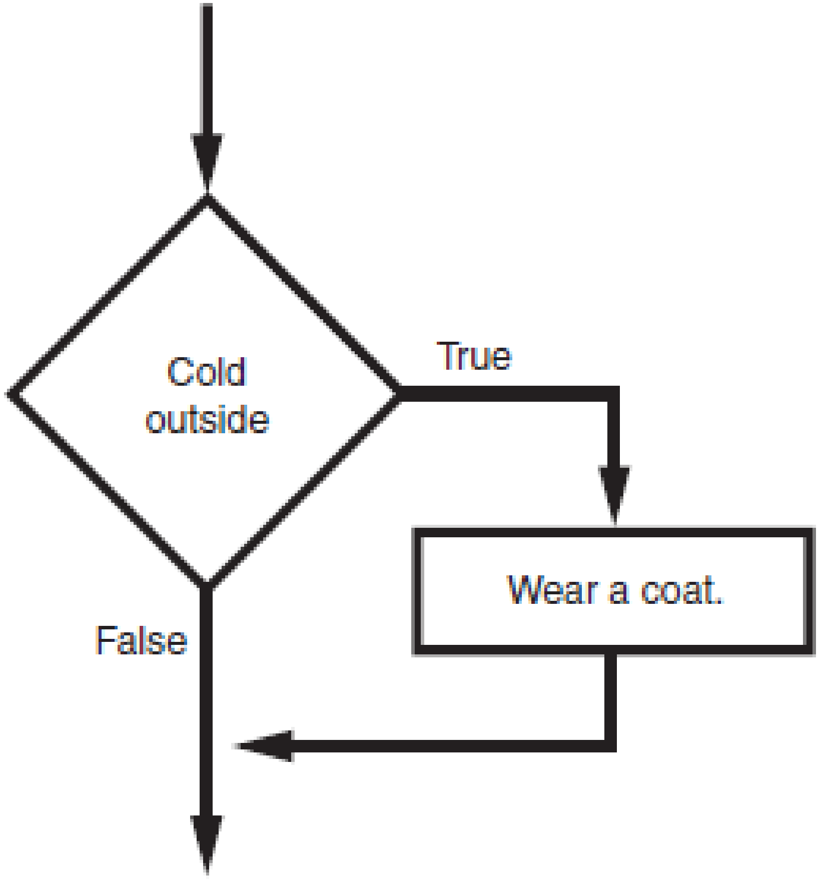


Is amount 50?

False

True

Figure 4-1 A simple decision structure



The `if` Statement (cont’d.)

- **Python syntax:**

```
if condition:
    statement
    statement
```

```
if amount == 50:
    statement
    statement
```

- **First line know as the `if` clause**
  - 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

- **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

- **>= 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

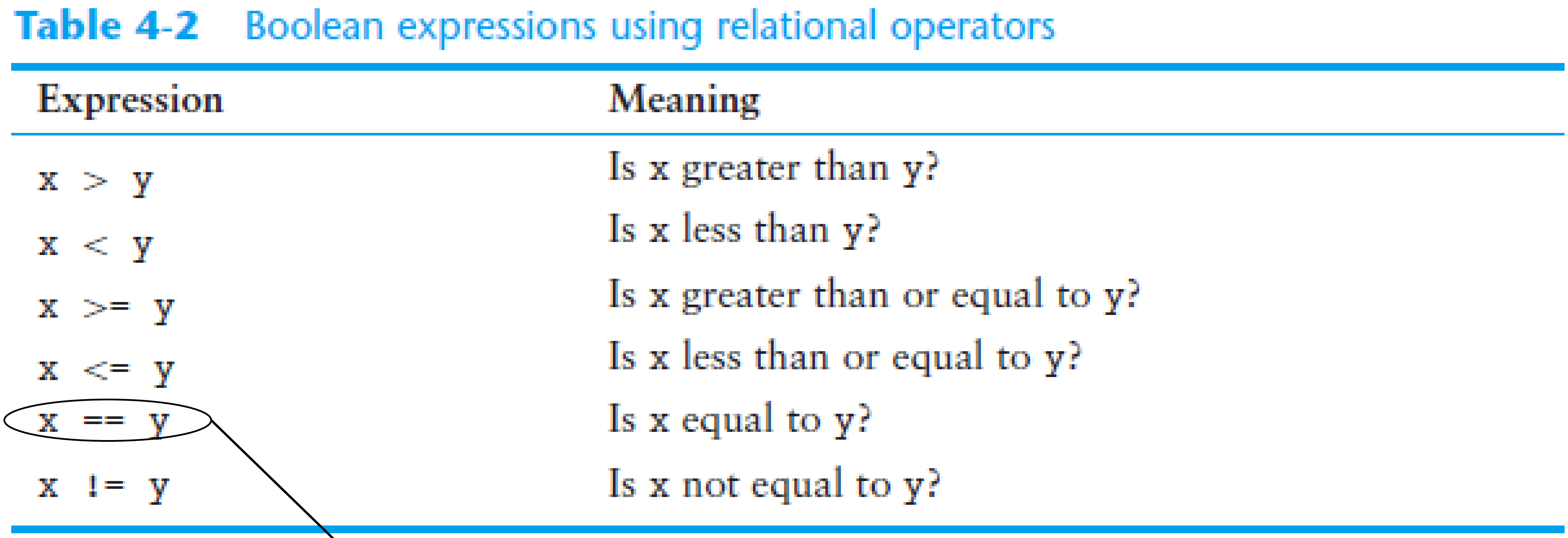


Table 4-2 Boolean expressions using relational operators

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?

Do not confuse with assignment operator (=)

Boolean Expressions and Relational Operators

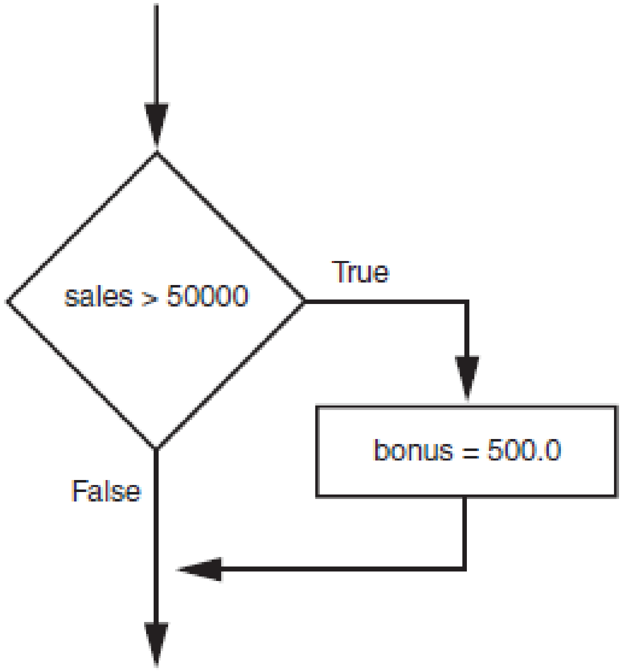
- **Using a Boolean expression with the > relational operator**

Figure 4-3 Example decision structure

```
…
bonus = 100.0

if sales > 50000:
        bonus = 500.0

wage = pay + bonus
…
```



Boolean Expressions and Relational Operators (cont’d.)

- **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

if_Wage.py

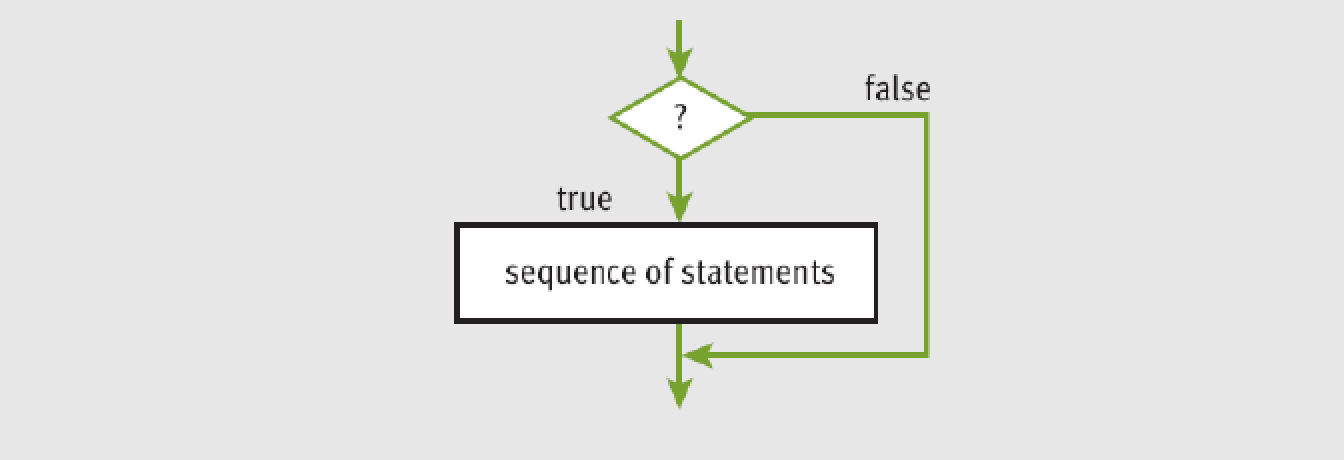
The `if-else` Statement

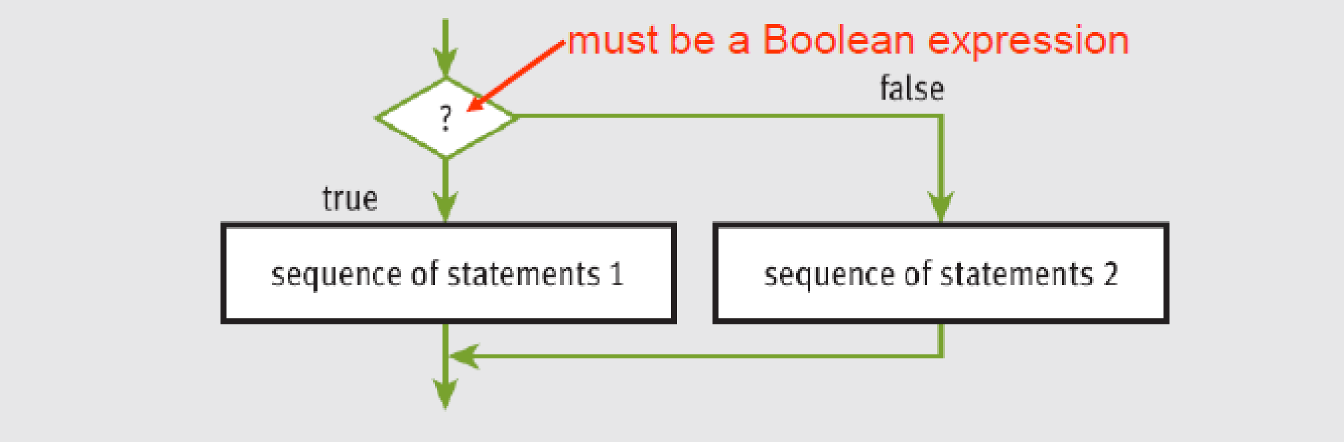
- **Dual alternative decision structure: two possible paths of execution**
  - One is taken if the condition is true, and the other if the condition is false

```
if condition:                 if sale > 50000:
   statements                      bonus = 500
else:                         else:
   other statements                bonus = 200
```

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

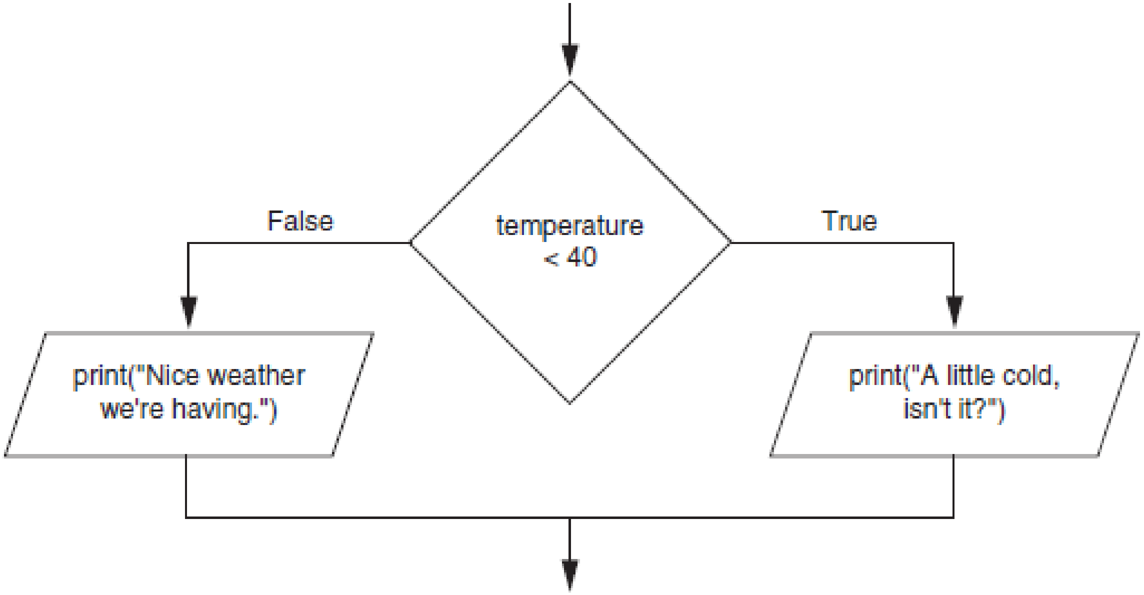
The `if-else` Statement





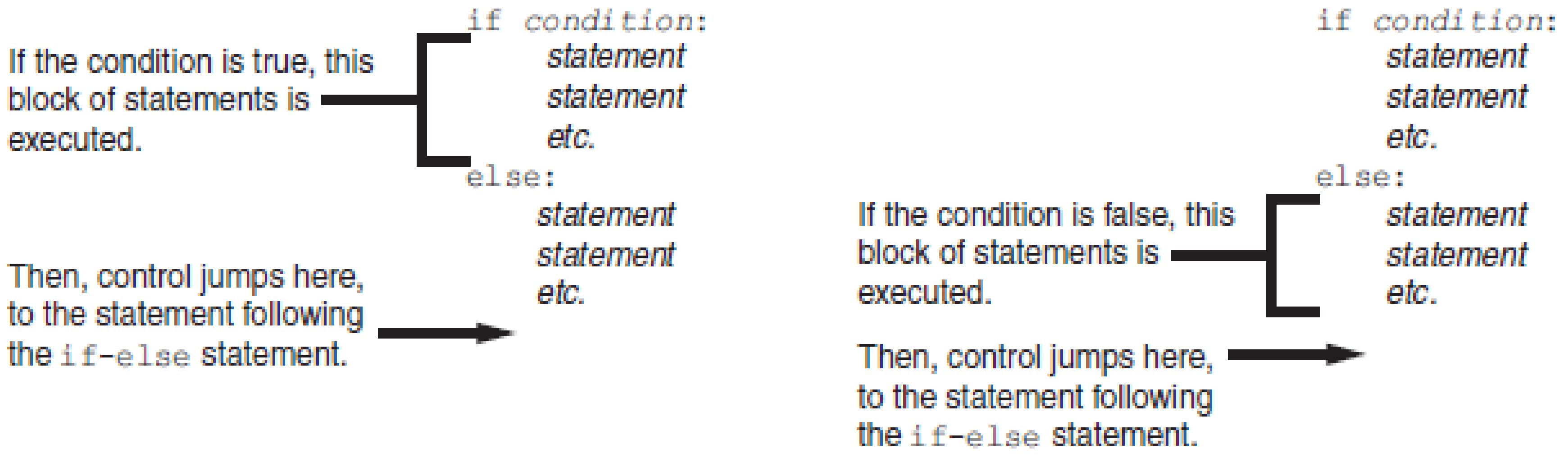
The `if-else` Statement

Figure 3-5 A dual alternative decision structure



The `if-else` Statement

Figure 4-7 Conditional execution in an `if-else` statement



```
if condition:
    statement
    statement
    etc.
else:
    statement
    statement
    etc.
```

If the condition is true, this block of statements is executed.

Then, control jumps here, to the statement following the `if-else` statement.

```
if condition:
    statement
    statement
    etc.
else:
    statement
    statement
    etc.
```

If the condition is false, this block of statements is executed.

Then, control jumps here, to the statement following the `if-else` statement.

```
if TRUE:
    do this
else:
    do that
```

ifElse_Score.py

Nested Decision Structures and the `if-elif-else` Statement

- **A decision structure can be nested inside another decision structure**
  - Commonly needed in programs
  - Example:
    - Determine if someone qualifies for a <u>super bonus</u>, they must meet two conditions:
      - Must have sold at least $50,000
      - Must have been employed for at least two years
    - Check first condition, and if it is true, check second condition

The `if-elif-else` Statement

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

```
if condition1                  if sale > 50000:
     statements                     bonus = 500
elif condition2                elif years > 2:
     statements                     bonus = 100
else                           else:
     statements                     bonus = 0
```

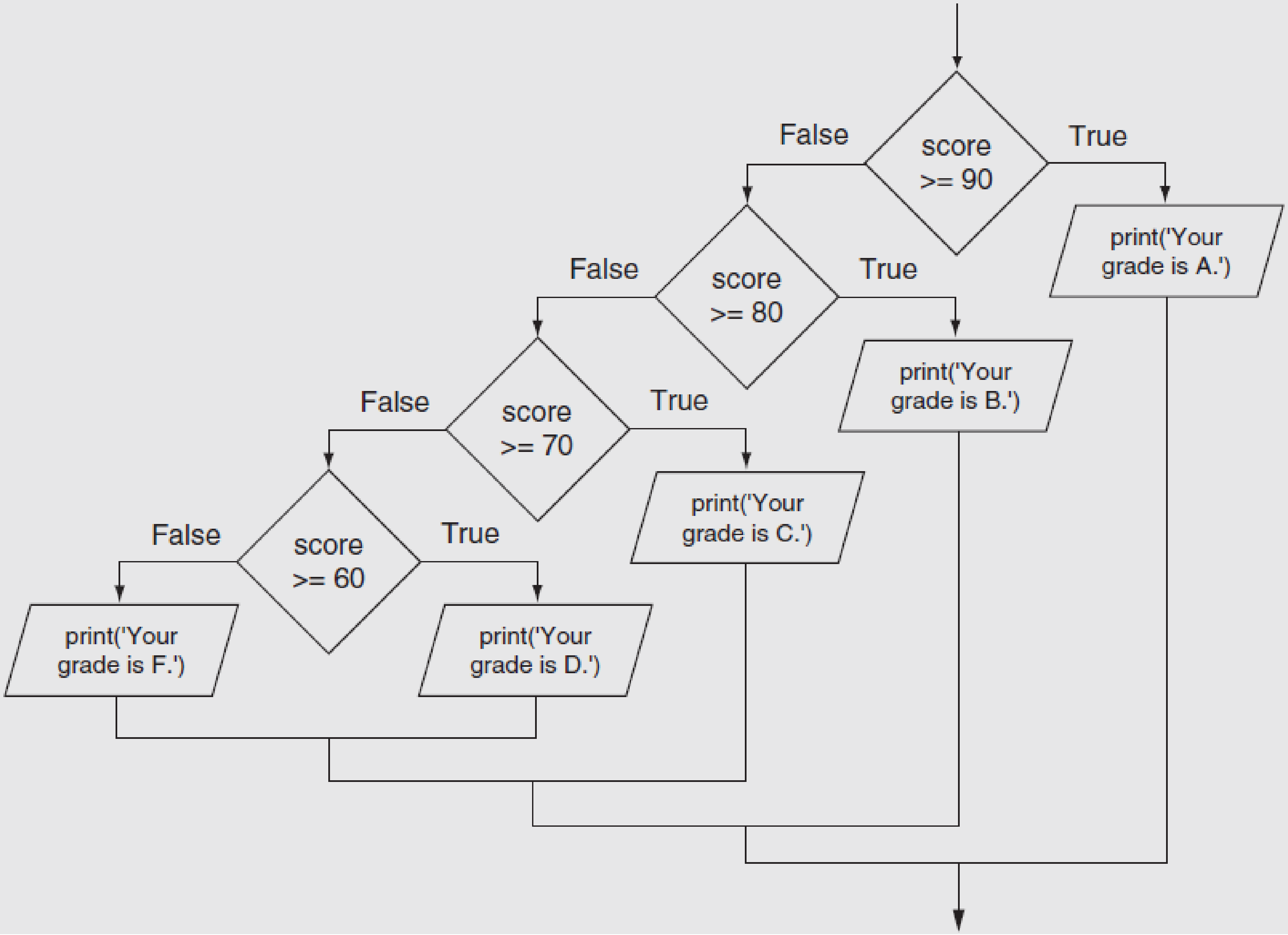
The `if-elif-else` Statement

- **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

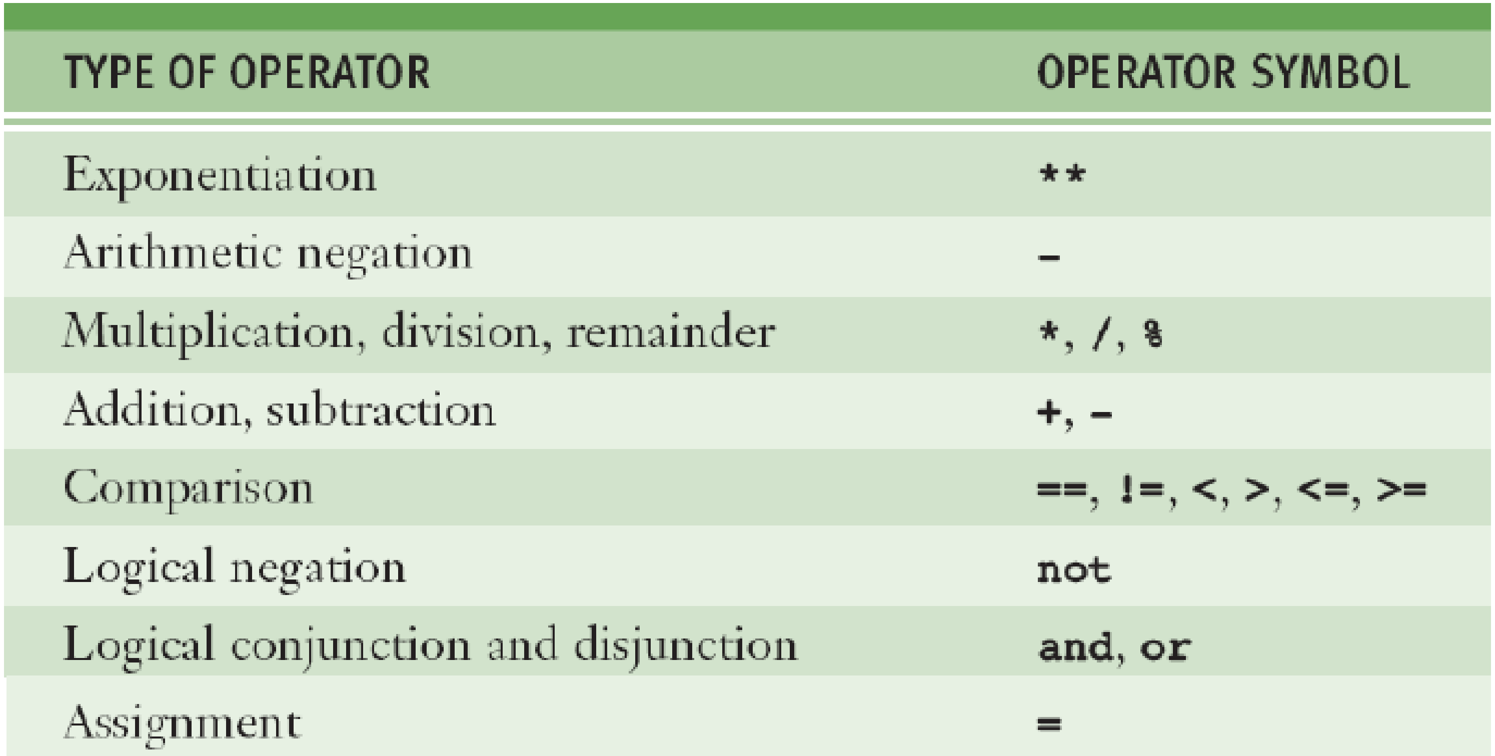
ifElifElse_Grade.py

ifElifElse_Wage.py

Figure 3-15 Nested decision structure to determine a grade


score >= 90
False
True
print('Your grade is A.')
score >= 80
False
True
print('Your grade is B.')
score >= 70
False
True
print('Your grade is C.')
score >= 60
False
True
print('Your grade is F.')
print('Your grade is D.')

Logical Operators and Compound Boolean Expressions



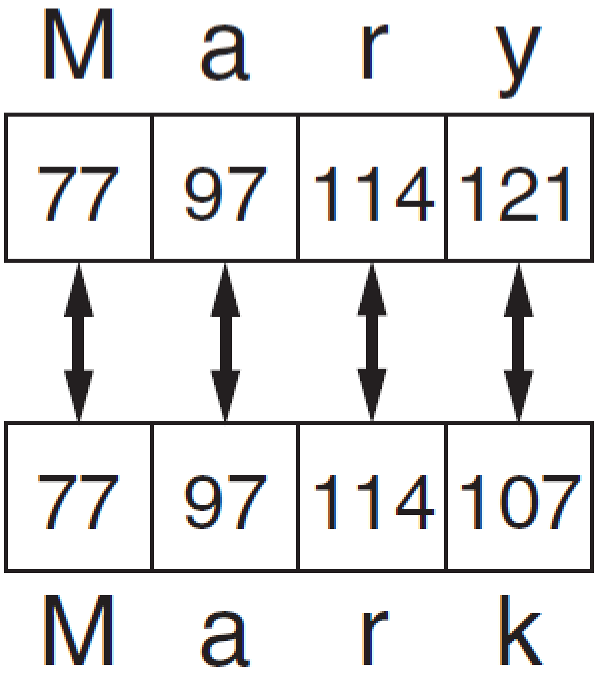
TYPE OF OPERATOR	OPERATOR SYMBOL
Exponentiation	`**`
Arithmetic negation	`-`
Multiplication, division, remainder	`*`, `/`, `%`
Addition, subtraction	`+`, `-`
Comparison	`==`, `!=`, `<`, `>`, `<=`, `>=`
Logical negation	`not`
Logical conjunction and disjunction	`and`, `or`
Assignment	`=`

Comparing Strings

- **Strings can be compared using the == and != operators**
- **String comparisons are case sensitive**
- **Strings can be compared using >, <, >=, and <=**
  - 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 (cont'd.)

Figure 3-9 Comparing each character in a string



M a r y

77 97 114 121

77 97 114 107

M a r k

ifElse_Password.py

Logical Operators

- **Logical operators: operators that can be used to create complex Boolean expressions**
  - `and` operator and `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**

Expression	Value of the Expression
false and false	false
false and true	false
true and false	false
true and 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**

Expression	Value of the Expression
false and false	false
false and true	true
true and false	true
true and true	true

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**

Expression	Value of the Expression
true	false
false	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

Checking Numeric Ranges with Logical Operators

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

Compound if

```
if unit != 'w' or 'd':
    print("Pass")
```

```
IS NOT:
if unit != 'w' or unit !=  'd':
    print("Pass")
```

Compound if

```
unit = int(input("Input: "))

if 0 <= unit <= 50:
    print("Pass")

USE:
if 10 <= unit or unit <= 50:
    print("Pass")
```

IF

```
if size >= 50:
     … code …
```

IF-ELSE

```
if size >= 50:
     … code …
else:
     … code …
```

IF- ELIF - ELSE

```
if size >= 50:
     … code …
elif size > 40:
     … code …
else:
     … code …
```

Every IF-ELIF-ELSE must end with an **else**

```
if number == 1:
        print("Roman numeral is: I")
elif number == 2:
        print("Roman numeral is: II")
elif number == 3:
        print("Roman numeral is: III")
elif number == 4:
        print("Roman numeral is: IV")
elif number == 5:
        print("Roman numeral is: V")
elif number == 6:
        print("Roman numeral is: VI")
elif number == 7:
        print("Roman numeral is: VII")
elif number == 8:
        print("Roman numeral is: VIII")
elif number == 9:
        print("Roman numeral is: IX")
elif number == 10:
        print("Roman numeral is: X")
else:
        print(“Number is out of range")
```

```
if number == 1:
        print("Roman numeral is: I")
if number == 2:
        print("Roman numeral is: II")
if number == 3:
        print("Roman numeral is: III")
if number == 4:
        print("Roman numeral is: IV")
if number == 5:
        print("Roman numeral is: V")
if number == 6:
        print("Roman numeral is: VI")
if number == 7:
        print("Roman numeral is: VII")
if number == 8:
        print("Roman numeral is: VIII")
if number == 9:
        print("Roman numeral is: IX")
if number == 10:
        print("Roman numeral is: X")
if number > 10 or number < 1:
        print(“Number is out of range")
```

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