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

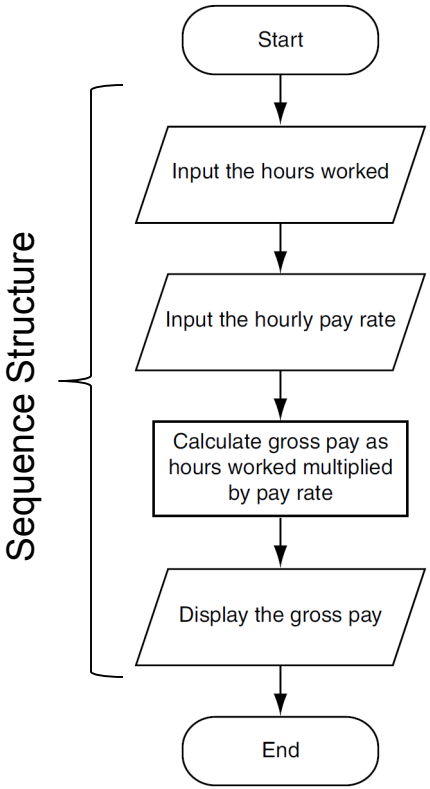
- **<u>Control structure</u>: logical design that controls order in which set of statements execute in a program**
- **<u>Sequence structure</u>: set of statements that execute in the order they appear**
  - All the programs that we have written in Chapter 2 has sequence structure.
- **<u>Decision structure</u>: specific action(s) performed only if a condition exists – `if` statement used in Python**
  - Also known as selection structure

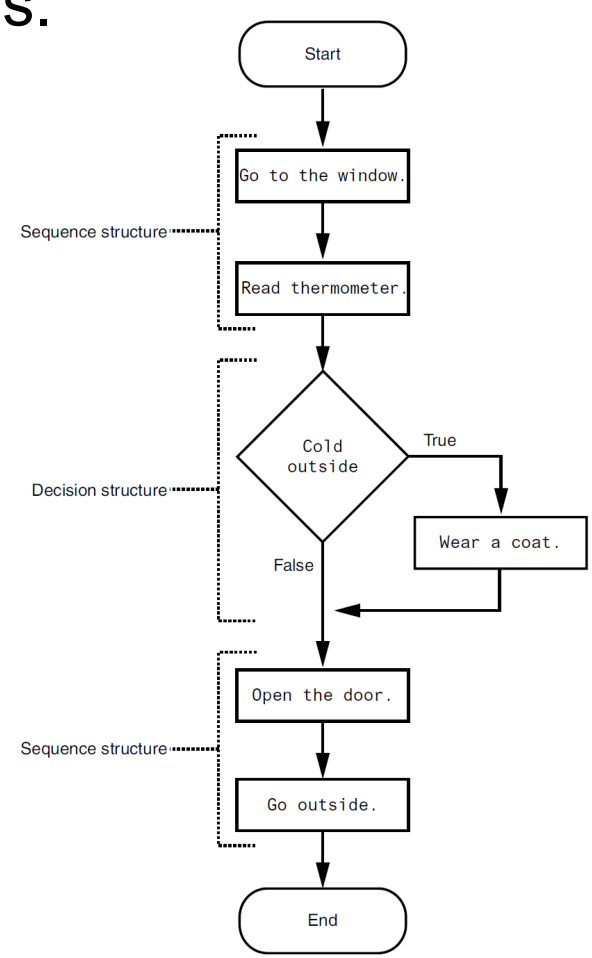
  **<u>Example:</u>** Calculating extra payments if the worked hours is more than 40 hours/week.

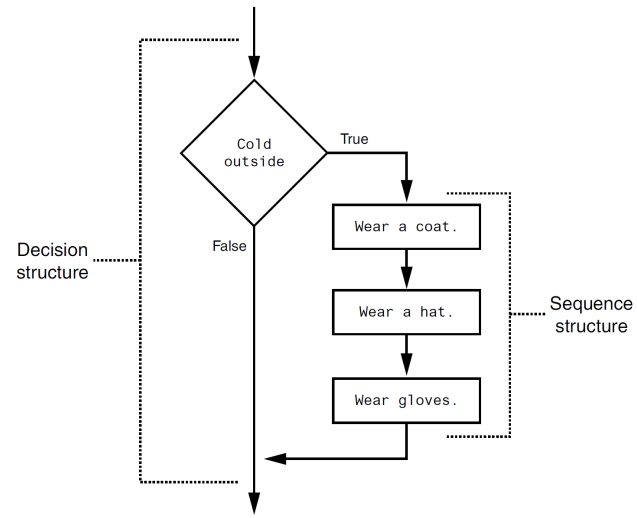
- **There is also <u>Repetition structures</u> (Chapter 4)**

# Examples to Control Structures

- Programs are usually designed as combinations of different control structures.



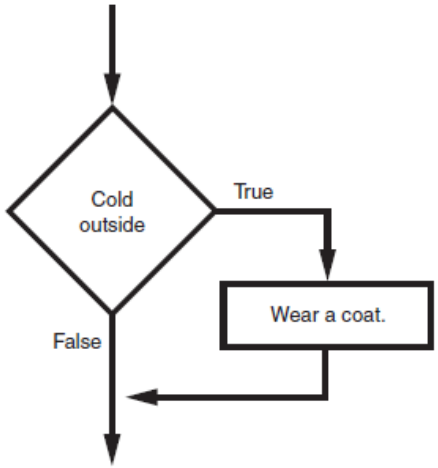




<u>Note:</u> In Section 3.4 we will study Nested – if Statements where a decision structure is placed another decision structure.

# The `if` Statement

- **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 – called simple `if`**
  - If condition is not True, exit the structure and proceed

**Figure 3-1** A simple decision structure

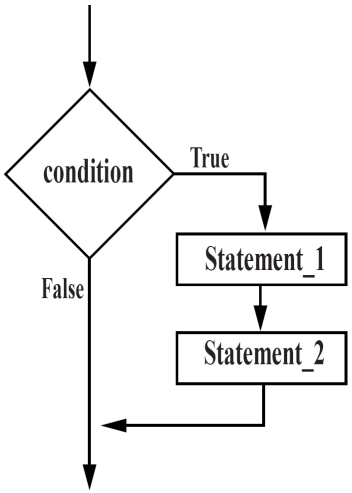



**In this example condition is "cold outside". If "cold outside" condition is True than "Wear a coat" statement is executed.**

**Python syntax:**

```
if Cold Outside:
        Wear a coat.
```

# The `if` Statement (cont'd.)




- **<u>Python syntax:</u>**

```
if condition:
    statement_1
    statement_2
    ...............................
```

*Part with same indentation called block in Python*
*More statement can be placed as many as needed.*

..... => *this is outside* `if`

- **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
    - Statements within decision structure must have the same amount of indentation.
    - It is possible to put one if in another if but indentation must be done properly. This is called nested – if structure.

# Boolean Expressions - Conditions

- **<u>Boolean expression</u>: expression tested by *if statement* to determine if it is True or False**
  - <u>Example:</u> a > b
    - Result is `True` if a is greater than b; `False` otherwise
- **Conditions also called as Boolean Expression can be created by using relational operators. The result of conditions are `True` or `False`.**

**Table 3-1** Relational operators

| Operator | Meaning |
|---|---|
| > | Greater than |
| < | Less than |
| >= | Greater than or equal to |
| <= | Less than or equal to |
| == | Equal to |
| != | Not equal to |

- Do not confuse == operator with assignment operator (=)

# Boolean Expressions and Relational Operators

**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? |

Example: Assume `x = 1 , y =0` and `z=1`

```
>>> x > y Enter
True
>>> y > x
False
>>> x >= y Enter
True
>>> x >= z Enter
True
>>> x <= z Enter
True
>>> x <= y Enter
False
```

```
>>> x == y Enter
False
>>> x == z Enter
True
>>> x != y Enter
True
>>> x != z Enter
False
```

# Example: Using simple-if statement

**Write a program that takes two integer numbers from user then compares them with all possible conditions.**

```
# Taking two numbers as integer from user
num1=int(input('Enter the first number:'))
num2=int(input('Enter the second number:'))
#Comparing them with all possiblities
if num1 > num2:
    print( num1 , '>',num2)
if num1 < num2:
    print( num1 , '<',num2)
if num1 >= num2:
    print( num1 , '>=',num2)
if num1 <= num2:
    print( num1 , '<=',num2)
if num1 == num2:
    print( num1 , '=',num2)
if num1 != num2:
    print( num1 , '!=',num2)
```

### Some Program Outputs

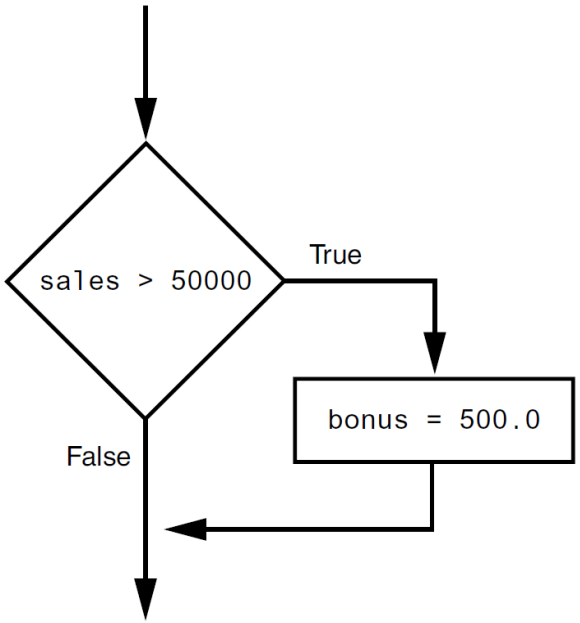
```
Enter the first number:15
Enter the second number:16
15 < 16
15 <= 16
15 != 16
```

```
Enter the first number:8
Enter the second number:14
8 < 14
8 <= 14
8 != 14
```

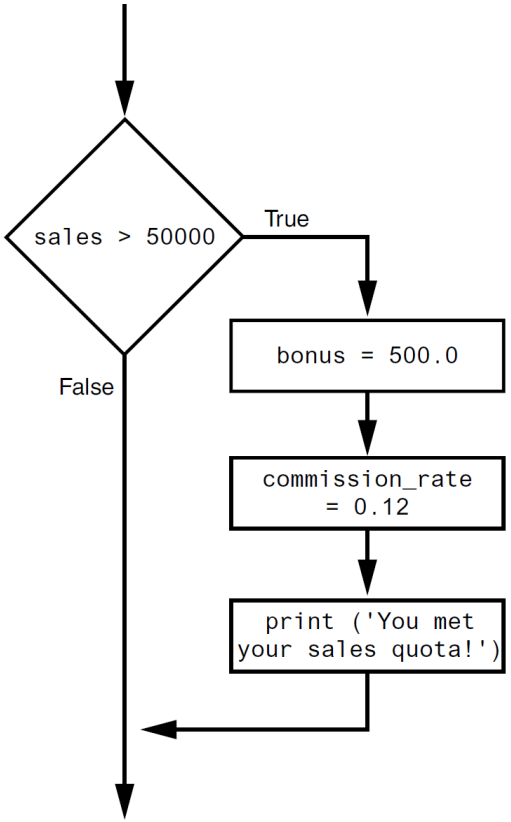
```
Enter the first number:12
Enter the second number:12
12 >= 12
12 <= 12
12 = 12
```

# Boolean Expressions and Relational Operators (cont'd.)

- **Using a Boolean expression with the relational operators**




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




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

**3.4** What is a Boolean expression?

Expressions tested by `if` statement to determine True or False. Also called condition.

**3.5** What types of relationships between values can you test with relational operators?

6 different relations can be tested. `>, <, >=, <=, ==, !=`

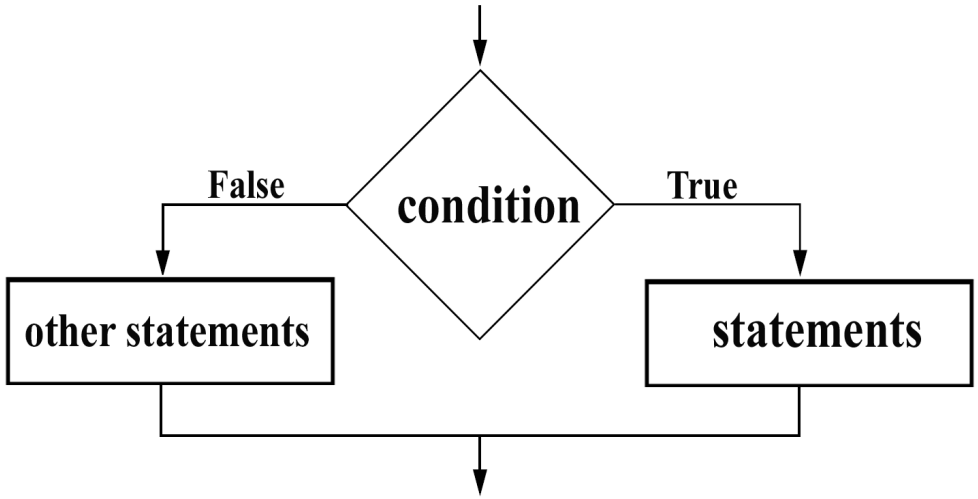
**3.6** Write an if statement that assigns `0` to `x` if `y` is equal to `20`.

```
if y == 20:
    x = 0
```

**3.7** Write an if statement that assigns `0.2` to `commissionRate` if `sales` is greater than or equal to `10000`.

```
if sales >= 10000:
    commissionRate = 0.2
```

# The `if-else` Statement

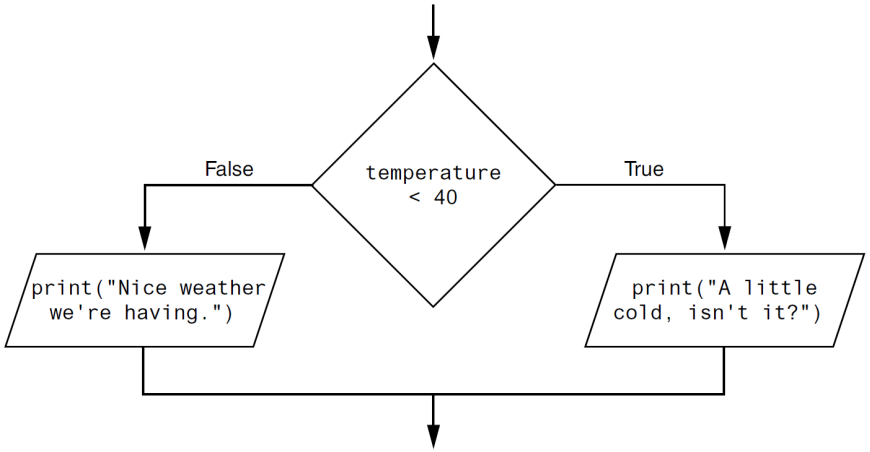



- **<u>Dual alternative decision structure</u>: two possible paths of execution**
  - One is taken if the condition is True, and the other if the condition is False
  - Syntax:

```
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.)




- **When you write an if-else statement, follow these guidelines for indentation:**
  - Make sure the if clause and the else clause are aligned.
  - The if clause and the else clause are each followed by a block of statements. Make sure the statements in the blocks are consistently indented.

Must be aligned

```
if temperature < 40:
        print("A little cold, isn't it?")
        print("Turn up the heat!")
else:
        print("Nice weather we're having.")
        print("Pass the sunscreen.")
```

Must have same indentation

# Example: Program ODD / EVEN

**Write a program that reads an integer and determines and prints whether it is odd or even.**

**Hint:** Use the remainder operator. An even number is a multiple of two. Any multiple of two leaves a remainder of zero when divided by 2.

### Single alternative / simple – if Statement

```
# Get the integer input first.
number = int(input('Enter an integer:'))
# Calculate the remainder
mod = number % 2
# Check if mod is 0
if mod == 0:
    print('Entered number is EVEN!')
# Check if mod is 1
if mod == 1:
    print('Entered number is ODD!')
```

### Dual alternative/ if - else Statement

```
# Get the integer input first.
number = int(input('Enter an integer:'))
# Calculate the remainder
mod = number % 2
# Check if mod is 0
if mod == 0:
    print('Entered number is EVEN!')
else:
    print('Entered number is ODD!')
```

When using simple –if statement, programmer should check all possibilities with appropriate if Boolean expressions.

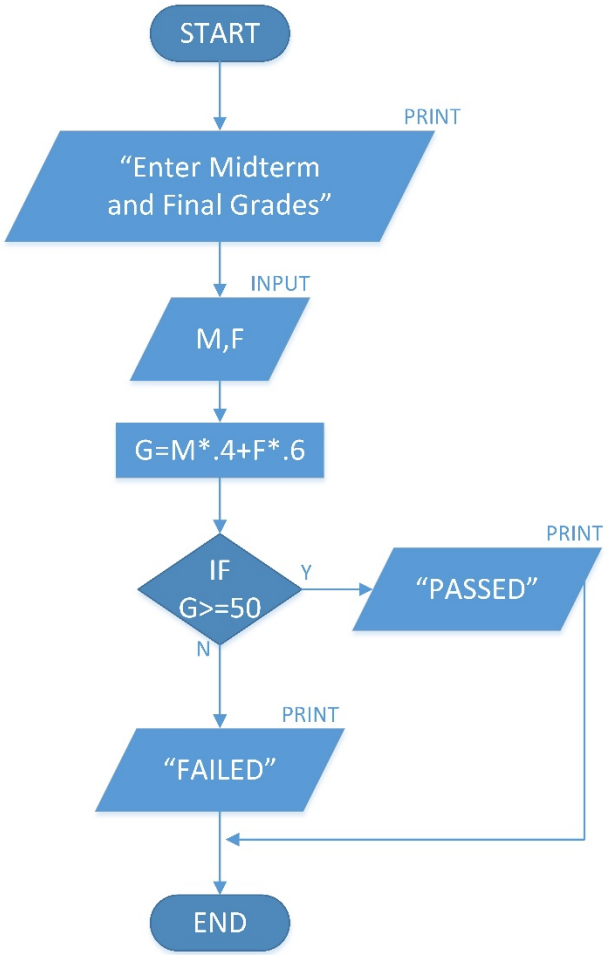
### Example Program Outputs

```
Enter an integer number: 11
Entered number is ODD!

Enter an integer number: -146
Entered number is EVEN!
```

# Example: Program Fail / Pass

**Write a python program. The program reads the midterm and final exam results of a student then calculates the final grades as 40% of the midterm and 60% of final exam. Then writes to the screen "PASSED" if the grade is equals or greater than 50 or "FAILED" if grade is less than 50.**




```
# Pass Grade is defined as named constant
PASS_GRADE = 50
# Get the test scores.
mid = int(input('Enter the score for midterm exam: '))
fin = int(input('Enter the score for final exam: '))
# Calculate the grade as 40% and 60%
avg = mid*.4 + fin*.6
# Check Failed or Passed
if avg >= PASS_GRADE:
    print('Congratulations!')
    print('You have passed!')
else:
    print('Sorry!')
    print('You have failed!')
#Now also printing the average
print('The  average score is', avg)
```

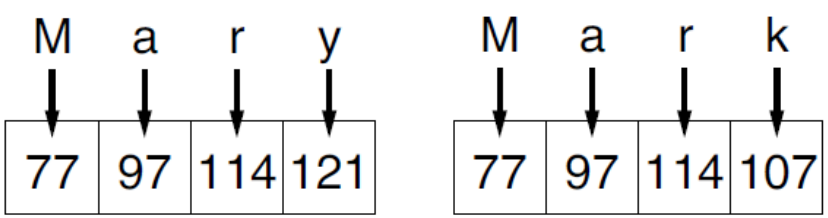
## Example Program Output

```
Enter the score for midterm exam: 34
Enter the score for final exam: 63
Congratulations!
You have passed!
The  average score is 51.4
```

# 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

**Figure 3-9** Comparing each character in a string




**String comparison is done according to the dictionary A-Z ordering for the strings.**
**<u>Example:</u> ‘bursa’ > ‘adana’**
**Also lower case letters are greater than capital case letters**
**<u>Example:</u> ‘adana‘ > ‘Adana’**

APPENDIX C The ASCII Character Set

| Code | Character | Code | Character |
|---|---|---|---|
| 78 | N | 104 | h |
| 79 | O | 105 | i |
| 80 | P | 106 | j |
| 81 | Q | 107 | k |
| 82 | R | 108 | l |
| 83 | S | 109 | m |
| 84 | T | 110 | n |
| 85 | U | 111 | o |
| 86 | V | 112 | p |
| 87 | W | 113 | q |
| 88 | X | 114 | r |
| 89 | Y | 115 | s |
| 90 | Z | 116 | t |
| 91 | [ | 117 | u |
| 92 | \ | 118 | v |
| 93 | ] | 119 | w |
| 94 | ^ | 120 | x |
| 95 | _ | 121 | y |
| 96 | ` | 122 | z |
| 97 | a | 123 | { |
| 98 | b | 124 | \| |
| 99 | c | 125 | } |
| 100 | d | 126 | ~ |
| 101 | e | 127 | DEL |
| 102 | f | | |
| 103 | g | | |

# Checkpoint



3.11 What would the following code display?

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

**Code Output:**

```
z is not less than a.
```

3.12 What would the following code display?

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

**Code Output:**

```
Boston
New York
```

# Nested Decision Structures

- **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

**Solution – Next Slide**

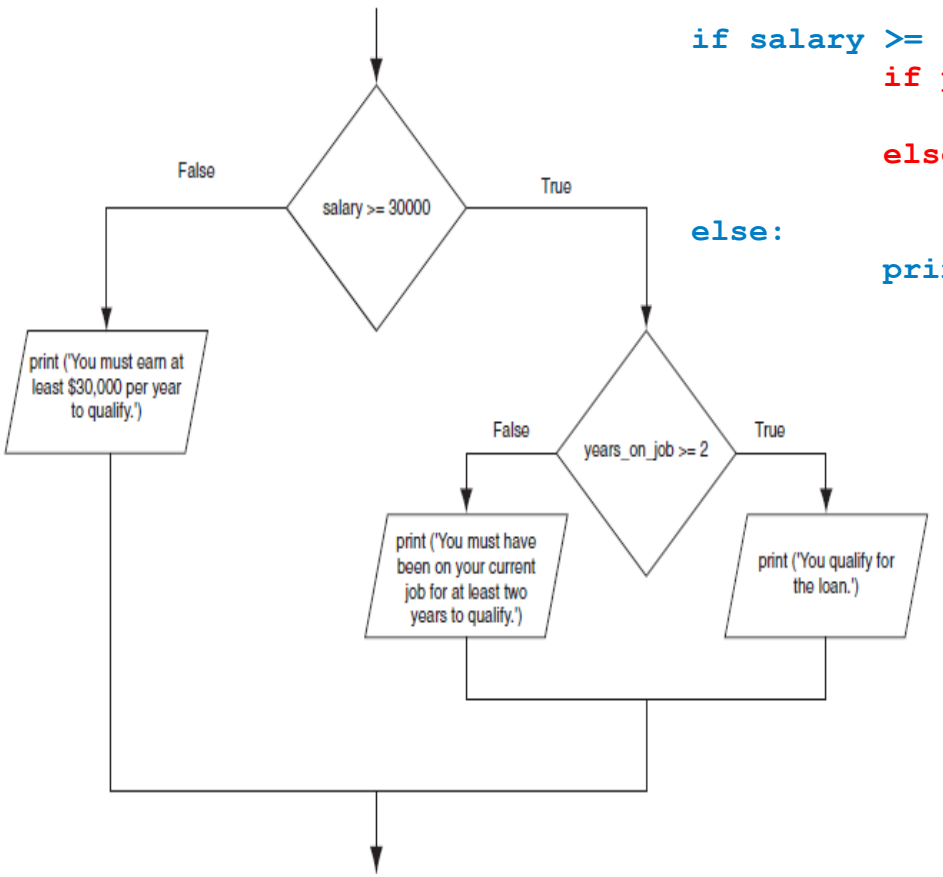
# Example: Loan Qualifier



## Two conditions

Salary must be >= 30000
Years on Job >= 2 years

**Figure 3-12** A nested decision structure




```
if salary >= 30000:
        if years_on_job >= 2:
                print('You qualify for the loan.')
        else:
                print('You must have been employed ....')
else:
        print('You must earn at least ...')
```

# The `if-elif-else` Statement

- **Connected/related conditions can be organized easily by using `if-elif-else` statement.**
- Makes code more readable for programmer
- Can be accomplished by nested `if-else` but code can be complex
- `if`, `elif`, and `else` clauses must be all aligned
- Statements in each block must be consistently indented

```
if-elif-else
Same
indentation

if condition_1:
        statement
        statement
        etc.
elif condition_2:
        statement
        statement
        etc.
else:
        statement
        statement
        etc.
```

If condition_1 is True then
These statements are executed.

If condition_2 is True then
These statements are executed.

If none is True then
These statements are executed.

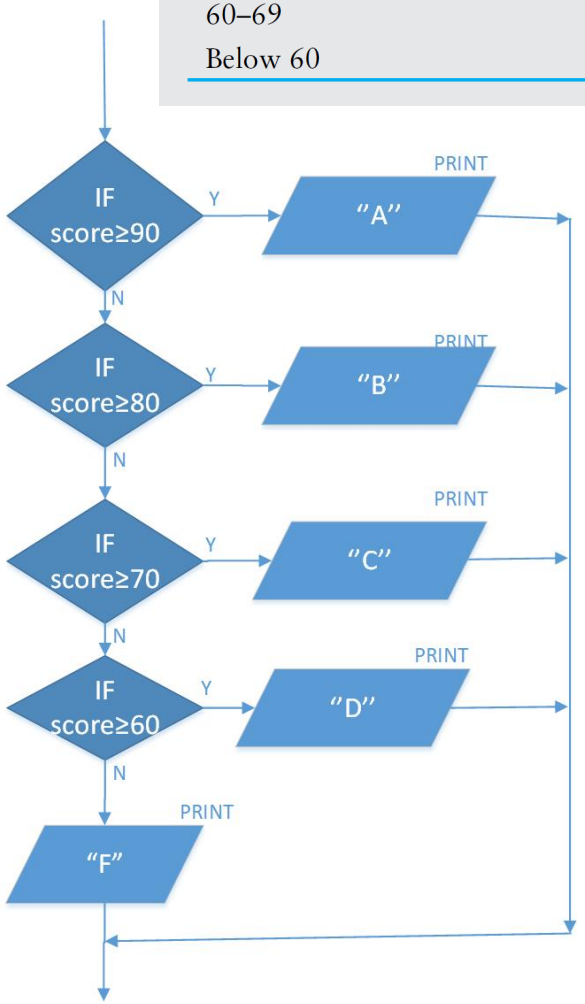
Multiple `elif` clauses can be used as much as needed.

# Example: `if-elif-else` Statement

**Letter Grade Calculator.**

Page 152

| Test Score | Grade |
|---|---|
| 90 and above | A |
| 80–89 | B |
| 70–79 | C |
| 60–69 | D |
| Below 60 | F |




```
.......................
.......................
if score >= 90:
      print('Your grade is A.')
elif score >= 80:
      print('Your grade is B.')
elif score >= 70:
      print('Your grade is C.')
elif score >= 60:
      print('Your grade is D.')
else:
      print('Your grade is F.')
.......................
.......................
```

3.13 Convert the following code to an `if-elif-else` statement:

```
if number == 1:
    print('One')

else:
    if number == 2:
        print('Two')
    else:
        if number == 3:
            print('Three')
        else:
            print('Unknown')
```

**`If-elif-else` Statement**

```
if number == 1:
    print('One')

elif number == 2:
    print('Two')

elif number == 3:
    print('Three')

else:
    print('Unknown')
```

# Logical Operators

- **<u>Logical operators</u>: operators that can be used to create complex Boolean expressions – by using logical operators, we can connect conditions.**
  - `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

**Table 3-3** Logical operators

| Operator | Meaning |
|---|---|
| `and` | The `and` operator connects two Boolean expressions into one compound expression. Both subexpressions must be true for the compound expression to be true. |
| `or` | The `or` operator connects two Boolean expressions into one compound expression. One or both subexpressions must be true for the compound expression to be true. It is only necessary for one of the subexpressions to be true, and it does not matter which. |
| `not` | The `not` operator is a unary operator, meaning it works with only one operand. The operand must be a Boolean expression. The `not` operator reverses the truth of its operand. If it is applied to an expression that is true, the operator returns false. If it is applied to an expression that is false, the operator returns true. |

# 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_1 | expression_2 | expression_1 and expression_2 |
|---|---|---|
| False | False | False |
| False | True | False |
| True | False | False |
| True | True | True |

In `and` operator, only if all conditions are True then the result becomes 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_1 | expression_2 | expression_1 or expression_2 |
|---|---|---|
| False | False | False |
| False | True | True |
| True | False | True |
| True | True | True |

In `or` operator, only if one condition is True then the result becomes True.

# Short-Circuit Evaluation

- **<u>Short circuit evaluation</u>: deciding the value of a compound Boolean expression after evaluating only one sub expression**
  - Performed by the `or` and `and` operators

    - **<u>For `or` operator:</u>** If left operand is True, compound expression is True. Otherwise, evaluate right operand

      exp1 `or` exp2 `or` exp3 `or` exp4 `or` exp5 `or` …
      `if exp1 is True then result is True.`
      `if exp1 is False then check if exp2 True and so on.`

    - **<u>For `and` operator:</u>** If left operand is False, compound expression is False. Otherwise, evaluate right operand

      exp1 `and` exp2 `and` exp3 `and` exp4 `and` exp5 `and` …
      `if exp1 is False then result is False`
      `if exp1 is True then check if exp2 False and so on.`

# 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
  - Sometimes also called as inversion operator
- **Truth table for the `not` operator**

| expression | not expression |
|---|---|
| True | False |
| False | True |

# Checking Numeric Ranges with Logical Operators

- **To determine whether a numeric value is within a specific range of values, use `and`**
  - Example: $10 \leq x \leq 20$

```
x >= 10 and x <= 20
```

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

```
x < 10 or x > 20
```

# Example: Loan Qualifier - Revisited

## Two conditions

Salary must be >= 30000
Years on Job >= 2 years

Let's write the loan qualifier program by using `if-else` statement

```
.....................................
# Determine whether the customer qualifies.
if salary >= 3000 and years_on_job >=  2:
      print('You qualify for the loan.')
else:
      print('You do not qualify for this loan.')
```

# Boolean Variables

- **<u>Boolean variable</u>: references one of two values, `True` or `False`**
  - Represented by `bool` data type
- **Commonly used as flags**
  - <u>Flag</u>: 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**

```
>>> turtle.forward(100)
>>> turtle.xcor()
100.0
>>> turtle.ycor()
0.0
```

**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)


if turtle.ycor() < 0:
        turtle.done()
```

# 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.)**

```
>>> import turtle
>>> turtle.heading()
0.0
>>> turtle.right(45)
>>> turtle.heading()
315.0
```

**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.**

```
>>> import turtle
>>> turtle.isdown()
True
>>> turtle.penup()
>>> turtle.isdown()
False
>>> turtle.pendown()
>>> turtle.isdown()
True
```

**Examples 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.**

```
>>> import turtle
>>> turtle.isvisible()
True
>>> turtle.hideturtle()
>>> turtle.isvisible()
False
```

**Examples 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.**

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

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

```
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:**

```
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.**

```
>>> import turtle
>>> turtle.pencolor()
'black'
>>> turtle.fillcolor()
'black'
>>> 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.**

```
>>> import turtle
>>> turtle.pensize()
1
>>> turtle.pensize(10)
>>> turtle.pensize()
10
```

**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.**

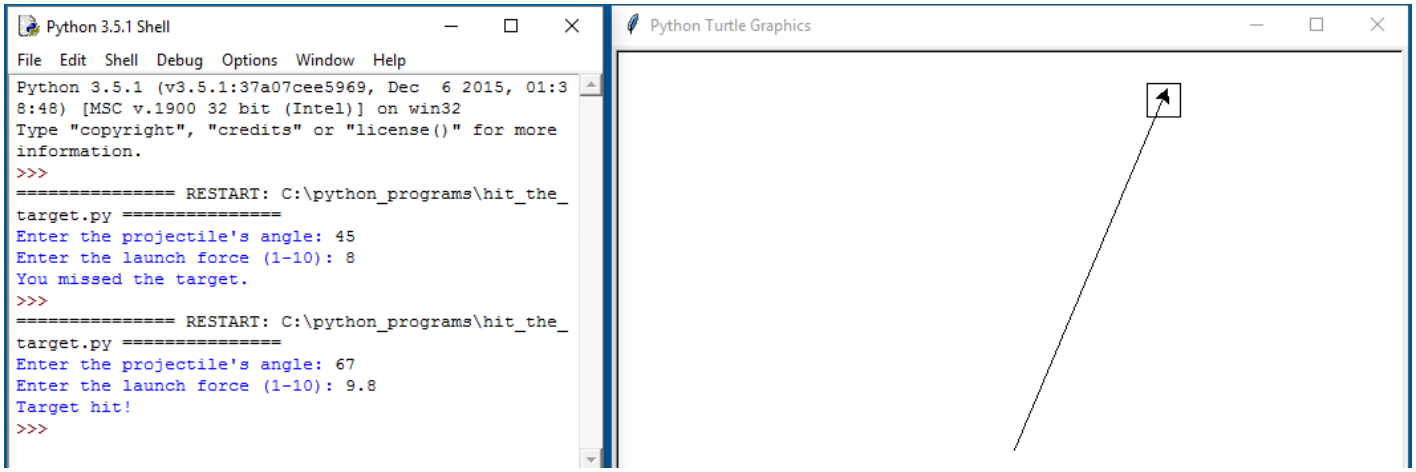
```
>>> import turtle
>>> turtle.speed()
3
>>> turtle.speed(7)
>>> turtle.speed()
7
```

**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 Page 167 for numerous examples of determining the state of the turtle.**



3.26 How do you determine the turtle's pen color? How do you determine the current fill color? How do you determine the current background color of the turtle's graphics window?

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

3.28 How do you determine the turtle's current animation speed?

```
>>> turtle.speed()
3
```

# Review Questions

Page 170-171

## Multiple Choice

1. A _____ structure can execute a set of statements only under certain circumstances.
   a. sequence
   b. circumstantial
   ✓c. decision
   d. Boolean
2. A _____ structure provides one alternative path of execution.
   a. sequence
   ✓b. single alternative decision
   c. one path alternative
   d. single execution decision

7. You use a(n) _____ statement to write a dual alternative decision structure.
   a. `test-jump`
   b. `if`
   ✓c. `if-else`
   d. `if-call`

11. The _____ operator takes a Boolean expression as its operand and reverses its logical value.
   a. `and`
   b. `or`
   ✓c. `not`
   d. `either`

## Algorithm Workbench



2. Write an `if` statement that assigns 10 to the variable `b`, and 50 to the variable `c` if the variable `a` is equal to 100.

```
if a == 100:
        b=10
        c=50
```

6. Write an `if-else` statement that assigns `True` to the `again` variable if the `score` variable is within the range of 40 to 49. If the `score` variable's value is outside this range, assign `False` to the `again` variable.

```
if score >= 40 and score <= 49:
        again = True
else:
        again=False
```

7. Write an `if-else` statement that determines whether the `points` variable is outside the range of 9 to 51. If the variable's value is outside this range it should display "Invalid points." Otherwise, it should display "Valid points."

```
if points < 9 or points > 51:
        print ( 'Invalid points.' )
else:
        print ( 'Valid points.' )
```

8. Write an `if` statement that uses the turtle graphics library to determine whether the turtle's heading is in the range of 0 degrees to 45 degrees (including 0 and 45 in the range). If so, raise the turtle's pen.

```
if turtle.heading >= 0 and turtle.heading <= 45:
        turtle.penup()
```

# Programming Exercises



## 1. Number Analyzer

Write a program that asks the user to enter an integer. The program should display "Positive" if the number is greater than 0, "Negative" if the number is less than 0, and "Zero" if the number is equal to 0. The program should then display "Even" if the number is even, and "Odd" if the number is odd.

**Program**

```
# Get the number.
number = int(input('Enter an integer: '))

# Determine if the number is positive, negative or zero.
if number > 0:
    print('Positive')
elif number < 0:
    print('Negative')
else:
    print('Zero')

# Determine if the number is even or odd.
if number % 2 == 0:
    print('Even')
else:
    print('Odd')
```

**Example Program Output**

```
Enter an integer: 17
Positive
Odd
```

# Programming Exercises



## 3. Quarter of the Year

Write a program that asks the user for a month as a number between 1 and 12. The program should display a message indicating whether the month is in the first quarter, the second quarter, the third quarter, or the fourth quarter of the year.

**Program**

```
# Get the number for the month.
month = int(input('Enter a number (1-12) for the month: '))

# Determine the quarter of the year and display it.
if month >= 1 and month <= 3:
    print('First Quarter')
elif month >= 4 and month <= 6:
    print('Second Quarter')
elif month >= 7 and month <= 9:
    print('Third Quarter')
elif month >= 10 and month <= 12:
    print('Fourth Quarter')
else:
    print('Error: Please enter a number between 1 and 12.')
```

**Example Program Output**

```
Enter a number (1-12) for the month: 7
Third Quarter
```

# Programming Exercises



## 12. Software Sales

A software company sells a package that retails for $99. Quantity discounts are given according to the table. Write a program that asks the user to enter the number of packages purchased. The program should then display the amount of the discount (if any) and the total amount of the purchase after the discount.

**Program**

```
RETAIL_PRICE = 99 # Named constant
# Get number of packages to be purchased
quantity = int(input('Enter the number of packages purchased: '))
# Calculate the discount rate
if quantity > 99:
    discountRate = 0.40
elif quantity > 49:
    discountRate = 0.30
elif quantity > 19:
    discountRate = 0.20
elif quantity > 9:
    discountRate = 0.10
else:
    discountRate = 0
# Calculate the full price
fullPrice = quantity * RETAIL_PRICE
# Calculate the discount amount
discountAmount = fullPrice * discountRate
# Calculate the total amount
totalAmount = fullPrice - discountAmount

# Print results
print ('Discount Amount: $', format(discountAmount, '.2f'))
print ('Total Amount: $', format(totalAmount, '.2f'))
```

**Example Program Output**

```
Enter the number of packages purchased: 100
Discount Amount: $ 3960.00
Total Amount: $ 5940.00
```

# Programming Exercises



## 14. Body Mass Index

Write a program that calculates and displays a person's body mass index (BMI). The BMI is often used to determine whether a person is overweight or underweight for his or her height. A person's BMI is calculated with the following formula: ***BMI* = *weight* (m) / *height*$^2$ (kg)** .The program should ask the user to enter his or her weight and height, then display the user's BMI. *The program should also display a message indicating whether the person has optimal weight, is underweight, or is overweight.* A person's weight is considered to be **optimal** if his or her BMI is between **18.5** and **25**. If the BMI is **less than 18.5**, the person is considered to be **underweight**. If the BMI value is **greater than 25**, the person is considered to be **overweight**.

| BMI | Weight status |
|---|---|
| Below 18.5 | Underweight |
| 18.5–24.9 | Healthy |
| 25.0–29.9 | Overweight |

**Program**

```
# Get the weight and height from the user.
weight = float(input('Enter your weight (kg): '))
height = float(input('Enter your height (m): '))
# Calculate the body mass.
BMI = weight / (height * height)

# Display BMI.
print('Your Body Mass Indicator is', format(BMI, '.2f'))
# Determine and display weight category.
if BMI > 25:
    print('You are overweight.')
elif BMI > 18.5:
    print('You are underweight.')
else:
    print('Your weight is optimal.')
```

**Example Program Output**

```
Enter your weight (kg): 75
Enter your height (m): 1.68
Your Body Mass Indicator is 26.57
You are overweight.
```

# 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