Introduction to:
Computers & Programming:
Booleans, Conditionals and Loops:
Flow of Control in Python

Adam Meyers

New York University

Outline

- What is flow of control?
- Order of statements
  - Within a function
  - Functions within functions
- Boolean Data Type
- Logical Operators
- Conditional Statements
  - Conditional Keywords: if, else, elif
  - Application: Decision Trees
- Loops: Next Topic

Flow of Control

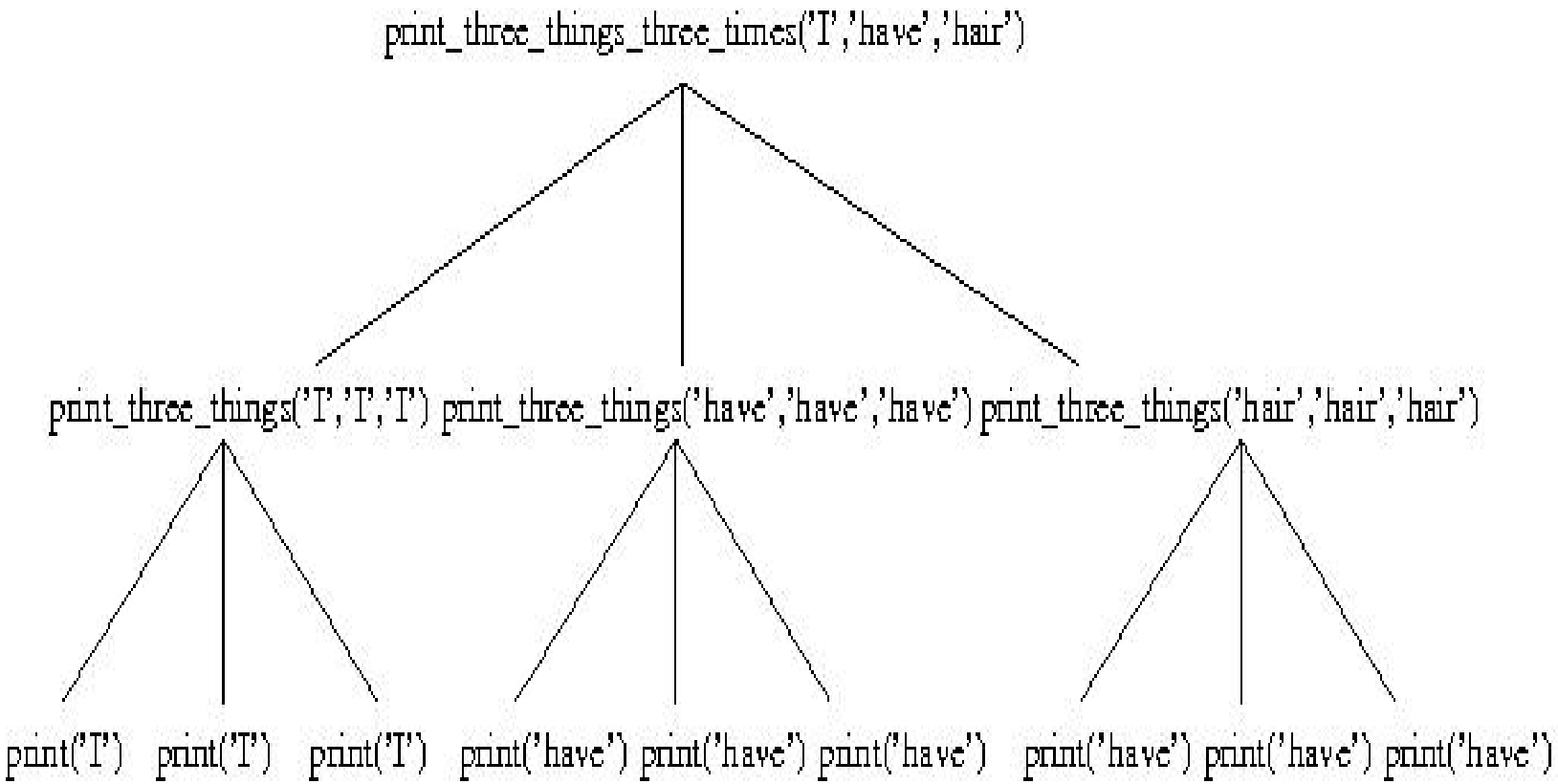
- The determination of when and if
  - instructions execute, functions are called, variables are set, output is returned, etc.
- Simple cases
  - Within a block, instructions execute top to bottom

```
def print_three_things(thing1, thing2, thing3):
    print(thing1)
    print(thing2)
    print(thing3)
```

  - Nested blocks are also executed in order

```
def print_three_things_three_times(thing1, thing2, thing3):
    print_three_things(thing1, thing1, thing1)
    print_three_things(thing2, thing2, thing2)
    print_three_things(thing3, thing3, thing3)
```

Example of Simple Flow of Control



Flow and Control and Boolean Values

- Sometimes a statement will only execute
  - **if** some expression evaluates to **True** or **False**
- ***while*** loops keep repeating a block of text until
  - A particular expression evaluates to **True** or **False**
- Other types of flow of control may also depend on **True** or **False** statements
- So first, we will describe the nature of Boolean Expressions, expressions which evaluate as **True** or **False**

Boolean Data Type and Logical Operators

- There are two objects of type ***Boolean***: ***True*** & ***False***
- Logical operators – operators which output Boolean values

Operator	Arguments	Definition
p == q	p and q are of any type	True iff p and q are equal
p != q	p and q are of any type	False iff p and q are equal
not p	p is of type boolean	True iff p is False
p and q	p and q are of type boolean	True iff both p and q are True
p or q	p and q are of type boolean	True if p is True or q is True or both are True

– ***not*** is a unary operator (occurs before its one argument)

– ***or*** is ***inclusive or*** not ***exclusive or***

– == is a logical operator; = is the assignment operator

OR

- In English, the word ***or*** is ambiguous
  - *Are you a boy or a girl?*
    - ***Both*** would be an unusual answer
    - This kind of ***or*** is called ***exclusive or***
  - *Do you own a hair dryer or a toaster oven?*
    - ***Both*** would be a normal answer
    - This kind of ***or*** is called ***inclusive or***
  - Note: ***Either/or*** is usually exclusive or, e.g., Do you want either a hair dryer or a toaster oven?
- In python and most programming languages
  - ***or*** means ***inclusive or*** only
- However, we can define ***exclusive or***
  - def xor (p, q):

    return((p or q) and (not (p and q)))

More Boolean Operators (Math Only)

Operator	Arguments	Definition
x < y	x and y are integers/floats	True iff x is less than y
x <= y	x and y are integers/floats	True iff x is less than or equal to y
x > y	x and y are integers/floats	True iff x is greater than y
x >= y	x and y are integers/floats	True iff x is greater or equal to y

Boolean Data Type & Logical Operators 2

- Boolean values (and therefore logical operators) are used for conditional statements in programs
  - Different statements may activate depending on whether a variable has a ***True*** or ***False*** value
  - Or some cycle will repeat until a variable has a ***True*** or ***False*** value
- Logical Operators combine Boolean values together in various ways
- Truth Tables (from propositional logic) are useful for correctly interpreting combinations of Boolean values

Truth Table for combinations of p & q

p	q	p == q	p != q	p and q	p or q	not p
False	False	True	False	False	False	True
False	True	False	True	False	True	True
True	False	False	True	False	True	False
True	True	True	False	True	True	False

Order of Precedence: ***not, and, or***

- Use parentheses to avoid ambiguity when linking more than two expressions with not/and/or
- Example ambiguity:
  - **(Not True) or False or True == True**
  - **Not (True or False or True) == False**
  - **(True and False) or ((True or True) and False) == False**
  - **(True and (False or True or (True and False))) == True**
- Precedence: parentheses, ==, !=, not, and, or
- Using parentheses is easier for humans than relying on default precedence rules
  - Ambiguity similar to situation in arithmetic

Conditionals: if and else

- These keywords divide blocks of statements
  - Based on the evaluation of Booleans as True or False
- For example, consider the following code

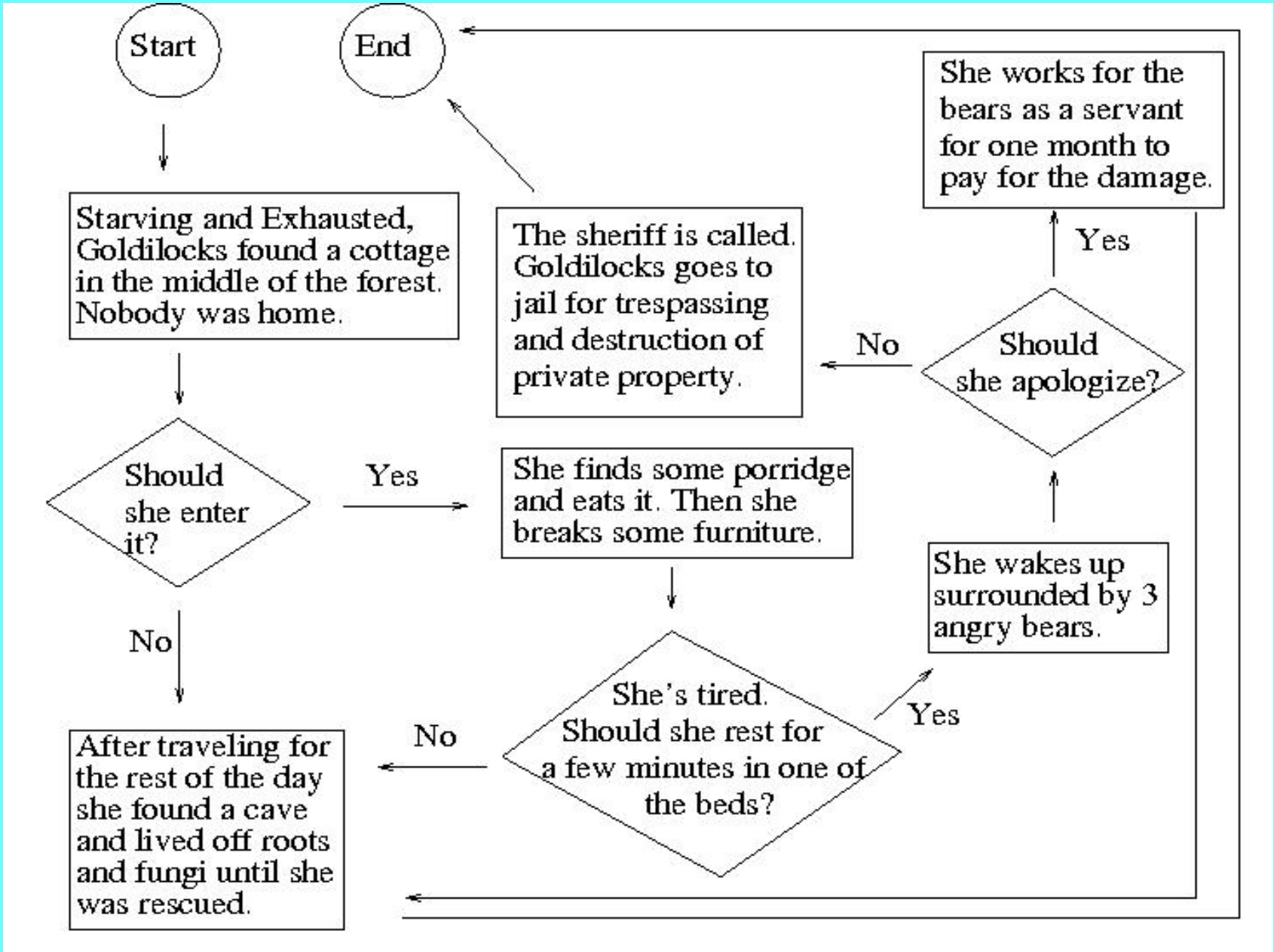
```
def is_your_name_bruce (name):
    if (name == 'Bruce' or name == 'bruce'):
        print('Your name IS Bruce!')
        return(True)
    else:
        print("Well, I guess your name isn't Bruce, now is it?")
        return(False)
```

Syntax of *if* and *else*

- ***if*** is followed by:
  - A boolean expression, a colon and a block of text
    - The block of text is indented
    - We will call the boolean expression the ***if statement***
  - The text block is executed if the boolean expression is true
    - We will call this the ***then statement***
- Optionally, ***else:*** can introduce another text block
  - this executes if the boolean expression is false.
    - We will call this the ***else statement***

Sample Application: Interactive Fiction

- The first adventure game was text based
  - It was written by W. Crowther in the 1975
  - Available for Windows: ftp://ftp.ifarchive.org/ifarchive/games/pc/adv_crowther_win.zip
  - Available for MAC: http://www.lobotomo.com/products/Adventure/index.html
  - Source code (Fortran): http://jerz.setonhill.edu/if/crowther/
  - Inspired by role playing game: Dungeons and Dragons (1974)
- Interactive version of *Goldilocks and the 3 Bears*


Start
End
She works for the bears as a servant for one month to pay for the damage.
Starving and Exhausted, Goldilocks found a cottage in the middle of the forest. Nobody was home.
The sheriff is called. Goldilocks goes to jail for trespassing and destruction of private property.
Yes
No
Should she apologize?
Should she enter it?
Yes
She finds some porridge and eats it. Then she breaks some furniture.
She wakes up surrounded by 3 angry bears.
No
She's tired. Should she rest for a few minutes in one of the beds?
Yes
No
After traveling for the rest of the day she found a cave and lived off roots and fungi until she was rescued.

Basic Idea for Program

- An interactive program
  - User answers a series of yes/no questions
  - Use the *input* function to get keyboard input

    answer = input("Type 'yes' or 'no': ")
    - Sets the variable answer to the string input by the user
- Uses ***if*** and ***else*** to divide the ***yes*** and ***no*** choices in the flowchart.
- Most of the program involves printing different sections of the text.

goldilocks.py

- The program performs as in the flowchart
  - Some minor changes in the text
- The main function
  - Uses a series of nested instances of ***if*** and ***else***
  - Calls ***get_yes_or_no*** to query if the user types ***yes***
    - Allowances are made for imprecise responses
  - Calls ***print_ending1*** to print the most common ending
  - Some *if statements* are (***not get_yes_or_no()***)
    - It is easier to read if the shorter path is listed first
      - Quick paths to the end are listed first

elif

- The following structure

```
if x:
    else:
        if y:
            else:
                if z: ....
```

- Can be abbreviated using elif (else if)

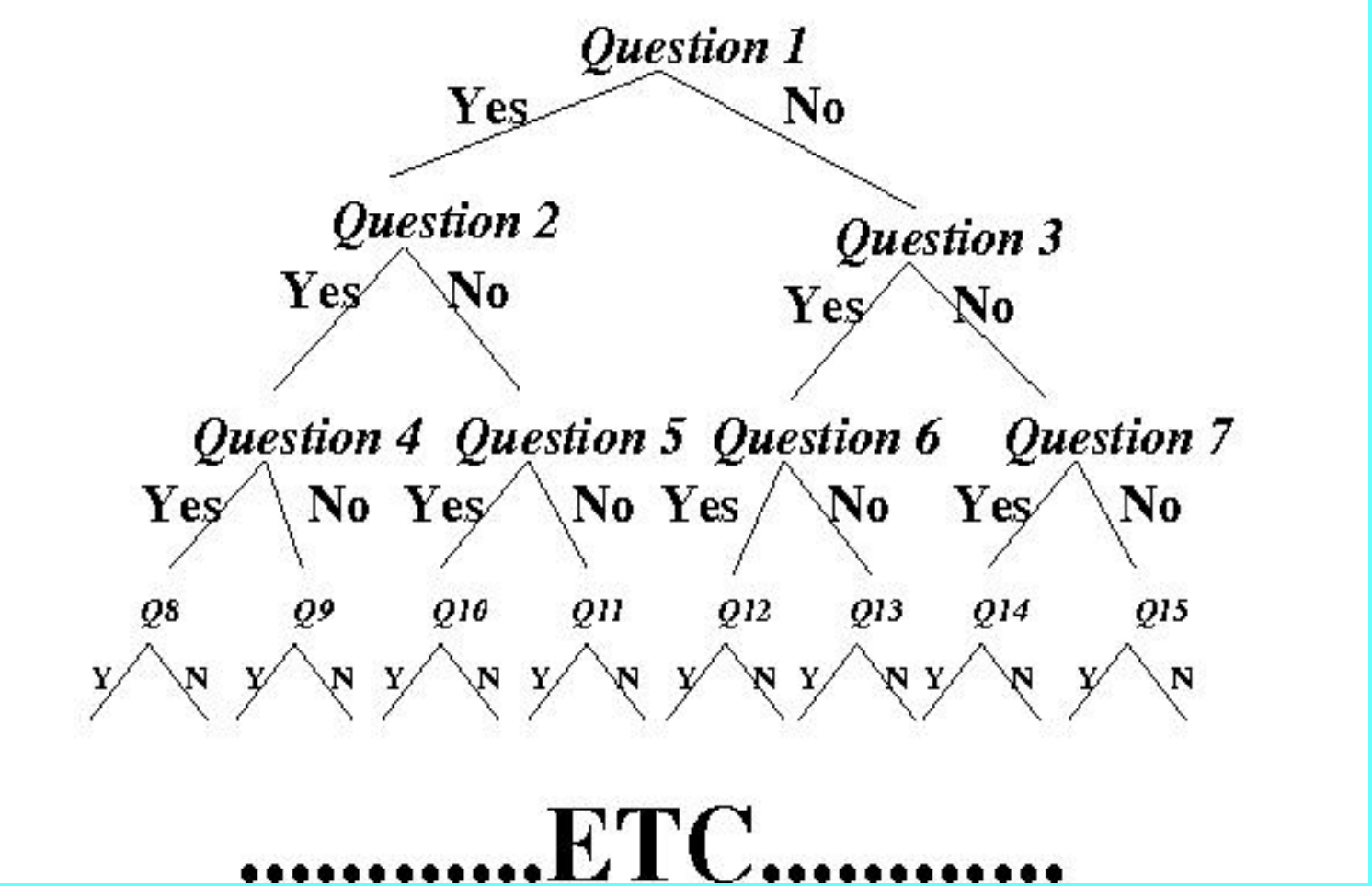
```
if x:
elif y:
elif z: ....
```

- This can make the code easier to read

Binary Branching Decision Trees

- Complex decisions can be broken down into a series of yes/no questions, forming a binary branching tree.
- The graph on the following page suggests how the flow of control can proceed in such programs.
- Programs using decision trees can have a similar structure to the goldilocks program
- Only 4 out of the 15 questions in a binary decision tree are ever asked when the function is called.
  - The system asks $\log_2 N + 1$ out of N questions
- Applications include expert systems (medical, automotive, etc.) and automatic teller machines

Binary Decision Trees



Conditionals Can Be Used to Identify Errors in User Input

- The function ***get_yes_or_no*** in the Goldilocks program
  - If the answer is yes or Yes, return True
  - Else if the answer is no or No, return False
  - Else
    - Print "your answer is unclear, but we think you mean no"
    - Return False
- Other possibilities:
  - Use while loops (coming up soon) to keep asking the user for more input until they provide well-formed input
  - Print "this is an error" and return "error"

Examples of Decision Tree Programs

- Goldilocks (previous slides)
  - Similar to: interactive fiction games, educational software, children's stories, adventure-type games
- Automatic Bank Teller Machines
- Expert Systems
- Automated Phone Systems

Example 2: Bank Teller Machine

- Flow Charts
  - http://www.cse.unl.edu/~goddard/Courses/CSCE310J/StandardHandouts/ShortUMLreference.pdf
    - Page 9
  - http://wakasmalik.blogspot.com/2010/10/atm-flowchart.html
- Flow Chart symbols: conventions seem to vary; some additional shapes
  - Circle (or Ovals) = Start/End/Continue
  - Rectangle (or Ovals) = Commands
  - Diamonds (or Vertical Bars) = Decisions
  - Parallelograms (slanted rectangles) = Input/Output

Example 3: A (toy) Expert System to Distinguish a Cold from the Flu

- Source:
  - http://www.webmd.com/cold-and-flu/cold-guide/flu-cold-symptoms
- 1st Step: Sum up all the factors involved
- 2nd Step: Model them as a decision tree, an organized series of yes/no questions
- 3rd Step: Implement them as a Python program

Table from 2nd Page of Web-MD Article

Symptom	Cold	Flu
Fever	Mild, more common in children	Usually Higher (100° to 102° F), lasts 3-4 days
Headache	Occasional	Common
Aches/Pains	Slight (implied not always)	Usual, often severe
Fatigue, weakness	Sometimes	Usual, can last 2 to 3 weeks
Extreme Exhaustion	Never	Usual, at beginning of illness
Stuffy Nose	Common	Sometimes
Sneezing	Usual	Sometimes
Sore Throat	Common	Sometimes
Chest Discomfort, Cough	Mild to moderate; hacking cough (implied not always)	Common; can become severe

Info about Colds Taken from text

- Duration:
  - Contagious for a few days
  - Symptoms last about one week
    - If more than a week, may be bacterial infection or allergic rhinitis (hay fever) – allergic reaction
- Symptom1: sore throat for 1-2 days
- Symptom2: runny nose, congestion
- Symptom3: cough (after 4-5 days)
- Variable symptom:
  - fever in children
  - possibly slight fever in adults
- Caused by several hundred different viruses
- Complications: sinus congestion; middle ear infection

Info about the Flu taken from text

- Two types of Flu: Seasonal and Swine Flu
- Symptoms: sore throat, fever, headache, muscle ache, soreness, congestion, cough
- Swine flu specific symptoms: vomiting and diarrhea
- Duration: a few days to a few weeks
- Symptoms can take a few days to a week to dissipate
- Possible complications
  - Pneumonia (possibly life threatening)
    - Symptom: shortness of breath
    - A fever that goes away and then returns after 1-2 days
  - Others: sinusitis, bronchitis, ear infection

How can we model the Information in the article in our program?

- The WebMD article is not written in formal language – we have to interpret it so we can codify it in a program
- Many symptoms indicate either cold or flu
  - Some of these have informal frequencies associated with cold and flu as indicated by words like:
    - never, occasional/sometimes/Mild/Slight, common, usual
  - We can interpret these using a point system that we share with the user, e.g.,
    - never = 0, occasional/sometimes/mild/slight = 1, common = 3, usual = 6, always = 10
  - Severity of Symptoms can be treated the same way:
    - Nonexistant = 0, mild = 1, moderate = 5, extreme = 10

More Considerations for Modeling the Problem

- The text provides clues that are not in the table
- It mentions which symptoms occur first
- It tells about symptoms specific to Swine Flu, a subtype of Flu
- Some of the questions imply human knowledge that we have to incorporate into a program.
  - A fever is a temperature that is probably at least 99ºF
  - A child is probably someone who is under a certain age
    - We are guessing that age is 16 for purposes of this program

Generalizations and Simplifications

- Assumption: These are common symptoms of many ailments. We need at least 3 symptoms before guessing that Flu or Cold is a possible diagnosis. This will prevent false diagnoses.
- We can divide all the symptoms into the following classes:
  - Symptoms that absolutely favor Flu over Cold
  - Symptoms that tend to favor Flu
  - Symptoms that tend to favor Cold
  - Symptoms that absolutely favor Swine Flu over Seasonal Flu
- We can try a voting scheme
  - We don't know if this will work, but we can test it
  - This is for demo purposes only. We won't have an extensive testing phase and must assume the program is not accurate.

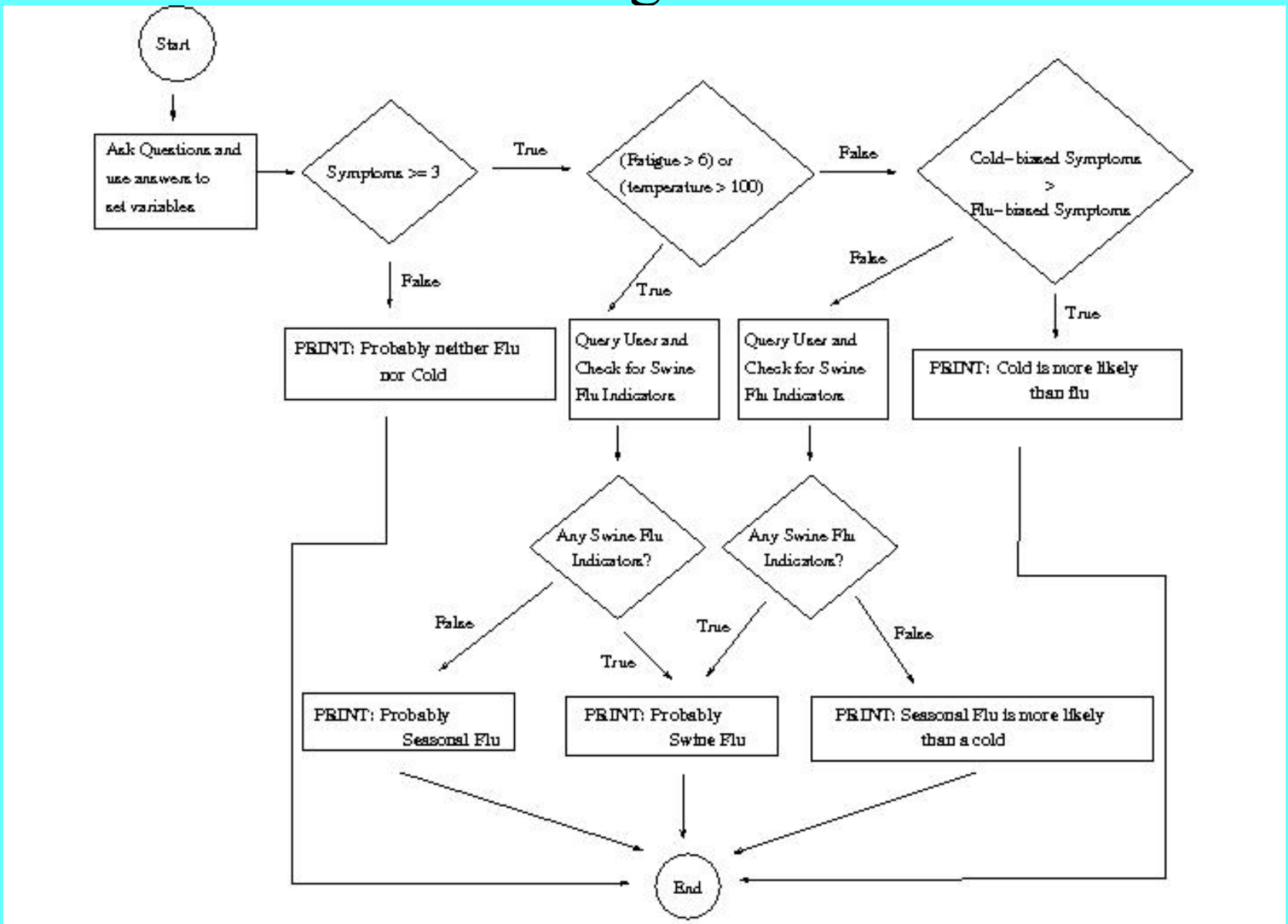
Assumptions in Our Program

- Our list of symptoms:
  - fever, tiredness, headache, fatigue, other aches and pains, chest discomfort and coughing, stuffy/runny nose, sneezing, sore throat
  - These can be true/false or have a range of values
- Symptoms absolutely favoring Flu
  - High level of fatigue or high temperature
- Symptoms favoring cold
  - low fever in children, sneezing, sore throat, stuffy/runny nose

More Assumptions in Our Program

- Symptoms favoring flu
  - headache, other aches, medium level of fatigue, coughing, illness longer than 1 week
- Symptoms absolutely favoring Swine over Seasonal Flu
  - vomiting and diarrhea
- Definition of Child for our purposes: Age < 16
- Fever: temperature >= 99
  - Low: temperature >= 99 and temperature <=100
  - High: temperature > 100

Our Algorithm



Ways We Could Improve the Program

- Include information about the sequence of symptoms.
  - Colds often begin with sore throats, which go away after a few days and are followed by nasal symptoms
- Consult other articles
- Consult a doctor
- Test the accuracy of the program on real data (real instances where we know the diagnosis and symptoms).
  - Modify the program to better account for the data
  - Test the program on new data

Implementation Details: The function *is_yes_or_no*

- Takes one argument: the question to be asked
- Uses *input* function to ask the question and retrieve answer
- Converts yes or no answer into True or False
- Assumes unexpected answers are equivalent to 'No'
  - Alternative: Give user an error and exit the program
  - Alternative: Give user an error and ask for Yes or No again
- Uses Counters: symptoms, flu_symptoms, cold_symptoms
  - These are incremented by 1 when we identify a new symptom that fits the appropriate category
    - counter = 1 + counter
  - Some of the boolean tests involve counters and boolean operators (>, <, >=, <=, ==, !=)

More Implementation Details

- *number_from_zero_to_ten*
  - Makes sure that an integer from 0 to 10 is used
  - Rounds to the nearest integer
  - There would be an error if the user entered a non-number – there will be a section on proper error handling later this term.
- *check_for_swine_flu*
  - There are exactly 2 symptoms where either is evidence of Swine Flu, provided that flu is a possibility
  - One simple yes/no qustion covers this
  - This function is separated to make it easy to revise this in the future, should additional information be incorporated in the program.

Our System vs. Real Medical Expert Systems

- We need to include a warning in this program that the diagnosis should not be taken seriously
- Before releasing a real expert system, we would test it extensively and modify it so it performs accurately.
  - There was no quality control for this program
- Doctors are consulted for real systems – systems are not based on web articles written for non-doctors
- One expert system designed by a doctor is available online at: http://easydiagnosis.com/
  - Dr. Schueler, who designed this system, also warns that this program should not take the place of a real doctor

Expert Systems

- These can be represented by decision trees
  - They attempt to model human reasoning based on the order in which a human being would ask questions.
- Of course, there are other models for automatically making the same sorts of decisions
  - For example, predictions can be based on statistical correlations
- They are used in many fields: medicine, fixing machinery, how to choose a wine, picking an airplane flight, etc.
- Information on expert systems: http://edutechwiki.unige.ch/en/Expert_system

Summary

- Flow of Control refers to the determination of when commands are executed. Factors include:
  - order of statements
  - order of the blocks containing statements
  - evaluation of boolean expressions in ***if*** & ***elif*** clauses
    - If the boolean evaluates to ***True***
      - The body of ***if/elif*** executes
    - Otherwise, the body of the following ***else*** executes (if it exists)
- Flow of Control relies on boolean operators (==, !=, not, and, or), mathematical boolean operators (<,>,<=,>=) and other functions that return boolean values.
  - Parentheses recommended (or an understanding of precedence rules)
- The ***input*** function provides a simple means of user interaction
- The **decision tree** is a simple, but powerful algorithm for problem solving: interactive fiction and expert systems are 2 common applications of the decision tree

Homework: Slide 1

- Due in 2 or 3 classes
- Read Chapter 4
- Design a program that uses a decision tree
  - Write out your decision tree and include your plan as either a separate file or a set of comments
- Write a program that implements this decision tree
  - Interactive fiction or other game
  - A questionnaire that is geared towards solving a particular kind of problem (e.g., choosing a car)
  - An expert system for solving some problem that you are an expert on
  - A system for classifying objects
- Grading criteria provided on the next slide

Homework Slide 2: Grading Criteria

- Topic: interesting? A good fit?
- Planning
  - Did you implement what you planned?
  - Is it a good plan?
- The shape and size of the tree
  - How many questions are involved?
  - How deep is the tree?
- Does the program work?
- Did you do anything innovative?
- Is your code clear and well-written?

Homework Slide 3

- This is an opportunity to think through a logical problem that you have a lot of knowledge about and structure the information as a program
- Or write a type of fiction that includes lots of variables, but that could also be carefully thought out.
- There has been a lot of previous work on both these sorts of programs.