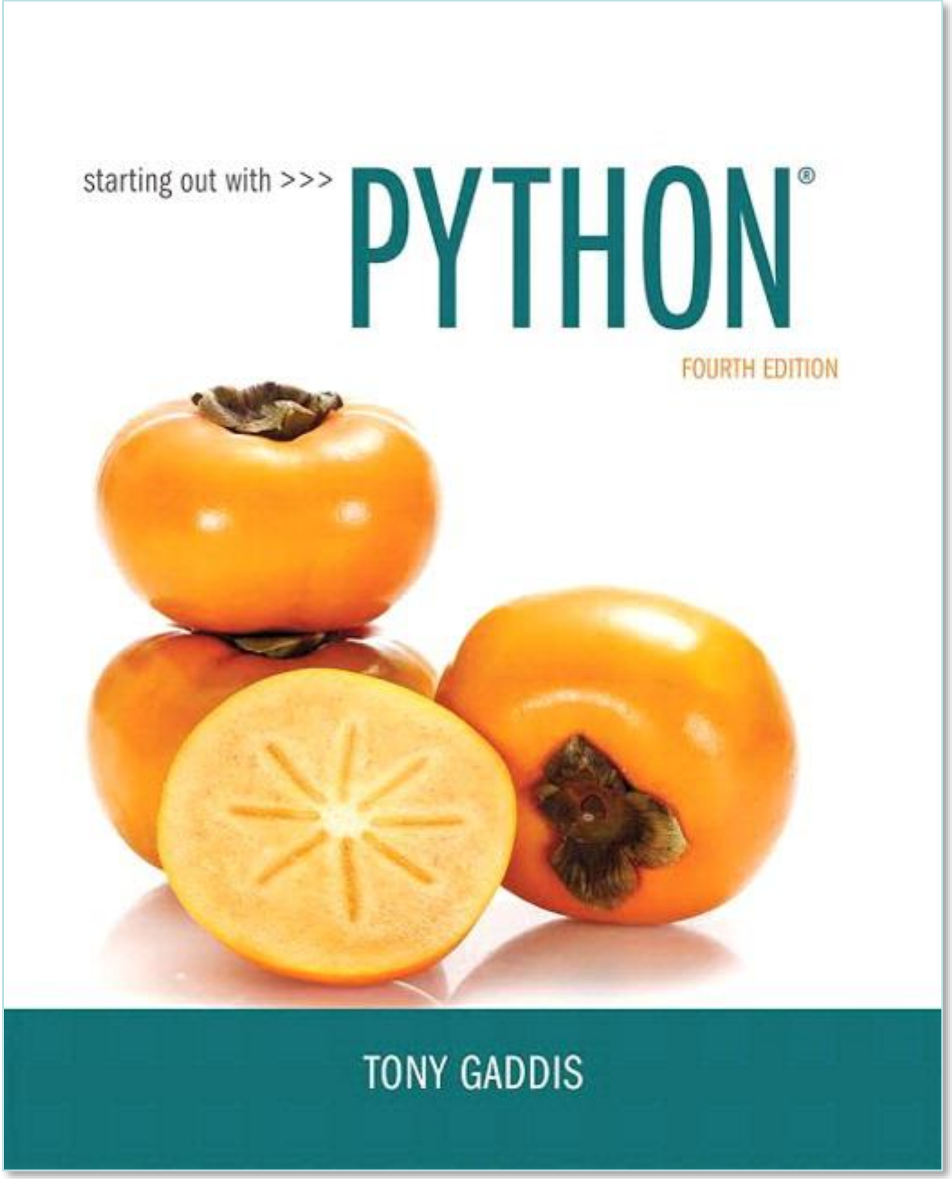
CHAPTER 3

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

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

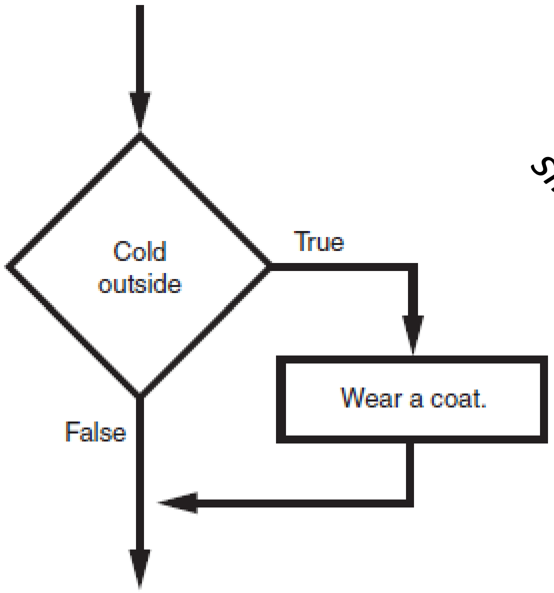
The `if` Statement

- 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

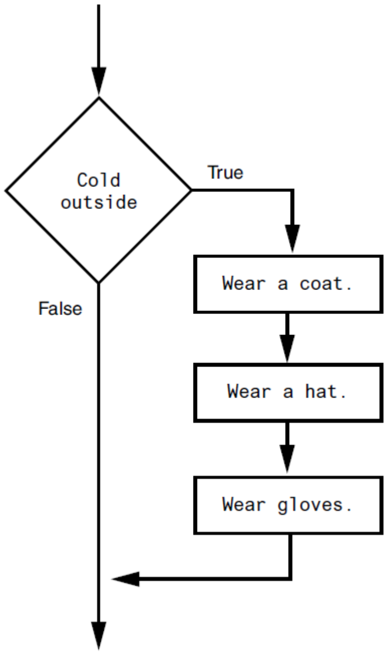
The `if` Statement (cont’d.)

- **In flowchart, diamond represents true/false condition that must be tested**
- **Actions can be *conditionally executed***
  - Performed only when a condition is true
- **<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.)



single alternative



single alternative

The `if` Statement (cont'd.)

- **Python syntax:**

```
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.)

- **>= 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 (cont’d.)

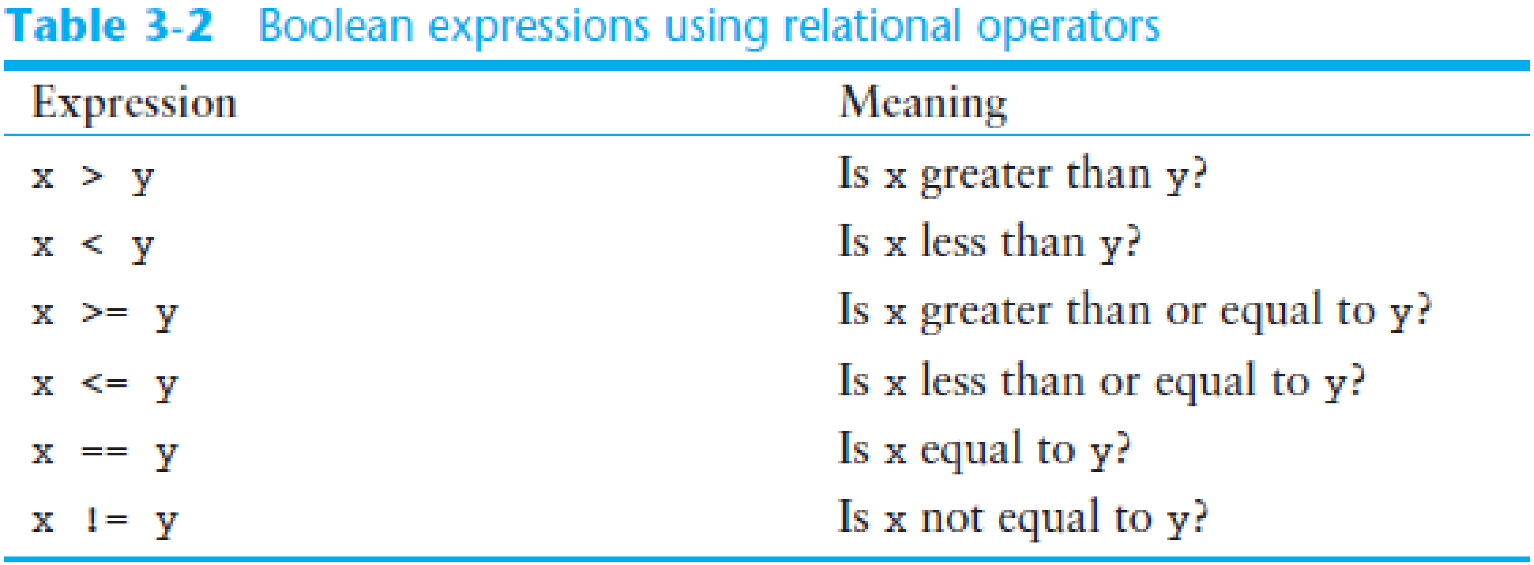


Table 3-2 Boolean expressions using relational operators

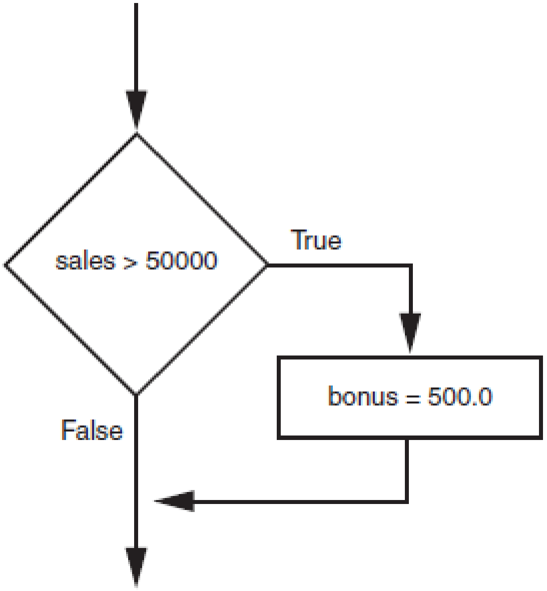
Expression	Meaning
`x > y`	Is x greater than y?
`x < y`	Is x less than y?
`x >= y`	Is x greater than or equal to y?
`x <= y`	Is x less than or equal to y?
`x == y`	Is x equal to y?
`x != y`	Is x not equal to y?

Boolean Expressions and Relational Operators (cont’d.)

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

```
>>> if sales > 50000:
>>>     bonus = 500.0
>>>
```

Figure 3-3 Example decision structure



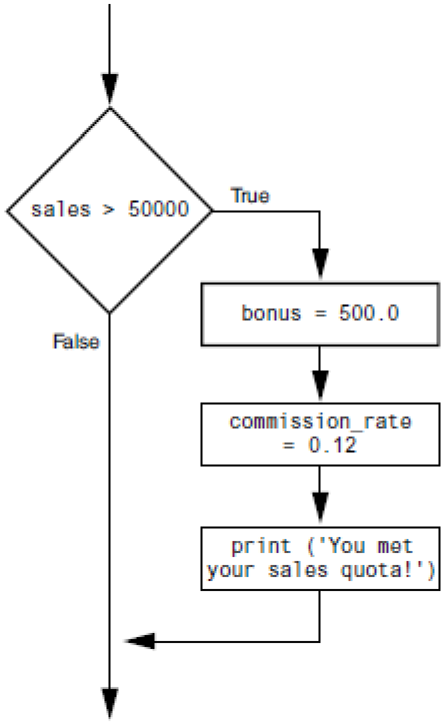
```
>>> x = 1
>>> y = 0 ↵
>> x > y
True
>>> x < y ↵
False
```

Boolean Expressions and Relational Operators (cont'd.)

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

```
>>> if sales > 50000:
>>>     bonus = 500.0
>>>     commission_rate = 0.12
>>>     print('You met your sales quota!')
```

Figure 3-4 Example decision structure



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

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-else` Statement

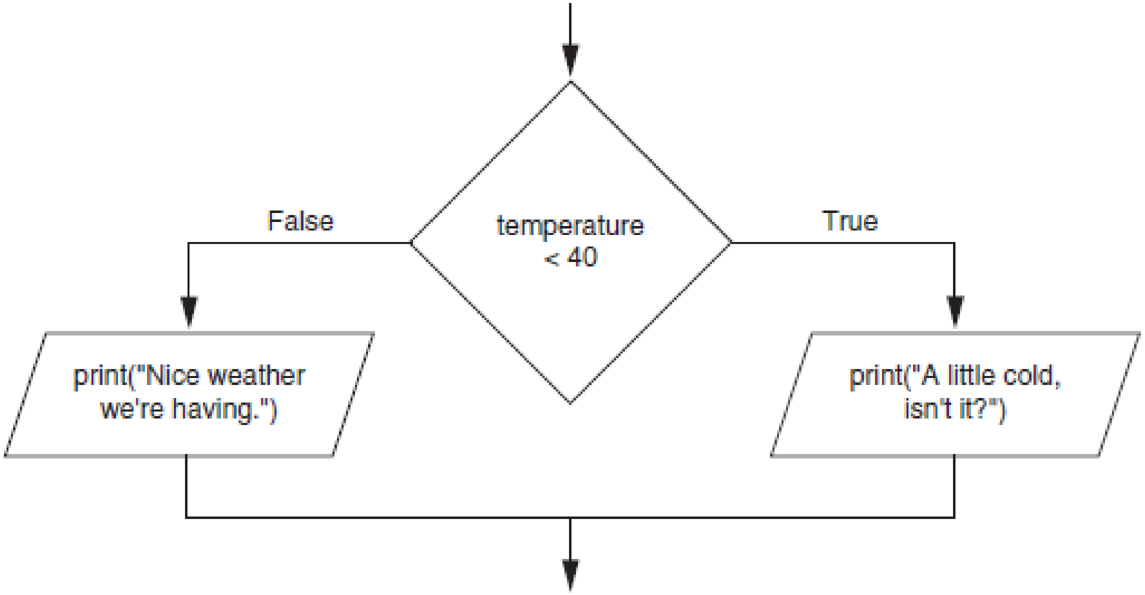
- **Dual alternative decision structure: two possible paths of execution**
  - One is taken if the condition is true, and the other if the condition is false
  - Syntax:

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

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

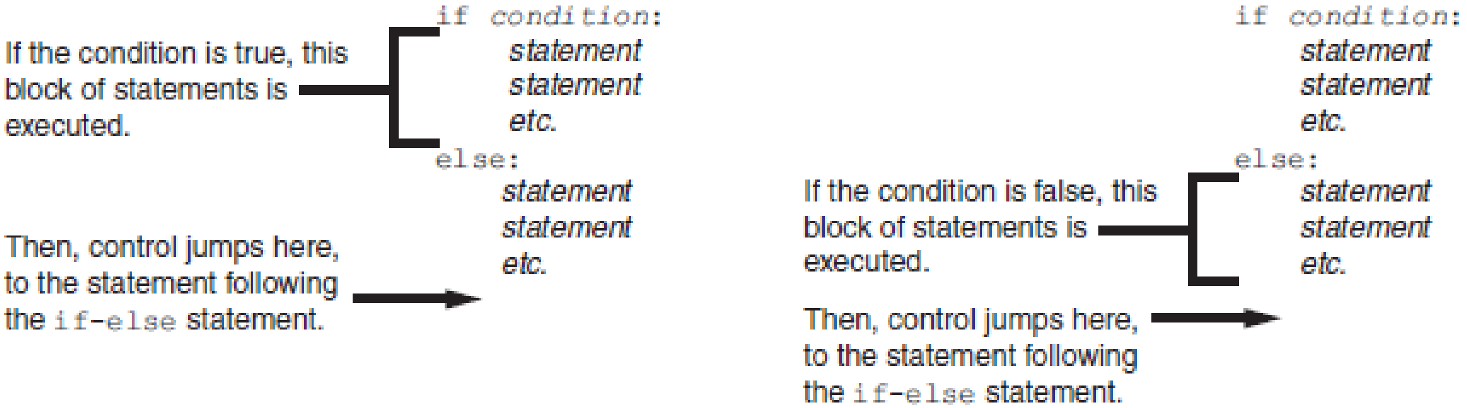
The `if-else` Statement (cont'd.)

Figure 3-5 A dual alternative decision structure


False
temperature < 40
True
print("Nice weather we're having.")
print("A little cold, isn't it?")

The `if-else` Statement (cont'd.)

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



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

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

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

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

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

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

Topics

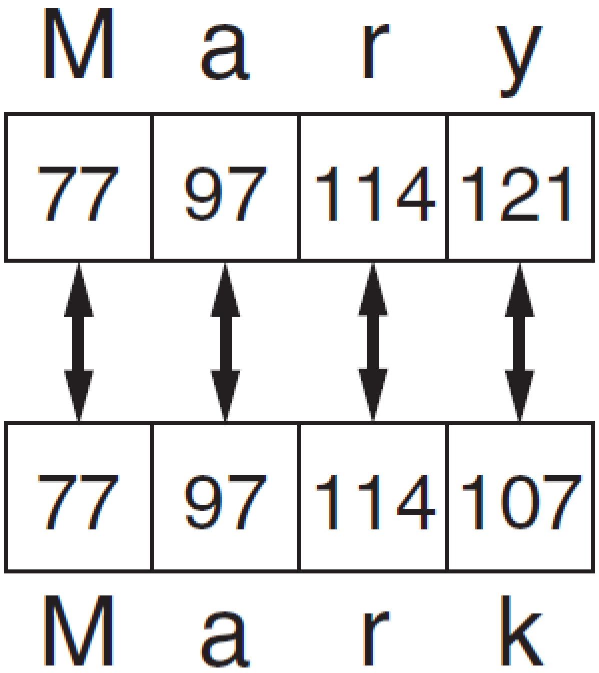
- The if Statement
- The if-else Statement
- **Comparing Strings**
- Nested Decision Structures and the `if-elif-else` Statement
- Logical Operators
- Boolean Variables
- Turtle Graphics: Determining the State of the Turtle

Comparing Strings

- **Strings can be compared using the == and != operators**
- **String comparisons are case sensitive**
- **Strings can also 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.)

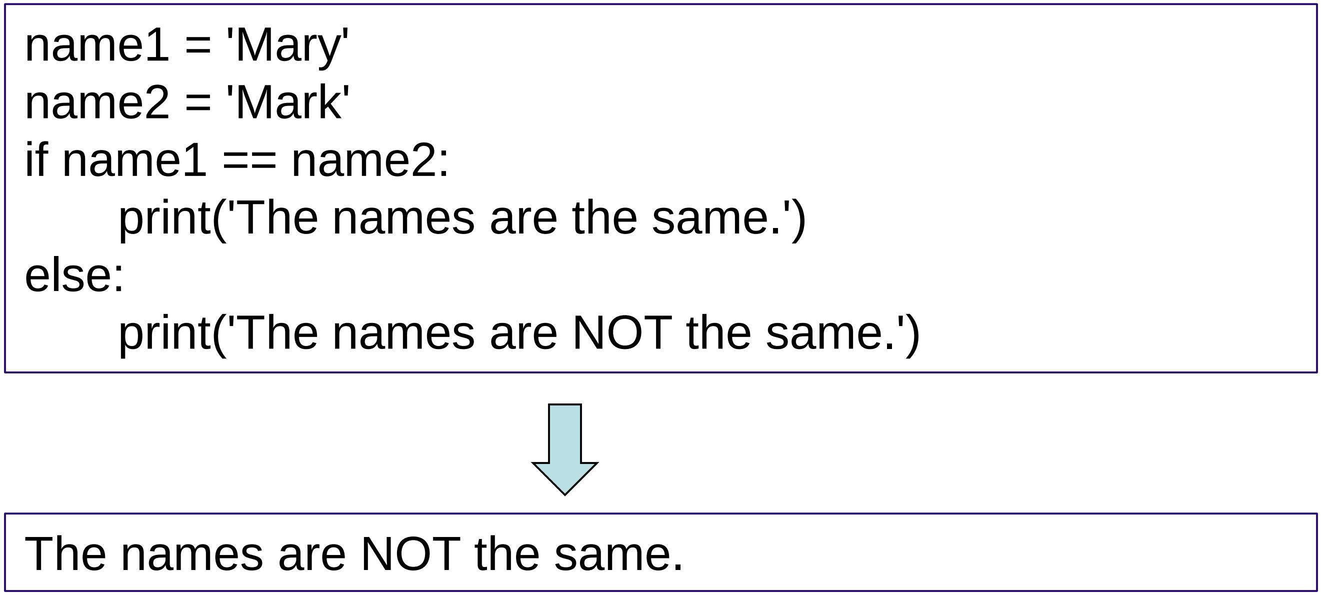
Figure 3-9 Comparing each character in a string



M	a	r	y
77	97	114	121
77	97	114	107
M	a	r	k

ASCII values

Comparing Strings (cont’d.)



```
name1 = 'Mary'
name2 = 'Mark'
if name1 == name2:
        print('The names are the same.')
else:
        print('The names are NOT the same.')
```

```
The names are NOT the same.
```

Comparing Strings (cont’d.)

```
# This program compares two strings.
# Get a password from the user.
password = input('Enter the password: ')

# Determine whether the correct password
# was entered.
if password == 'prospero':
    print('Password accepted.')
else:
    print('Sorry, that is the wrong password.')
```

password.py

```
Enter the password: Prospero ↲
```

```
Sorry, that is the wrong password
```

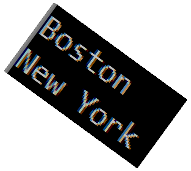
Quiz

- What would the following code display?

```
if 'z' < 'a’:
      print ('z is less than a.')
else:
      print ('z is not less than a.')
```

```
>>> ord('a’) ↵
97
>>> ord(‘z’) ↵
122
```

```
s1 = 'New York'
s2 = 'Boston’
if s1 > s2:
      print (s2)
      print (s1)
else:
      print (s1)
      print (s2)
```



```
Boston
New York
```

Convert Characters to ASCII Codes

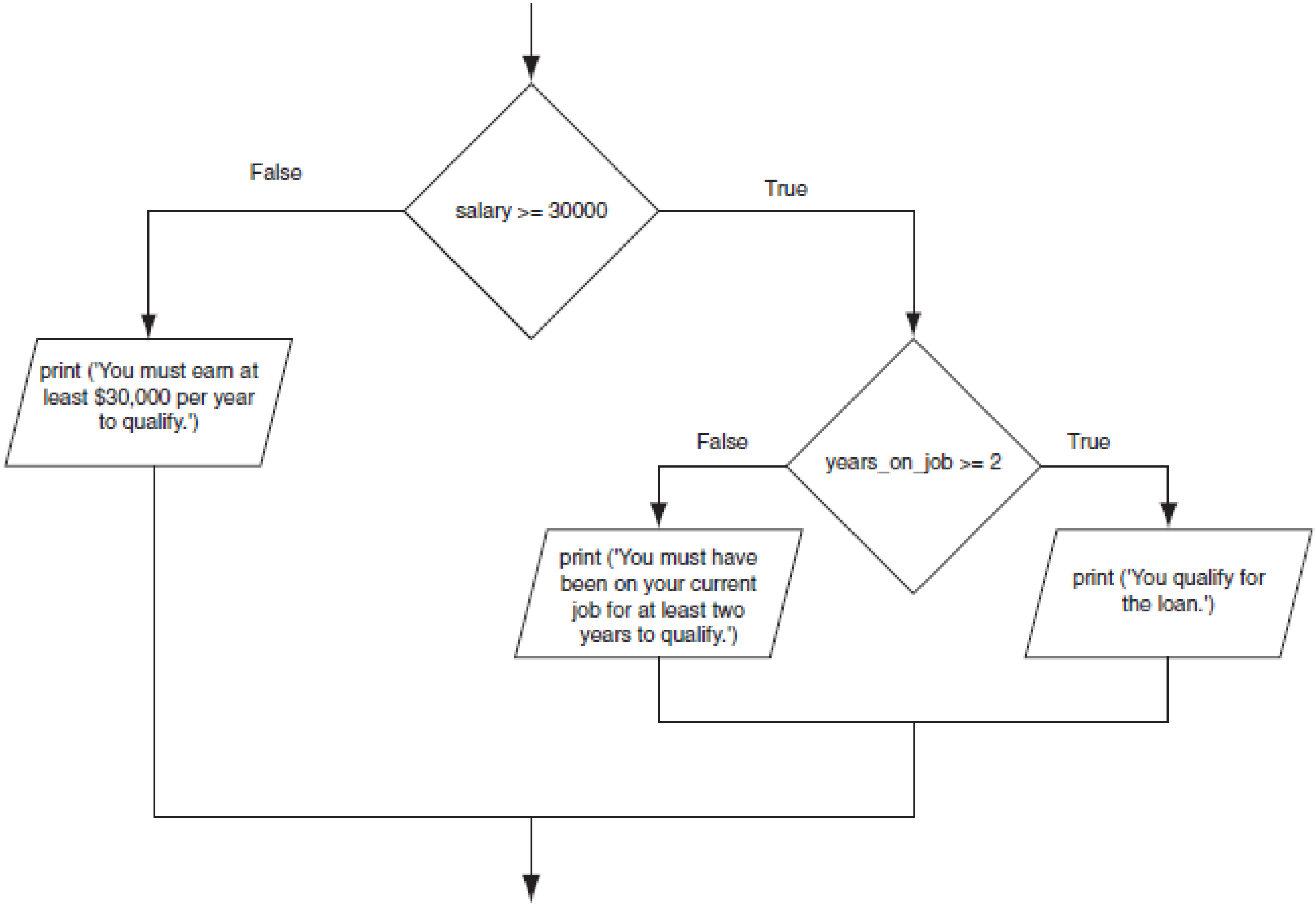
Topics

- 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

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

Figure 3-12 A nested decision structure


False
salary >= 30000
True
print ('You must earn at least $30,000 per year to qualify.')
False
years_on_job >= 2
True
print ('You must have been on your current job for at least two years to qualify.')
print ('You qualify for the loan.')

```
# This program determines whether a bank customer
# qualifies for a loan.

MIN_SALARY = 30000.0 # The minimum annual salary
MIN_YEARS = 2 # The minimum years on the job

# Get the customer's annual salary.
salary = float(input('Enter your annual salary: ‘))

# Get the number of years on the current job.
years_on_job = int(input('Enter the number of' +
'years employed: ‘))

# Determine whether the customer qualifies.
if salary >= MIN_SALARY:
    if years_on_job >= MIN_YEARS:
        print ('You qualify for the loan.’)
    else:
        print ('You must have been employed’,
                'for at least', MIN_YEARS,
                'years to qualify.')
else:
    print ('You must earn at least $’,
            format(MIN_SALARY, ',.2f’),
           ' per year to qualify.', sep='')
```

loan_qualifier.py

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**

```
# This program gets a numeric test score from the
# user and displays the corresponding letter grade.

# Variables to represent the grade thresholds
A_score = 90
B_score = 80
C_score = 70
D_score = 60

# Get a test score from the user.
score = int(input('Enter your test score: '))

# Determine the grade.
if score >= A_score:
    print('Your grade is A.')
else:
    if score >= B_score:
        print('Your grade is B.')
    else:
        if score >= C_score:
            print('Your grade is C.')
        else:
            if score >= D_score:
                print('Your grade is D.')
            else:
                print('Your grade is F.')
```

Tedious!

grader.py

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:

```
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.

grader.py

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:

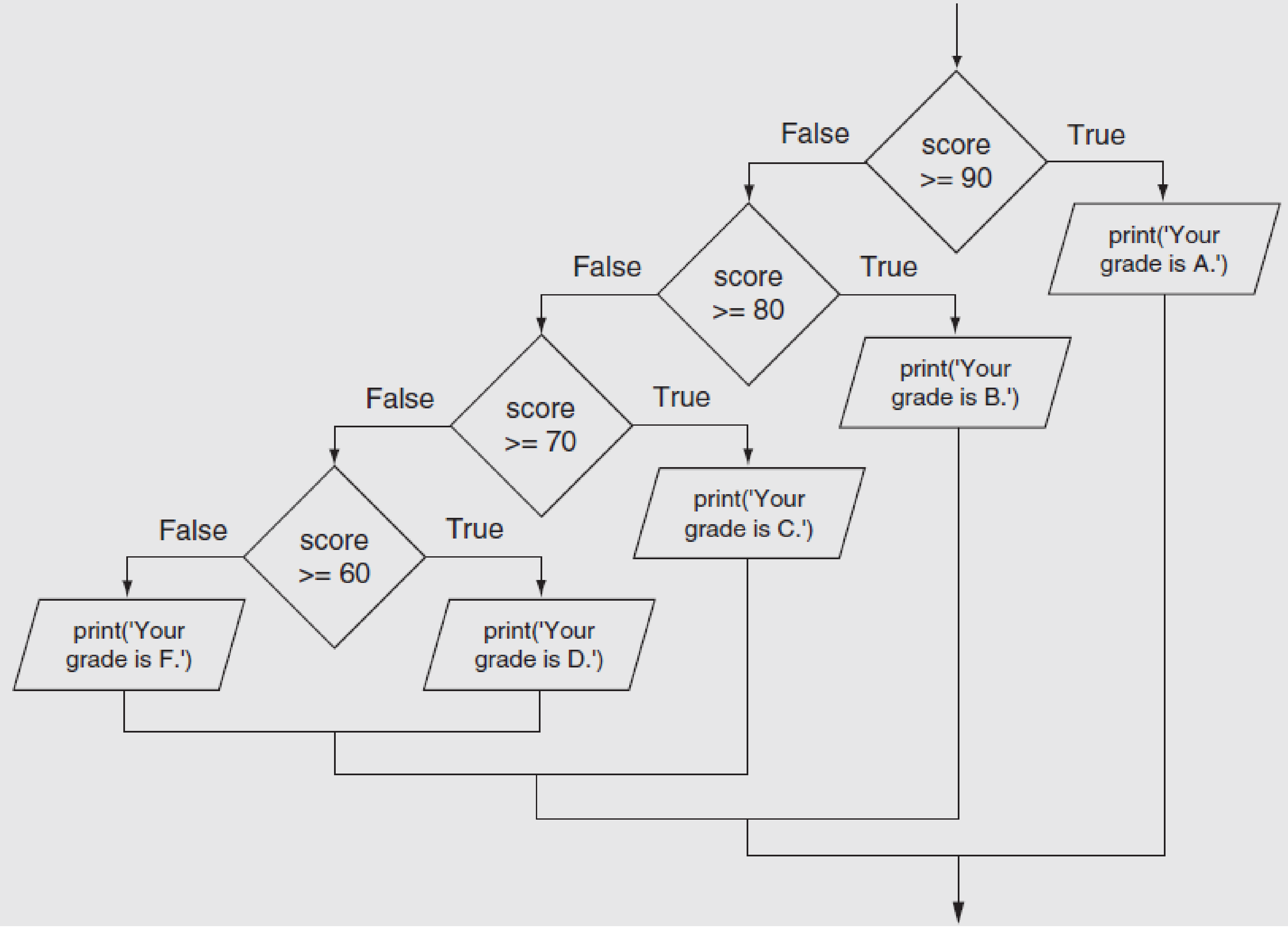
```
if score >= A_score:
    print ('Your grade is A.')
elif score >= B_score:
    print ('Your grade is B.')
elif score >= C_score:
    print ('Your grade is C.')
elif score >= D_score:
    print ('Your grade is D.')
else:
   print ('Your grade is F.')
```

grader02.py

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

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.')

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

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

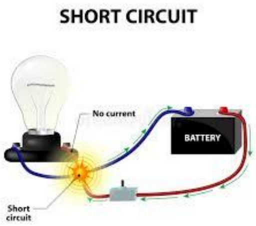
The `or` Operator

- **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

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**

Expression	Value of the Expression
true	false
false	true

```
if not(temperature > 100):
    print ('This is below the maximum temperature.')
```

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`

The Loan Qualifier Program Revisited Ver#1

```
if salary >= MIN_SALARY:
   if years_on_job >= MIN_YEARS:
      print('You qualify for the loan.’)
   else:
      print('You must have been employed’,
               'for at least', MIN_YEARS,
               'years to qualify.')
else:
   print('You must earn at least $’,
           format(MIN_SALARY, ',.2f’),
           ' per year to qualify.', sep='')
```

The Loan Qualifier Program Revisited Ver#2

loan_qualifier2.py

```
# This program determines whether a bank customer
# qualifies for a loan.

MIN_SALARY = 30000.0     # The minimum annual salary
MIN_YEARS = 2            # The minimum years on the job

# Get the customer's annual salary.
salary = float(input('Enter your annual salary: '))

# Get the number of years on the current job.
years_on_job = int(input('Enter the number of ‘ +
                         'years employed: '))

Determine whether the customer qualifies.
if salary >= MIN_SALARY and years_on_job >= MIN_YEARS:
    print('You qualify for the loan.')
else:
    print('You do not qualify for this loan.')
```

The Loan Qualifier Program Revisited Ver#3

```
This program determines whether a bank customer
# qualifies for a loan.

MIN_SALARY = 30000.0 # The minimum annual salary
MIN_YEARS = 2 # The minimum years on the job

# Get the customer's annual salary.
salary = float(input('Enter your annual salary: '))

# Get the number of years on the current job.
years_on_job = int(input('Enter the number of ‘ +
                         'years employed: ‘))
# Determine whether the customer qualifies.
if salary >= MIN_SALARY or years_on_job >= MIN_YEARS:
    print('You qualify for the loan.')
else:
    print('You do not qualify for this loan.')
```

loan_qualifier3.py

מה יצא
בהרצה?

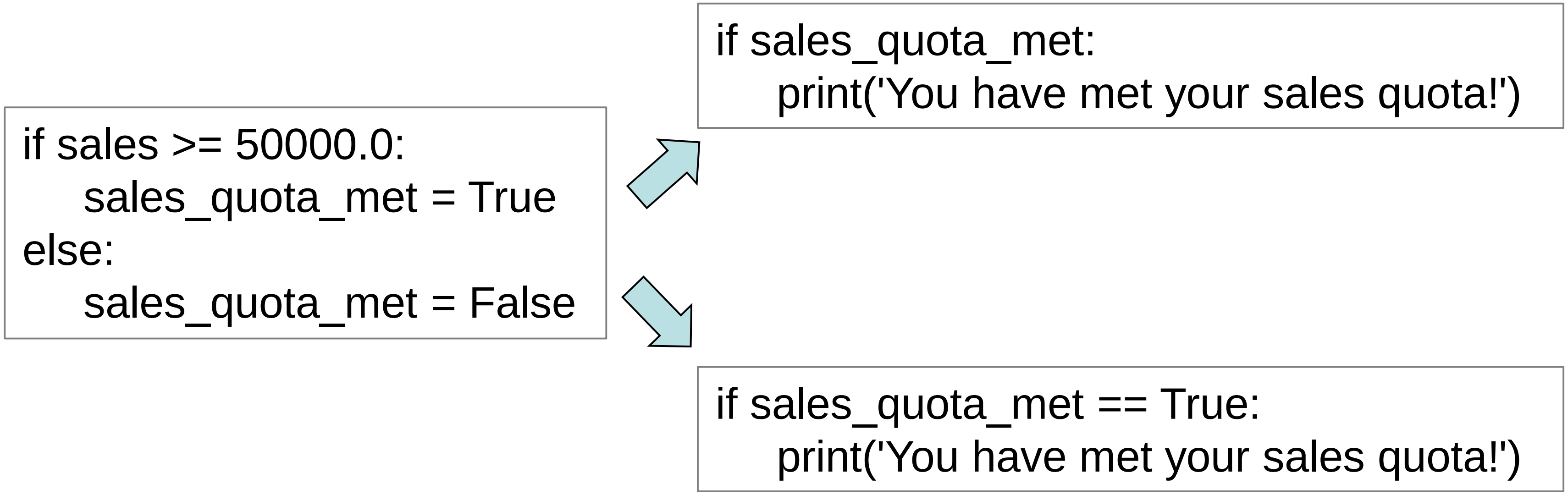
Topics

- 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

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 Variables



```
if sales >= 50000.0:
    sales_quota_met = True
else:
    sales_quota_met = False
```

```
if sales_quota_met:
    print('You have met your sales quota!')
```

```
if sales_quota_met == True:
    print('You have met your sales quota!')
```

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**

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:**

```
if turtle.ycor() < 0:
    turtle.goto(0, 0)
```

```
if turtle.xcor() > 100 and turtle.xcor() < 200:
    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.)**

```
>>> turtle.heading()
0.0
>>>
```

- **Example of calling the function in an `if` statement:**

```
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:**

```
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.**

```
>>> turtle.isvisible()
True
>>>
>>> not(turtle.isvisible())
False
```

- **Example of calling the function in an `if` statement:**

```
if turtle.isvisible():
    turtle.hideturtle()
if not(turtle.isvisible()):
    turtle.showturtle()
```

Turtle Graphics: Determining the State of the Turtle

- When you call `turtle.pencolor()` without passing an argument, the function returns the pen's current color as a string.

```
>>> turtle.pencolor()
'black'
>>>
```

- Example of calling the function in an `if` statement:

```
if turtle.pencolor() == 'red':
    turtle.pencolor('blue')
```

Turtle Graphics: Determining the State of the Turtle

- When you call `turtle.fillcolor()` without passing an argument, the function returns the current fill color as a string.

```
>>> turtle. fillcolor()
'black'
>>>
```

- Example of calling the function in an `if` statement:

```
if turtle.fillcolor() == 'blue':
    turtle.fillcolor('white')
```

Turtle Graphics: Determining the State of the Turtle

- **When you call `turtle.bgcolor()` without passing an argument, the function returns the current background color as a string.**

```
>>> turtle. bgcolor()
‘white'
>>>
```

- **Example of calling the function in an `if` statement:**

```
if turtle.bgcolor() == 'white':
    turtle.bgcolor('gray')
```

Turtle Graphics: Determining the State of the Turtle

- **When you call `turtle.pensize()` without passing an argument, the function returns the pen's current size as a string.**

```
>>> turtle.pensize()
1
>>>
```

- **Example of calling the function in an `if` statement:**

```
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.

```
>>> turtle.speed()
1
>>>
```

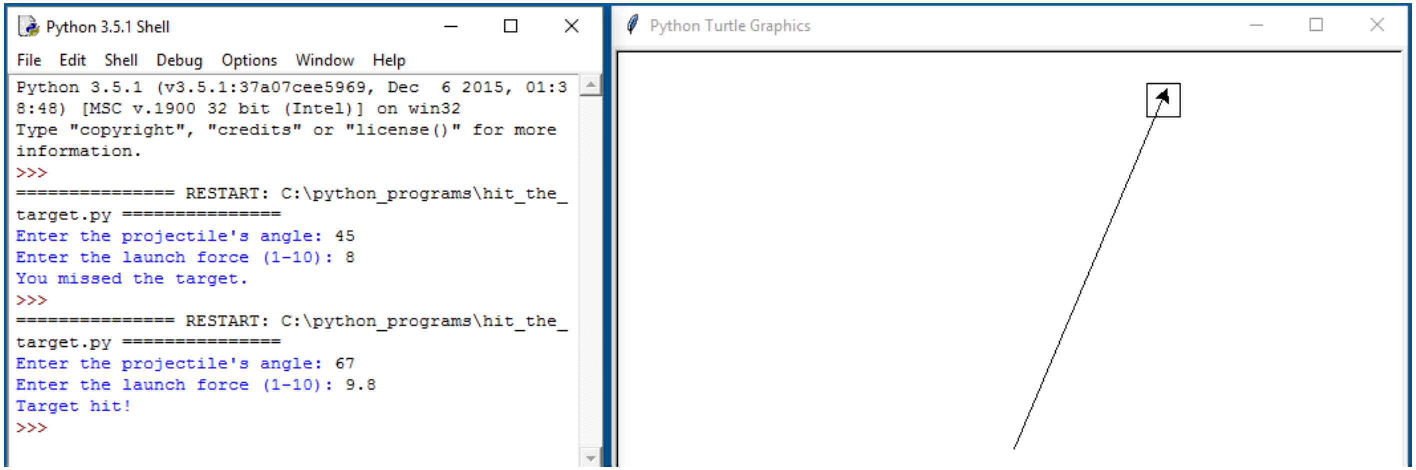
- Example of calling the function in an `if` statement:

```
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.

hit_the_target.py



```
Python 3.5.1 (v3.5.1:37a07cee5969, Dec  6 2015, 01:3
8:48) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more
information.
>>>
=============== RESTART: C:\python_programs\hit_the_
target.py ===============
Enter the projectile's angle: 45
Enter the launch force (1-10): 8
You missed the target.
>>>
=============== RESTART: C:\python_programs\hit_the_
target.py ===============
Enter the projectile's angle: 67
Enter the launch force (1-10): 9.8
Target hit!
>>>
```

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