



Computer Programming in Python

Chapter 4

Decisions and Boolean Logic

Chapter 4 Decisions and Boolean Logic

- Decision (Control) Structures
 - Determine the statements that execute based upon whether or not a condition is true
 - A conditional statement is used to determine whether or not a line or lines of code execute

Decisions are based on a True or False condition

Chapter 4 Decisions and Boolean Logic

- Decision (Control) Structures
 - Conditional statements provide multiple paths through a program based on the status of **Boolean** (true or false) conditions
 - If the condition is true, then a statement or statements are executed
 - Otherwise they are not executed

The program path is based on a True or False condition

Chapter 4 Decisions and Boolean Logic

- Example:
 - A Theater has seating for 400 participants
 - Once the Theater has sold 400 tickets, the show has been sold out
 - When this occurs, the Theater displays a “Sold Out” sign at the box office

If 400 tickets have been sold
Display the “Sold Out” sign



Chapter 4 Decisions and Boolean Logic

- **Decision Structures**

- Example:

- If 400 tickets have been sold

- Display the “Sold Out” sign

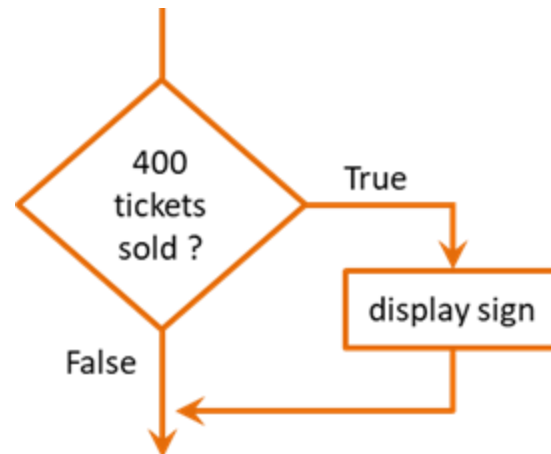
- The condition is tested (have 400 tickets been sold)
 - If it is true, then the “Sold Out” sign is displayed
 - If the condition is false and 400 tickets have not been sold, then the sign will not be displayed.

Chapter 4 Decisions and Boolean Logic

- Decision Structures

- Represented in flowcharts using diamonds

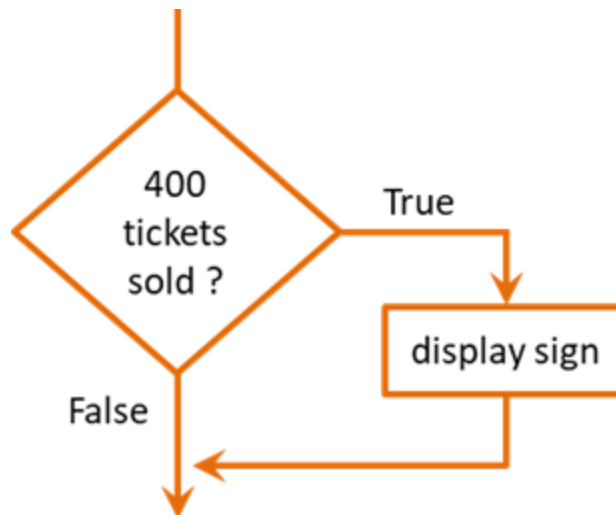
- The different paths that the program can take are shown using lines from the corners of the diamond with arrows indicating the direction with text to indicate the result



Chapter 4 Decisions and Boolean Logic

- Decision Structures

- If the condition is true, then follow the path to display the sign
- Otherwise (if it is false), the program continues
- These paths are the *Flow of Control* or *Order of Operations*



Chapter 4 Decisions and Boolean Logic

- The ‘if’ statement
 - The conditional statement begins with the word *if*, followed by the condition, and ends with a colon
 - This is often referred to as the “if clause”
 - The statement to be executed based upon the condition is below the condition and indented (one tab space)

```
if condition:  
    statement
```


Chapter 4 Decisions and Boolean Logic

- The 'if' statement
 - The interpreter associates the indented statement with the condition
 - If the condition is true, the statement will be executed
 - If the condition is false, then the statement will be skipped

```
if condition:  
    statement
```

Chapter 4 Decisions and Boolean Logic

- The 'if' statement
 - Multiple statements can be associated with a condition and form what is commonly referred to as a ***block of code***
 - A group of statements associated with a condition:

```
if condition:  
    statement1  
    statement2  
    statement3  
    etc.
```

Multiple indented statements form a Block of Code

Chapter 4 Decisions and Boolean Logic

- The 'if' statement
 - If the condition is true, all of the indented statements (the block of code) will be executed
 - If the condition is false, all of the indented statements will be skipped by the interpreter

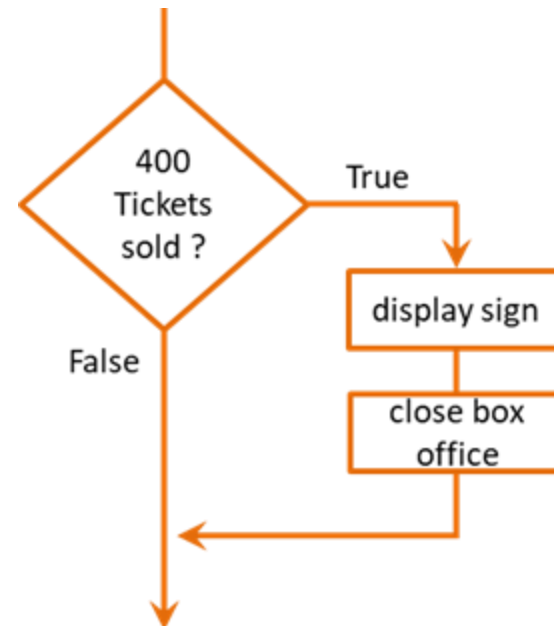
```
if condition:  
    statement1  
    statement2  
    statement3  
    etc.
```

Chapter 4 Decisions and Boolean Logic

- The Theater example continued:
 - Assume that when the show is sold out, in addition to the sold out sign being displayed, the box office is closed

If 400 tickets have been sold

- Display the "Sold Out" sign
- Close the box office



Chapter 4 Decisions and Boolean Logic

- Boolean Expressions
 - Conditional statements are either True or False
 - Referred to as **Boolean** Expressions
 - Named after George Boole
 - Implemented using Relational Operators
 - A value can either be equal to another, greater than another, or less than another

Chapter 4 Decisions and Boolean Logic

- Relational operators
 - Used to test the relationship between items to determine the next step for the program

Operator	Description
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

Chapter 4 Decisions and Boolean Logic

- The Theater example continued:
 - A Boolean expression with a relational operator would be used in the conditional statement for the code
 - The number of tickets sold is either 400 or it is not
 - The expression is True or it is False

```
tickets_sold = int(input('Enter the tickets sold: '))

if tickets_sold == 400:
    print('Display the "Sold Out" sign')
    print('Close the box office')
```

Chapter 4 Decisions and Boolean Logic

- Conditional Expressions

- To allow for another option the *else* clause is used
- An else clause can be thought of as an “*otherwise*” condition for when the relational expression is not true
- In other words

 If this is true:

 Then do this

 else (otherwise):

 Do this

Chapter 4 Decisions and Boolean Logic

- Conditional Expressions

- When the “*if*” condition is true, the statements in the “*if*” block will be executed and the “*else*” block will be skipped
- When the “*if*” condition is false, the “*if*” block will be skipped and the “*else*” block will execute

```
if conditon:  
    statement1  
    statement2  
    etc.
```

```
else:  
    statement1  
    statement2  
    etc.
```

Chapter 4 Decisions and Boolean Logic

- Conditional Expressions

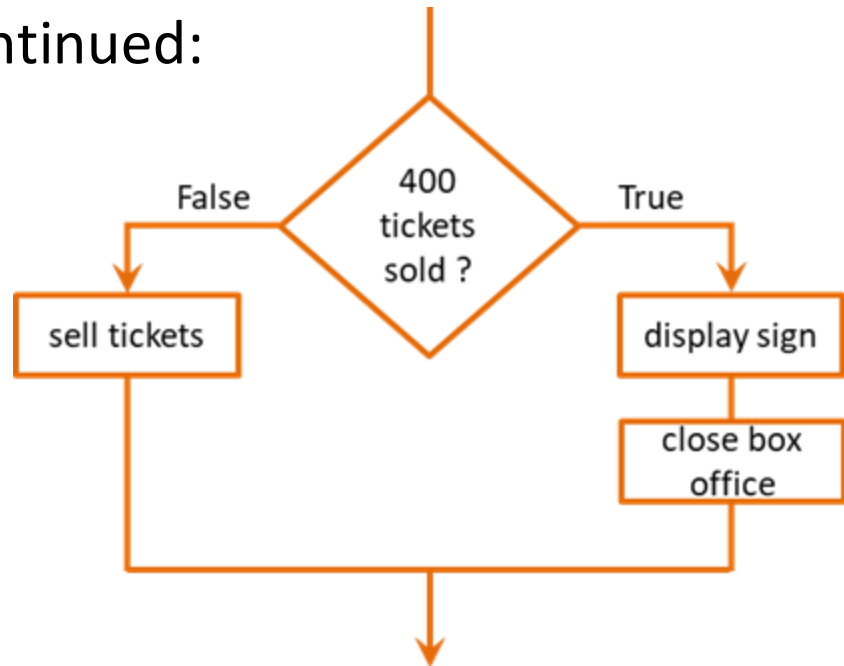
- The Theater example continued:

If 400 tickets have been sold:

- Display the "Sold Out" sign
- Close the box office

else:

- Continue to sell tickets



Chapter 4 Decisions and Boolean Logic

- Conditional Expressions

- The Theater example continued:

```
tickets_sold = int(input('Enter the tickets sold: '))

if tickets_sold == 400:
    print('Display the "Sold Out" sign')
    print('Close the box office')

else:
    print('Keep selling tickets')
```

Chapter 4 Decisions and Boolean Logic

- Conditional Expressions
 - When there are two conditions that must be tested, a second if clause can be added after the first

If this is true:

Then if this is true:

Do this

Chapter 4 Decisions and Boolean Logic

- Conditional Expressions
 - A *nested if* is an if clause within another if clause
 - If *condition1* is false, then *condition2* will not be tested and the statements are skipped

```
if condition1:  
    if condition2:  
        statement1  
        statement2  
        etc.
```

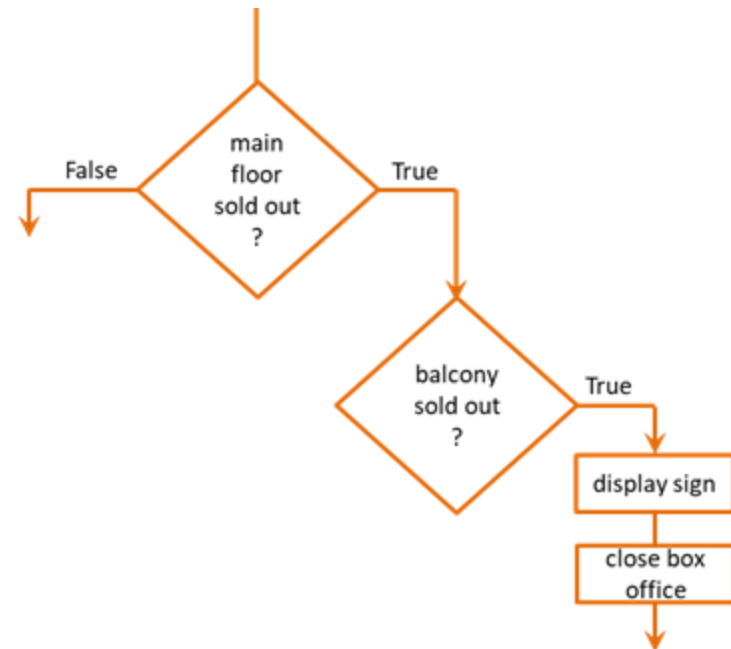
Chapter 4 Decisions and Boolean Logic

- Conditional Expressions
 - A *nested if* in the Theater example:
 - Consider a balcony section with 200 seats

If the 400 main floor seats are sold

If the 200 balcony seats are sold

- Display the "Sold Out" sign
- Close the box office



Chapter 4 Decisions and Boolean Logic

- An *if-elif-else* provides additional paths
 - The logic only tests a condition if the condition before it is false
 - The third condition is only tested if the first and second conditions are false

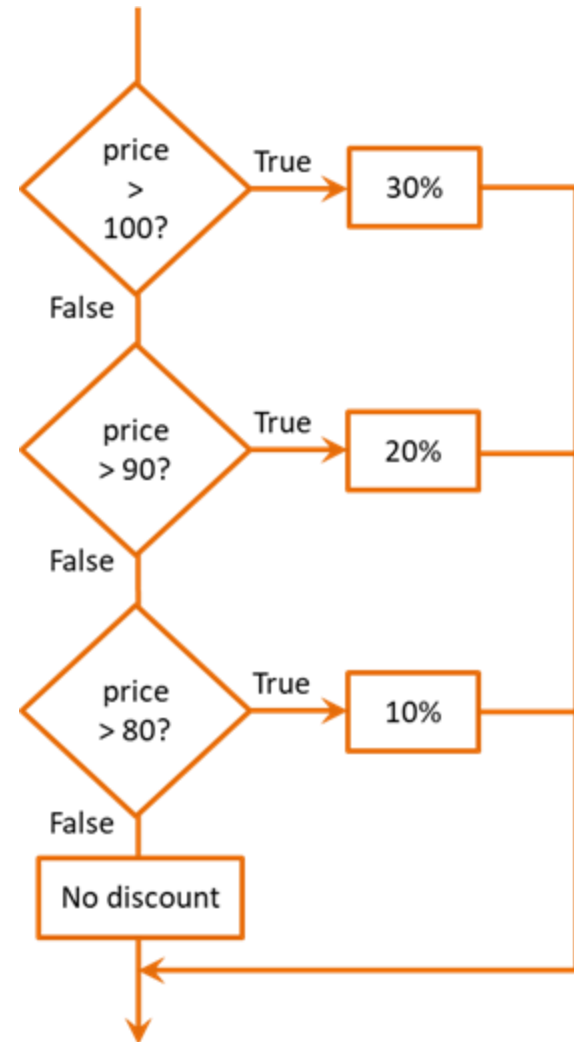
```
if → If the price > $100.00
    Discount is "30%"
elif → Otherwise-if the price > $90.00
    Discount is "20%"
elif → Otherwise-if the price is > $80.00
    Discount is "10%"
else → Otherwise
    No discount
```

Chapter 4 Decisions and Boolean Logic

- Conditional Expressions

- An *if-elif-else*

```
if price > 100:  
    discount = '30%'  
elif price > 90:  
    discount = '20%'  
elif price > 80:  
    discount = '10%'  
else:  
    discount = 'No discount'
```



Chapter 4 Decisions and Boolean Logic

Conditional Expressions and Boolean Logic

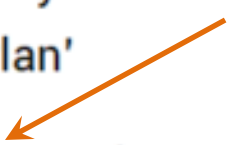
Chapter 4 Decisions and Boolean Logic

- Conditional Expressions - Comparing strings
 - Use the *equivalence operator*

```
word1 = 'Play'
word2 = 'Plan'

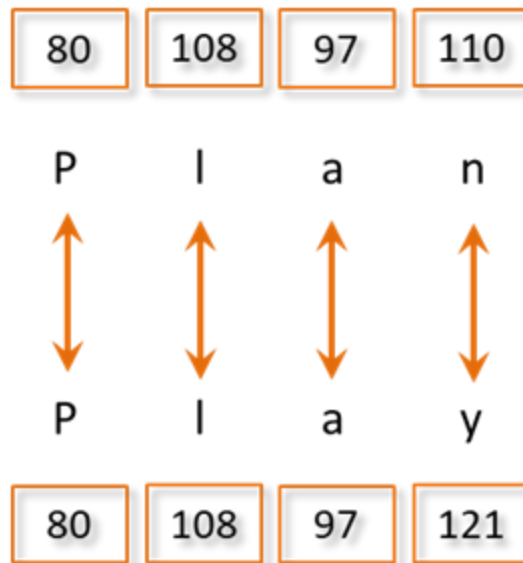
if word1 == word2:
    print('The words match')
else:
    print('They don't match')
```

tests for equivalence



Chapter 4 Decisions and Boolean Logic

- Conditional Expressions – Comparing Strings
 - Compared using the ASCII representation



Chapter 4 Decisions and Boolean Logic

- Compound Boolean Expressions
 - Multiple conditions can be combined using the ‘and’ and ‘or’ Logical Operators
 - For an expression to be true:
 - Logical *and*, both conditions must be true
 - Logical *or*, either condition must be true
 - Logical *not* operator negates the result

Chapter 4 Decisions and Boolean Logic

- Logical *and* operator
 - Both conditions must be true for the expression to be true

A	B	A and B
True	True	True
True	False	False
False	True	False
False	False	False

Chapter 4 Decisions and Boolean Logic

- Logical *and* operator
 - Both conditions must be true for the expression to be true

```
if condition1 and condition2:  
    statement1  
    statement2  
    etc.
```

Chapter 4 Decisions and Boolean Logic

- Logical *and* operator
 - Testing within a range
 - Useful for input validation

```
num = int(input('Enter a number between 1 and 100: '))  
  
if num > 0 and num <= 100:  
    print('That is a good number')
```

Validating input within a range of numbers

Chapter 4 Decisions and Boolean Logic

- Compound Boolean Expressions
 - The Theater example continued:

If the 400 main floor seats are sold

If the 200 balcony seats are sold

- Display the "Sold Out" sign
- Close the box office

```
if main_tickets_sold == 400 and balcony_tickets_sold == 200:  
    print('Display the "Sold Out" sign')  
    print('Close the box office')
```


Chapter 4 Decisions and Boolean Logic

- Logical *or* operator
 - If either condition is true, the expression is true

A	B	A or B
True	True	True
True	False	True
False	True	True
False	False	False

Chapter 4 Decisions and Boolean Logic

- Logical *or* operator
 - If either condition is true, the expression is true

```
if condition1 or condition2:  
    statement1  
    statement2  
    etc.
```

Chapter 4 Decisions and Boolean Logic

- Logical **or** operator
 - Testing a range
 - Useful for input validation

```
num = int(input('Enter a number between 1 and 100: '))  
  
if num < 1 or num > 100:  
    print('That is NOT a good number')
```

Chapter 4 Decisions and Boolean Logic

- Computers use what is called *short-circuit evaluation*
 - For a Logical “and” to be true, both sides of the compound condition must be true
 - If the left side is false, then the right side is not evaluated
 - It wouldn’t matter if the right side were true since the expression is already false

Chapter 4 Decisions and Boolean Logic

- Computers use what is called *short-circuit evaluation*
 - For a Logical “or” to be true, either side of the compound condition can be true
 - If the left side of the compound condition is true, the right side is not evaluated
 - It wouldn’t matter whether the right side is true or false since the expression is already true

Chapter 4 Decisions and Boolean Logic

- Common Logic Errors
 - Conditional statements require careful consideration

Expression	True when the value of x is:
if $x > 0$ and $x < 100$	any number 1 thru 99
if $x \geq 0$ and $x \leq 100$	any number 0 thru 100
if $x > 0$ or $x < 100$	any number
if $x < 0$ or $x > 100$	negative or above 100
if $x \leq 0$ or $x \geq 100$	negative, zero, or 100 or above

Chapter 4 Decisions and Boolean Logic

- Logical *not*
 - Returns the reverse of the condition

A	not A
True	False
False	True

- Often cause confusion and hard-to-find bugs

if not (x > y and x > z):

Chapter 4 Decisions and Boolean Logic

- Compound Boolean Expressions
 - Logical *not*
 - Often easier to reverse the logic
 - De Morgan's Law provides two forms: one for negation of an “and” expression and one for an “or” expression

$!(A \text{ and } B)$

$!A \text{ or } !B$

$!(A \text{ or } B)$

$!A \text{ and } !B$

Chapter 4 Decisions and Boolean Logic

- Boolean Variables
 - the *bool* data type in Python
 - Operates as true or false
 - Often used as flags or signals in code when something has occurred or a condition has been met
 - Can be used in conditional statements

Chapter 4 Decisions and Boolean Logic

- Boolean Variables

- The example below sets a Boolean variable to false, then changes it to true when enough tickets have been sold

```
sold_out = False

if main_sold == 400 and balcony_sold == 200:
    sold_out = True

if sold_out == True:
    print('Display the "Sold Out" sign')
    print('Close the box office')
```

Chapter 4 Decisions and Boolean Logic

- Boolean Variables

- The equivalence operator and true, is not required

This: if sold_out == True:

Is the same as: if sold_out:

- The logic can also be reversed

This: if sold_out == False:

Is the same as: if not sold_out:

Chapter 4 Decisions and Boolean Logic

- Common Errors

- Writing conditional expressions

- Always test around the threshold

if value < 90

excludes 90

if value <= 90

includes 90

- Use two (2) equal signs to test for equivalence

value = num

assigns num to value

value == num

tests for equivalence

Chapter 4 Decisions and Boolean Logic

- Common Errors
 - Writing conditional expressions

```
if value < 0 and value > 10:  
    print('This will never be true.')
```

```
if value > 0 or value < 10:  
    print('This will always be true.')
```

```
if value > 0 and value > 10 or value < 20:  
    print('Ambiguous, and always true.')
```

Chapter 4 Decisions and Boolean Logic

- Common Errors

- Not indenting correctly

- The interpreter relies on indentation to associate lines of code

```
if value < 90:  
    print('I am associated with the IF clause.')print('I am not associated with the IF clause.')
```

Chapter 4 Decisions and Boolean Logic

- Programmer Choice
 - Each of these accomplish the same thing

```
if number > 0:  
    if number < 101:  
        print('Number is valid.')
```

```
if number > 0 and number <= 100:  
    print('Number is valid.')
```

```
if number <= 0 or number > 100:  
    print('Number is not valid.')
```

```
else:  
    print('Number is valid.')
```

Chapter 4 Decisions and Boolean Logic

- **Avoiding Common Errors**
 - The interpreter will catch errors in syntax and grammar
 - Logic errors can only be avoided by careful implementation of the code
 - The best solution may be more deliberate and intentional than the use of complex statements
 - It is also a fact that code is written once, but read many times, so readability is always a consideration

Chapter 4 Decisions and Boolean Logic

Chapter 4 Decisions and Boolean Logic