**CHAPTER 3**

Decision Structures and Boolean Logic

---

**Topics**

- The if Statement
- The if-else Statement
- Comparing Strings
- Nested Decision Structures and the if-elif-else Statement
- Logical Operators
- Boolean Variables
- Turtle Graphics: Determining the State of the Turtle

---

**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

---

**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
- **Single alternative decision structure**: provides only one alternative path of execution
  - If condition is not true, exit the structure
The if Statement (cont’d.)

Python syntax:
```python
if condition:
    Statement
    Statement
```

First line known 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 (cont’d.)

- $\geq$ and $\leq$ 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 (cont’d.)

<table>
<thead>
<tr>
<th>Table 3-2</th>
<th>Boolean expressions using relational operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>Meaning</td>
</tr>
<tr>
<td>x &gt; y</td>
<td>Is x greater than y?</td>
</tr>
<tr>
<td>x &lt; y</td>
<td>Is x less than y?</td>
</tr>
<tr>
<td>x &gt;= y</td>
<td>Is x greater than or equal to y?</td>
</tr>
<tr>
<td>x &lt;= y</td>
<td>Is x less than or equal to y?</td>
</tr>
<tr>
<td>x == y</td>
<td>Is x equal to y?</td>
</tr>
<tr>
<td>x != y</td>
<td>Is x not equal to y?</td>
</tr>
</tbody>
</table>

• Using a Boolean expression with the > relational operator

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
  
  Syntax: if condition:
  statements
  else:
  other statements

• If clause and else clause must be aligned
• Statements must be consistently indented
The if-else Statement (cont’d.)

Figure 3-5  A dual alternative decision structure

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 3-6  Conditional execution in an if-else statement

Figure 3-9  Comparing each character in a string

<table>
<thead>
<tr>
<th>Mar y</th>
<th>77 97 114 121</th>
</tr>
</thead>
<tbody>
<tr>
<td>M a r k</td>
<td>77 97 114 107</td>
</tr>
</tbody>
</table>
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 loan, they must meet two conditions:
      - Must earn at least $30,000/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 (cont’d.)

- 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

- 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:
    ```python
    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 (cont’d.)

- 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

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

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value of the Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>false and false</td>
<td>false</td>
</tr>
<tr>
<td>false and true</td>
<td>false</td>
</tr>
<tr>
<td>true and false</td>
<td>false</td>
</tr>
<tr>
<td>true and true</td>
<td>true</td>
</tr>
</tbody>
</table>
The \textbf{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 \textbf{or} operator

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value of the Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>false and false</td>
<td>false</td>
</tr>
<tr>
<td>false and true</td>
<td>true</td>
</tr>
<tr>
<td>true and false</td>
<td>true</td>
</tr>
<tr>
<td>true and true</td>
<td>true</td>
</tr>
</tbody>
</table>

Short-Circuit Evaluation

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

The \textbf{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 \textbf{not} operator
- Truth table for the \textbf{not} operator

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value of the Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

Checking Numeric Ranges with Logical Operators

- To determine whether a numeric value is within a specific range of values, use \textbf{and}
  - Example: \( x \geq 10 \text{ and } x \leq 20 \)
- To determine whether a numeric value is outside of a specific range of values, use \textbf{or}
  - Example: \( x < 10 \text{ or } x > 20 \)
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` → condition does not exist
    - Flag set to `True` → condition exists

---

Turtle Graphics: Determining the State of the Turtle

- **The `turtle.xcor()` and `turtle.ycor()` functions** return the turtle's X and Y coordinates
- **Examples of calling these functions in an `if` statement**:

  ```python
  if turtle.xcor() > 100 and turtle.xcor() < 200:
      turtle.goto(0, 0)
  if turtle.ycor() < 0:
      turtle.goto(0, 0)
  ```

---

Turtle Graphics: Determining the State of the Turtle

- **The `turtle.heading()` function returns** the turtle's heading. (By default, the heading is returned in degrees.)
- **Example of calling the function in an `if` statement**:

  ```python
  if turtle.heading() >= 90 and turtle.heading() <= 270:
      turtle.setheading(180)
  ```

---

Turtle Graphics: Determining the State of the Turtle

- **The `turtle.isdown()` function returns** `True` if the pen is down, or `False` otherwise.
- **Example of calling the function in an `if` statement**:

  ```python
  if turtle.isdown():
      turtle.penup()
  if not(turtle.isdown()):
      turtle.pendown()
  ```
Turtle Graphics: Determining the State of the Turtle

- The `turtle.isvisible()` function returns `True` if the turtle is visible, or `False` otherwise.
- Example of calling the function in an `if` statement:
  ```python
  if turtle.isvisible():
    turtle.hideturtle()
  ```

- When you call `turtle.pencolor()` without passing an argument, the function returns the pen's current color as a string. Example of calling the function in an `if` statement:
  ```python
  if turtle.pencolor() == 'red':
    turtle.pencolor('blue')
  ```

- When you call `turtle.fillcolor()` without passing an argument, the function returns the current fill color as a string. Example of calling the function in an `if` statement:
  ```python
  if turtle.fillcolor() == 'blue':
    turtle.fillcolor('white')
  ```

- When you call `turtle.bgcolor()` without passing an argument, the function returns the current background color as a string. Example of calling the function in an `if` statement:
  ```python
  if turtle.bgcolor() == 'white':
    turtle.bgcolor('gray')
  ```

- When you call `turtle.pensize()` without passing an argument, the function returns the pen's current size as a string. Example of calling the function in an `if` statement:
  ```python
  if turtle.pensize() < 3:
    turtle.pensize(3)
  ```
Turtle Graphics: Determining the State of the Turtle

- When you call `turtle.speed()` without passing an argument, the function returns the current animation speed. Example of calling the function in an `if` statement:

```python
if turtle.speed() > 0:
    turtle.speed(0)
```

Turtle Graphics: Determining the State of the Turtle

- See *In the Spotlight: The Hit the Target Game* in your textbook for numerous examples of determining the state of the turtle.

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