Chapter Three

PART TWO: NESTED BRANCHES, MULTIPLE ALTERNATIVES, AND FLOWCHARTS

Code Academy

Chapter 3: Part 2

Goals

- To write nested if statement
- To write multiple alternatives
- To develop solution using Flowcharts

Contents

- Nested Branches
- Multiple Alternatives
- Problem Solving: Flowcharts

Nested Branches

Code Academy

- You can *nest* an if inside a branch of another if statement.

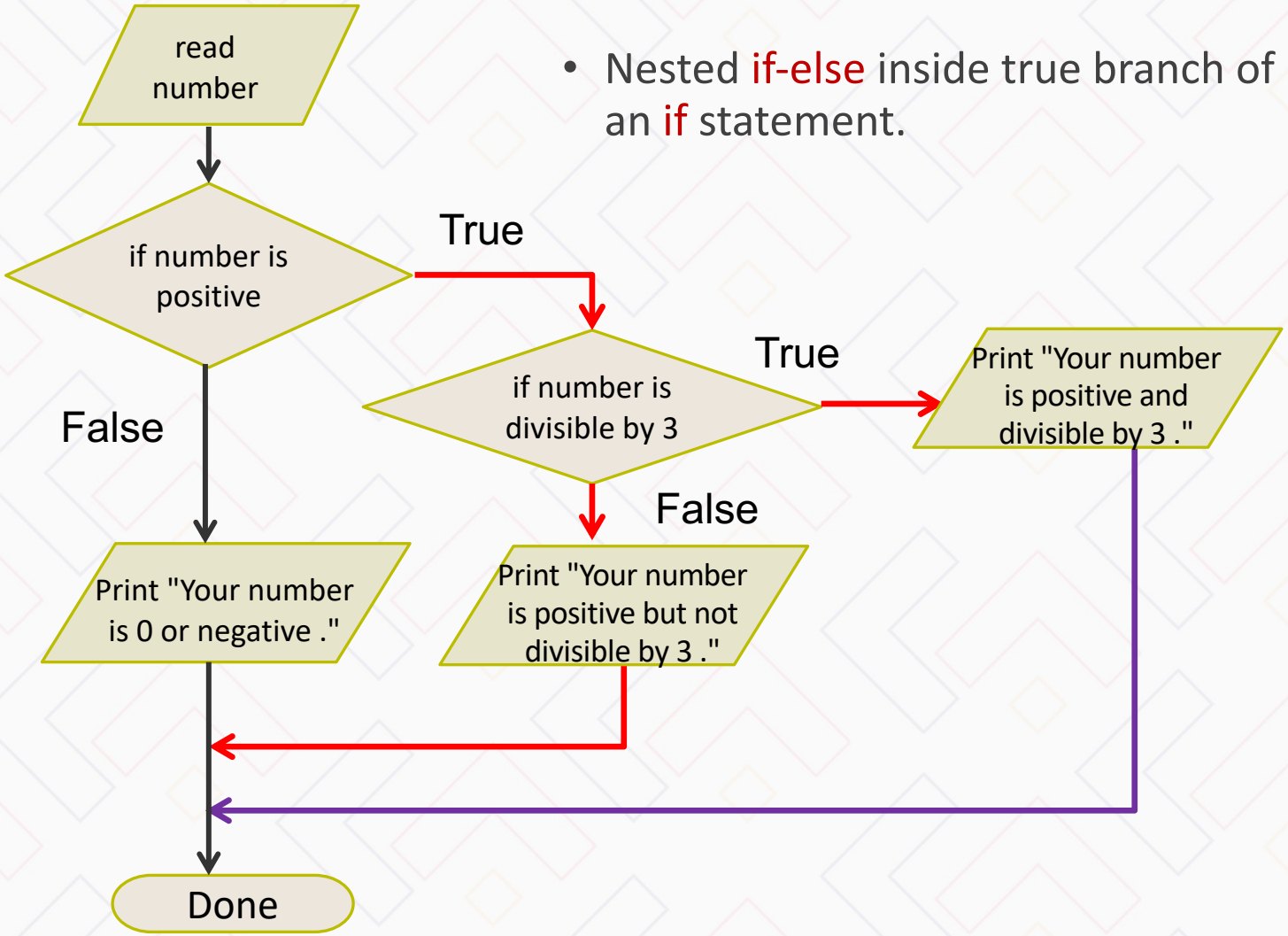
- Simple example: check if a **number is positive** and **divisible by 3**
- **Algorithm:**

```
read number
if number is positive
        if number is divisible by 3
            print "Your number is positive and divisible by 3 ."
        else
            print "Your number is positive but not divisible by 3 ."
else
        print "Your number is zero or negative."
```

Code Academy

Flowchart of a Nested if

- Nested if-else inside true branch of an if statement.



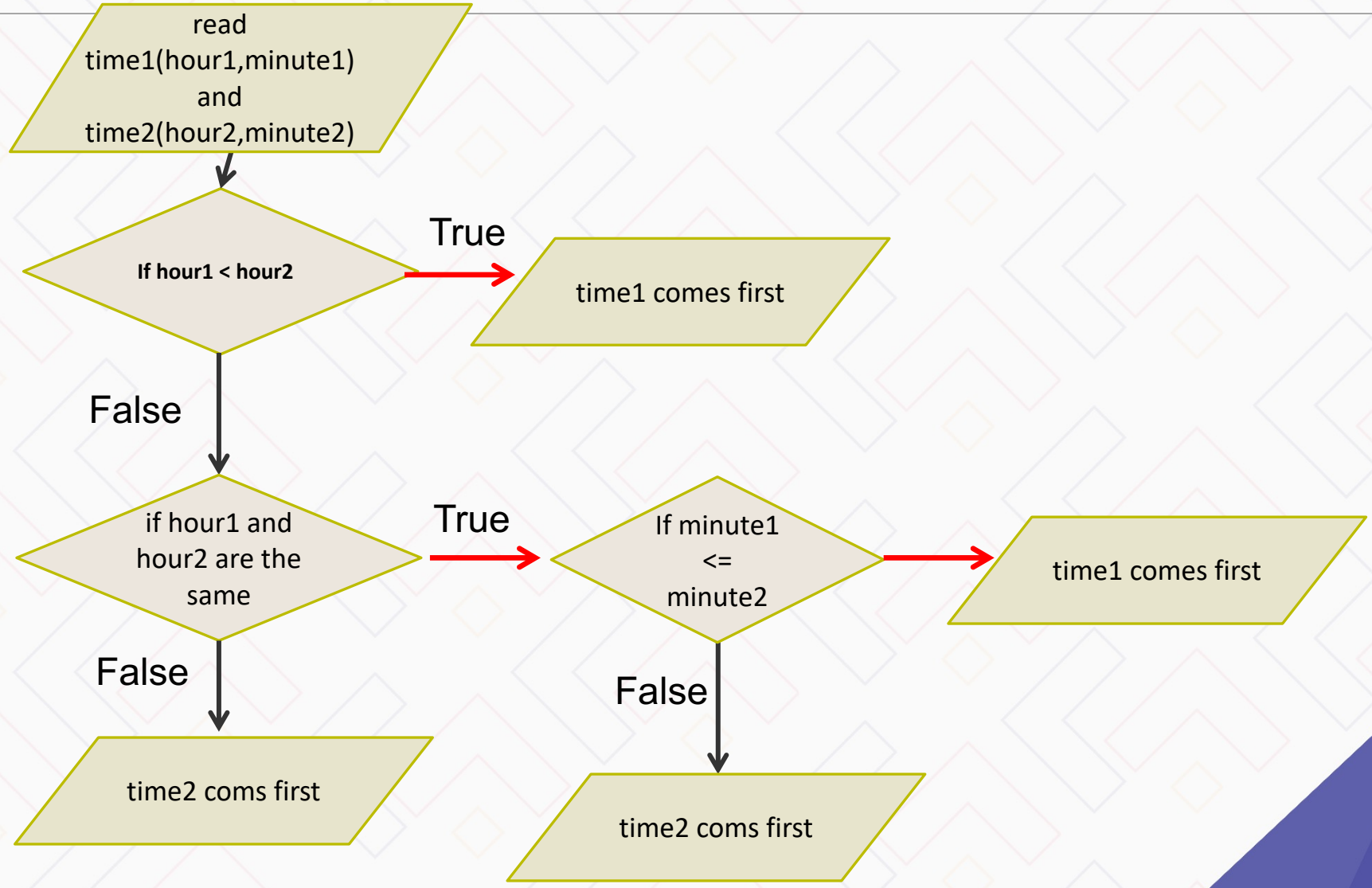
Times Example: nested `if`s

When two points in time are compared, each given as hours (in military time, ranging from 0 to 23) and minutes, the following pseudocode determines which comes first.

```
If hour1 < hour2
        time1 comes first.
Else
        if hour1 and hour2 are the same
                If minute1 <= minute2
                        time1 comes first.
                Else
                        time2 comes first.
        Else
                time2 comes first.
```

Code Academy

Flowchart for the Time Example



Code Academy

Times.py (1)

```
1  #  Determine which of two times comes first.
2
3
4  # Read the times from the user.
5  hour1 = int(input("Enter the hour for the first time: "))
6  minute1 = int(input("Enter the minute for the first time: "))
7
8  hour2 = int(input("Enter the hour for the second time: "))
9  minute2 = int(input("Enter the minute for the second time: "))
10
11 # Determine which time comes first.
12 if hour1 < hour2 :
13    first_hour = hour1
14    first_minute = minute1
15    second_hour = hour2
16    second_minute = minute2
17 else:
18     if hour1 == hour2 :
19         if minute1 <= minute2 :
20             first_hour = hour1
21             first_minute = minute1
22             second_hour = hour2
23             second_minute = minute2
24         else:
25             first_hour = hour2
26             first_minute = minute2
27             second_hour = hour1
28             second_minute = minute1
29     else :
30        first_hour = hour2
31        first_minute = minute2
32        second_hour = hour1
33        second_minute = minute1
34 # Display the result.
35 print("The first time is %02d:%02d" % (first_hour, first_minute))
36 print("The second time is %02d:%02d" % (second_hour, second_minute))
```

Times.py (2)

Code Academy

- The ‘True’ branch (Single)

```
11  # Determine which time comes first.
12  if hour1 < hour2 :
13      first_hour = hour1
14      first_minute = minute1
15      second_hour = hour2
16      second_minute = minute2
```

Times.py (3)

Code Academy

- The 'False' branch: nested if inside the outer else

```
17  else:
18      if hour1 == hour2 :
19          if minute1 <= minute2 :
20              first_hour = hour1
21              first_minute = minute1
22              second_hour = hour2
23              second_minute = minute2
24          else:
25              first_hour = hour2
26              first_minute = minute2
27              second_hour = hour1
28              second_minute = minute1
29      else :
30          first_hour = hour2
31          first_minute = minute2
32          second_hour = hour1
33          second_minute = minute1
```

Running the Times Example

- Run the program several times using different values for time1 and and time2
  - Use hour1 less than hour2
  - Use hour1 greater than hour2
  - Use hour1 is the same as hour2 but the minutes are different.
- What results do you get?

Code Academy

Hand-tracing

- Hand-tracing helps you understand whether a program works correctly
- Create a table of key variables
  - Use pencil and paper to track their values
- Works with pseudocode or code
  - Track location with a marker
- Use example input values that:
  - You know what the time should come the first
  - Will test each branch of your code

Hand-tracing the Times Example

- Setup
  - Table of variables
- Input variables
  - From user
  - Update table

```
4  # Read the times from the user.
5  hour1 = int(input("Enter the hour for the first time: "))
6  minute1 = int(input("Enter the minute for the first time: "))
7
8  hour2 = int(input("Enter the hour for the second time: "))
9  minute2 = int(input("Enter the minute for the second time: "))
```

hour1	minute1	hour2	minute2	first_hour	first_minute	second_hour	second_minute
3	5	23	45				

Hand-tracing the Times Example (2)

- Because hour1 < hour2, the else part is skipped and the below values are updated.

```
11  # Determine which time comes first.
12  if hour1 < hour2 :
13      first_hour = hour1
14      first_minute = minute1
15      second_hour = hour2
16      second_minute = minute2
```

hour1	minute1	hour2	minute2	first_hour	first_minute	second_hour	second_minute
3	25	5	45	3	25	5	45

The output:

The first time is 03:25

The second time is 05:45

Hand-tracing the Times Example (3)

- If the input is the below then the outer if is skipped.

hour1	minute1	hour2	minute2	first_hour	first_minute	second_hour	second_minute
8	36	6	15	6	15	8	36

- The below part is executed in if block.

```
29      else :
30          first_hour = hour2
31          first_minute = minute2
32          second_hour = hour1
33          second_minute = minute1
```

The output:
The first time is 06:15
The second time is 08:36

Hand-tracing the Times Example (4)

- Try if the input is the below.

hour1	minute1	hour2	minute2	first_hour	first_minute	second_hour	second_minute
7	13	7	45	?	?	?	?

- Which part is going to be executed?
- What is the updated values?
- What is the output?

3.4 Multiple Alternatives

- What if you have more than two branches?
- Count the branches for the following earthquake effect example:
  - 8 (or greater)
  - 7 to 7.99
  - 6 to 6.99
  - 4.5 to 5.99
  - Less than 4.5

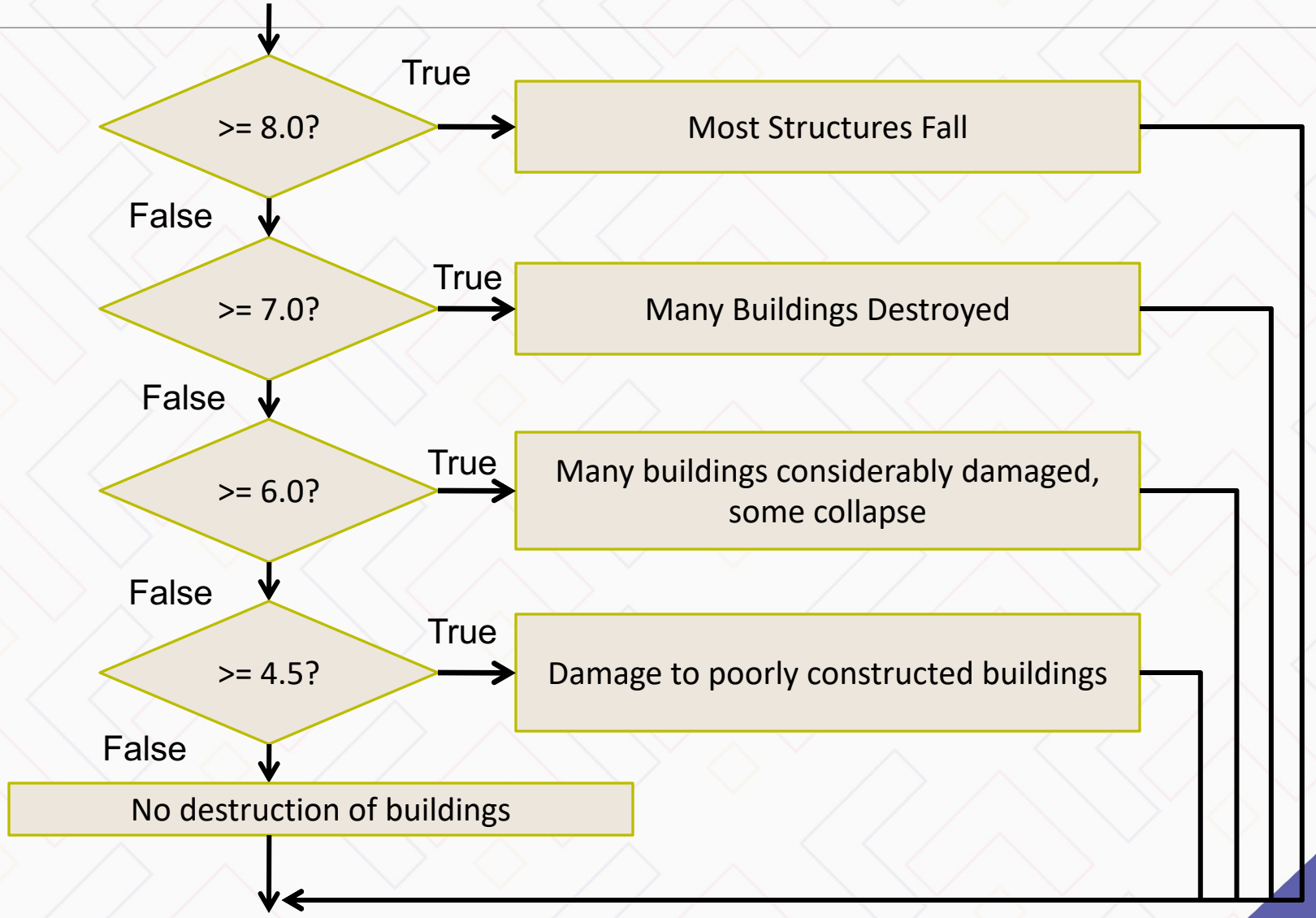
When using multiple `if` statements, test the general conditions after the more specific conditions.

Table 4 Richter Scale

Value	Effect
8	Most structures fall
7	Many buildings destroyed
6	Many buildings considerably damaged, some collapse
4.5	Damage to poorly constructed buildings

Code Academy

Flowchart of Multiway Branching



"elif" Statement

- `elif` Short for Else if…
- As soon as one on the test conditions succeeds, the statement block is executed
  - No other tests are attempted
- If none of the test conditions succeed the final else clause is executed

Code Academy

if, elif Multiway Branching

```
if richter >= 8.0 :   # Handle the ‘special case’ first
   print("Most structures fall")
elif richter >= 7.0 :
   print("Many buildings destroyed")
elif richter >= 6.0 :
   print("Many buildings damaged, some collapse")
elif richter >= 4.5 :
   print("Damage to poorly constructed buildings")
else :    # so that the ‘general case’ can be handled last
   print("No destruction of buildings")
```

earthquake Example:earthquake.py

```
1  ##
2  #  This program prints a description of an earthquake, given the Richter scale magnitude.
3  #
4
5  # Obtain the user input.
6  richter = float(input("Enter a magnitude on the Richter scale: "))
7
8  # Print the description
9  if richter >= 8.0 :
10    print("Most structures fall")
11 elif richter >= 7.0 :
12    print("Many buildings destroyed")
13 elif richter >= 6.0 :
14    print("Many buildings considerably damaged, some collapse")
15 elif richter >= 4.5 :
16    print("Damage to poorly constructed buildings")
17 else :
18    print("No destruction of buildings")
19
20
```

Run the program with several different inputs

Code Academy

What is Wrong With This Code?

```
if richter >= 8.0 :
   print("Most structures fall")
if richter >= 7.0 :
   print("Many buildings destroyed")
if richter >= 6.0 :
   print("Many buildings damaged, some collapse")
if richter >= 4.5 :
   print("Damage to poorly constructed buildings")
```

Assume richter = 5.9, what is the output?

Code Academy

Using Flowcharts to Develop and Refine Algorithms

3.5 Problem Solving: Flowcharts

- You have seen a few basic flowcharts
- A flowchart shows the structure of decisions and tasks to solve a problem
- Basic flowchart elements:

- Connect them with arrows

Each branch of a decision can contain tasks and further decisions

Code Academy

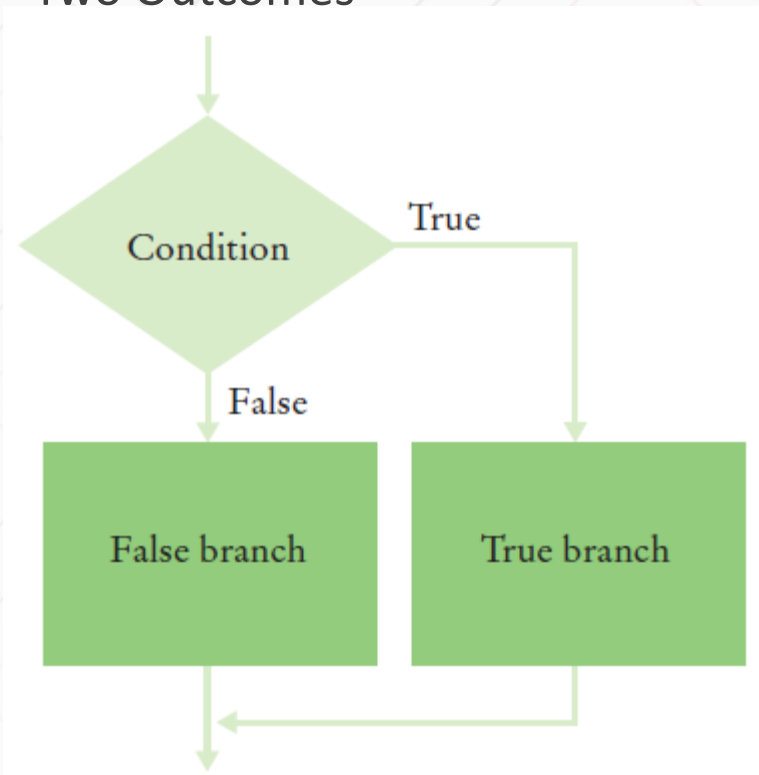
Using Flowcharts

- Flowcharts are an excellent tool to help you visualize the flow of your algorithm
- Building the flowchart
  - Link your tasks and input / output boxes in the sequence they need to be executed
  - When you need to make a decision use the diamond (a conditional statement) with two outcomes
  - Never point an arrow inside another branch

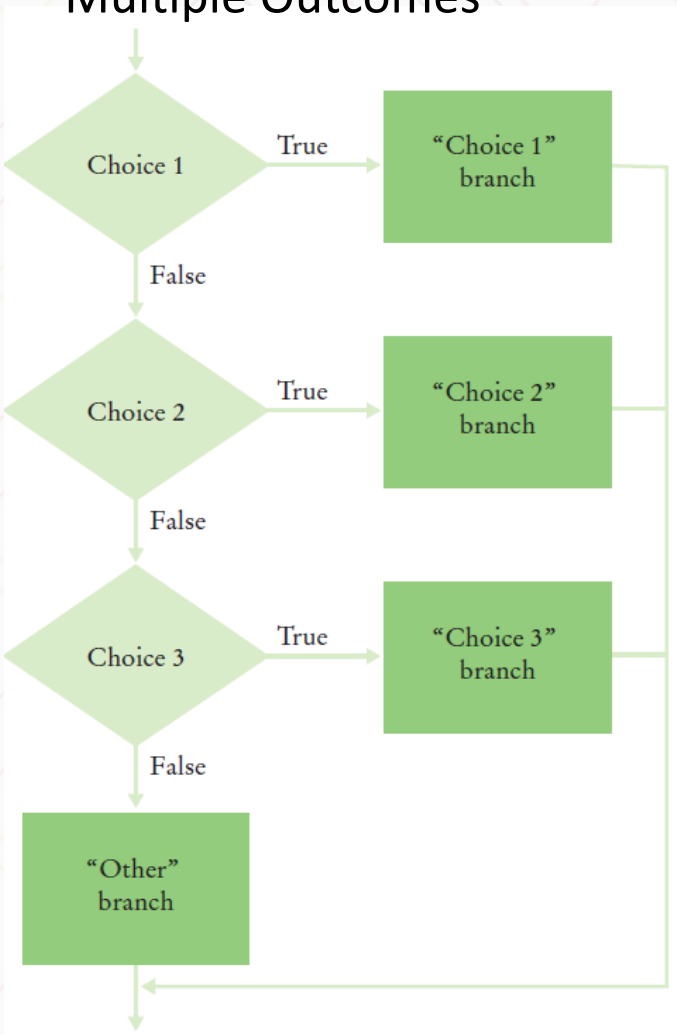
Conditional Flowcharts

Code Academy

Two Outcomes



Multiple Outcomes

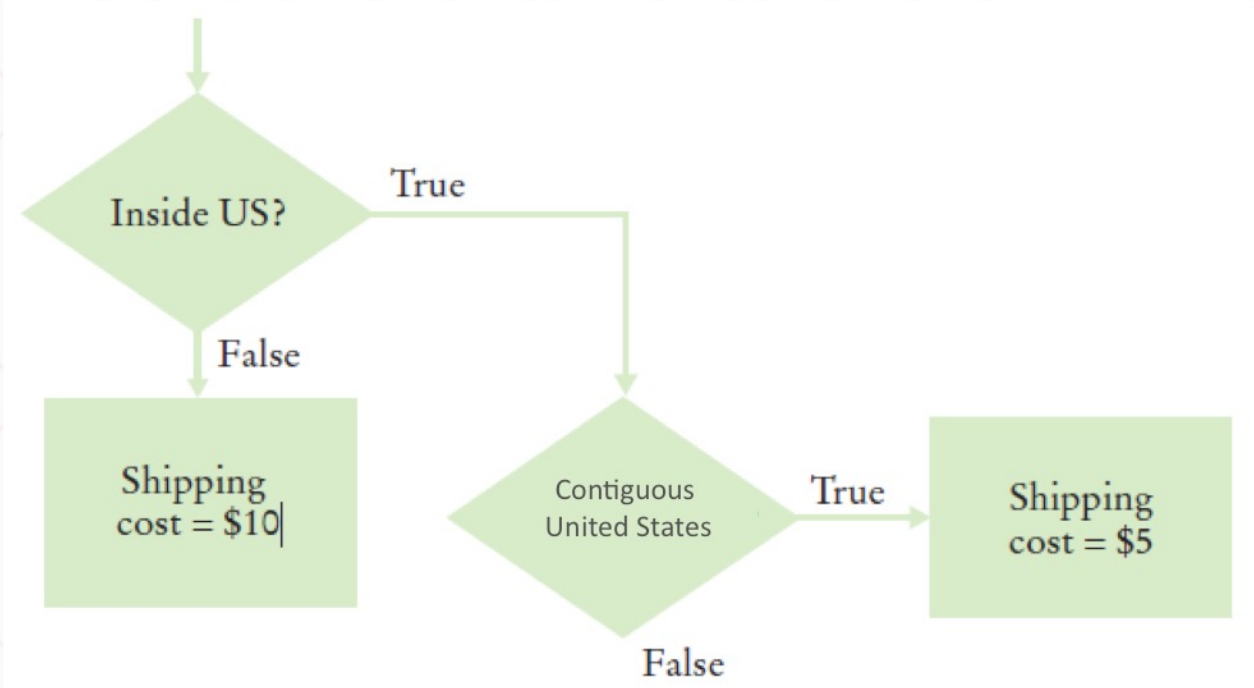


Shipping Cost flowchart

Code Academy

Shipping costs are $5 inside the contiguous the United States (Lower 48 states), and $10 to Hawaii and Alaska. International shipping costs are also $10.

- Three Branches:



International Branch

Hawaii/Alaska Branch

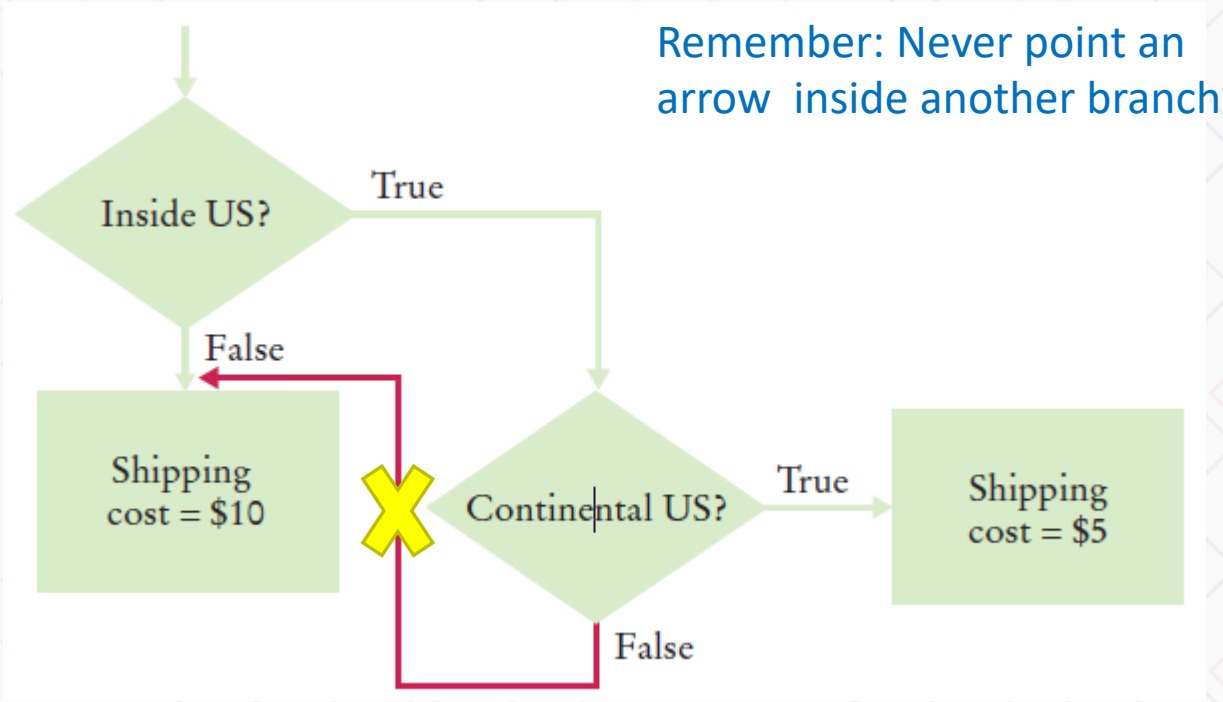
Lower 48 Branch

Don’t Connect Branches!

Code Academy

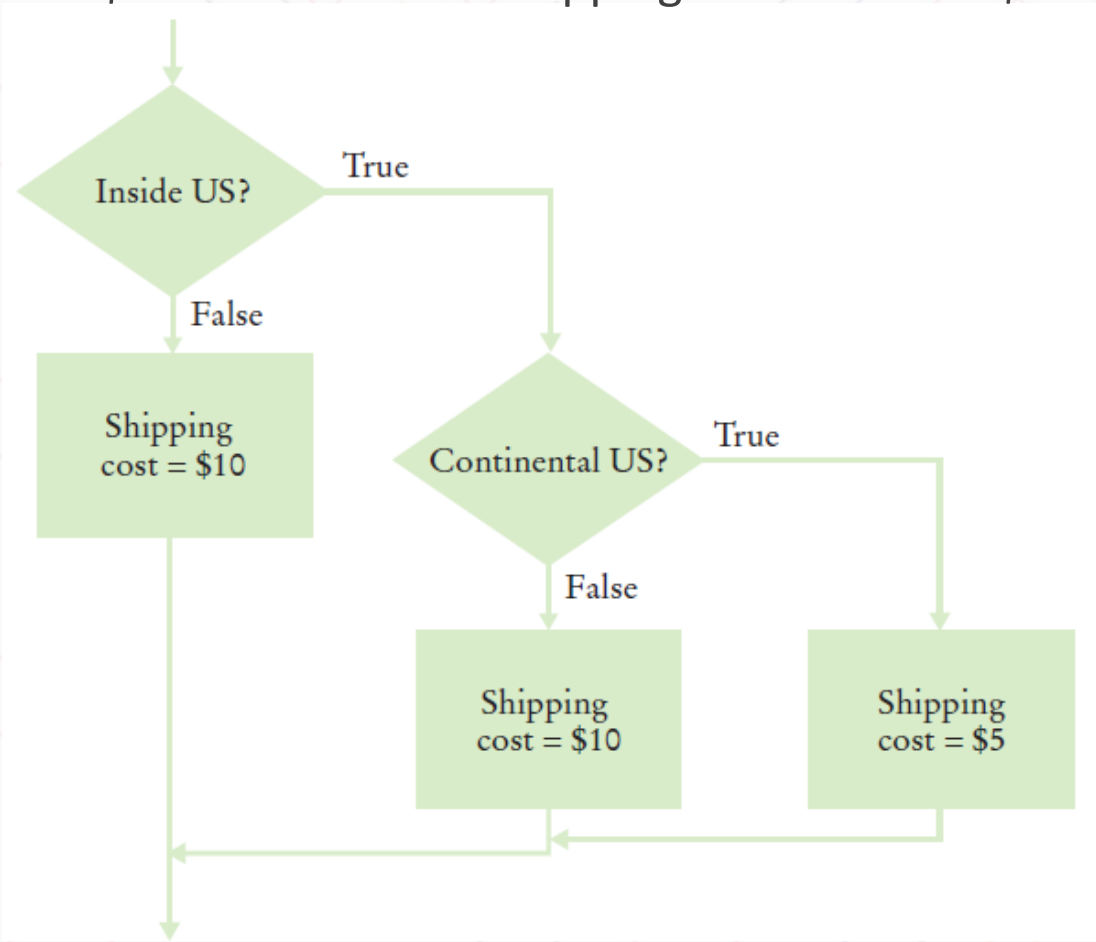
Shipping costs are $5 inside the United States, except that to Hawaii and Alaska they are $10. International shipping costs are also $10.

- Don’t do this!



Shipping Cost Flowchart

Shipping costs are $5 inside the United States, except that to Hawaii and Alaska they are $10. International shipping costs are also $10.



Shipping Example: shipping.py

```
1  ##
2  #  A program to compute shipping costs.
3  #
4
5  # Obtain the user input.
6  country = input("Enter the country: ")
7  state = input("Enter the state or province: ")
8
9  # Compute the shipping cost.
10 shippingCost = 0.0
11
12 if country == "USA" :
13    if state == "AK" or state == "HI" :  # See Section 3.7 for the or operator
14       shippingCost = 10.0
15    else :
16       shippingCost = 5.0
17 else :
18    shippingCost = 10.0
19
20 # Print the results.
21 print("Shipping cost to %s, %s: $%.2f" % (state, country, shippingCost))
22
```

- Run the program with several different inputs?
  - What happens if you enter “usa” as the country?
- We will learn several ways to correct the code later in this chapter

Problem Solving: Test Cases

- Aim for complete coverage of all decision points:
- Consider times.py program.

```
#  Determine which of two times comes first.


# Read the times from the user.
hour1 = int(input("Enter the hour for the first time: "))
minute1 = int(input("Enter the minute for the first time: "))

hour2 = int(input("Enter the hour for the second time: "))
minute2 = int(input("Enter the minute for the second time: "))

# Determine which time comes first.
if hour1 < hour2 :
   first_hour = hour1
   first_minute = minute1
   second_hour = hour2
   second_minute = minute2
else:
    if hour1 == hour2 :
        if minute1 <= minute2 :
            first_hour = hour1
            first_minute = minute1
            second_hour = hour2
            second_minute = minute2
        else:
            first_hour = hour2
            first_minute = minute2
            second_hour = hour1
            second_minute = minute1
    else :
       first_hour = hour2
       first_minute = minute2
       second_hour = hour1
       second_minute = minute1
# Display the result.
print("The first time is %02d:%02d" % (first_hour, first_minute))
print("The second time is %02d:%02d" % (second_hour, second_minute))
```

Problem Solving: Test Cases of taxes.py

- There are 4 possibilities for comparing times, yielding four test cases
- Test a handful of boundary conditions.
- If you are responsible for error checking (which is discussed in Section 3.9), also test an invalid input, such as a negative time or larger than 23.
- Each branch of your code should be covered with a test case

Choosing Test Cases

- Choose input values that:
  - Test boundary cases and 0 values
  - Test each branch

Make a Schedule…

- Make a reasonable estimate of the time it will take you to:
  - Design the algorithm
  - Develop test cases
  - Translate the algorithm to code and enter the code
  - Test and debug your program
- Leave some extra time for unanticipated problems

As you gain more experience your estimates will become more accurate. It is better to have some extra time than to be late

Summary: If Statements, Flow charts, Testing,

- If Statements
  - Multiple `if` statements can be combined to evaluate complex decisions.
  - When using multiple `if` statements, test general conditions after more specific conditions.
- Flow charts
  - Flow charts are made up of elements for tasks, input/output, and decisions.
  - Each branch of a decision can contain tasks and further decisions.
  - Never point an arrow inside another branch.
- Testing
  - Each branch of your program should be covered by a test case.
  - It is a good idea to design test cases before implementing a program.