Conditionals

Leen-Kiat Soh
Computer Science & Engineering
University of Nebraska, Lincoln, NE

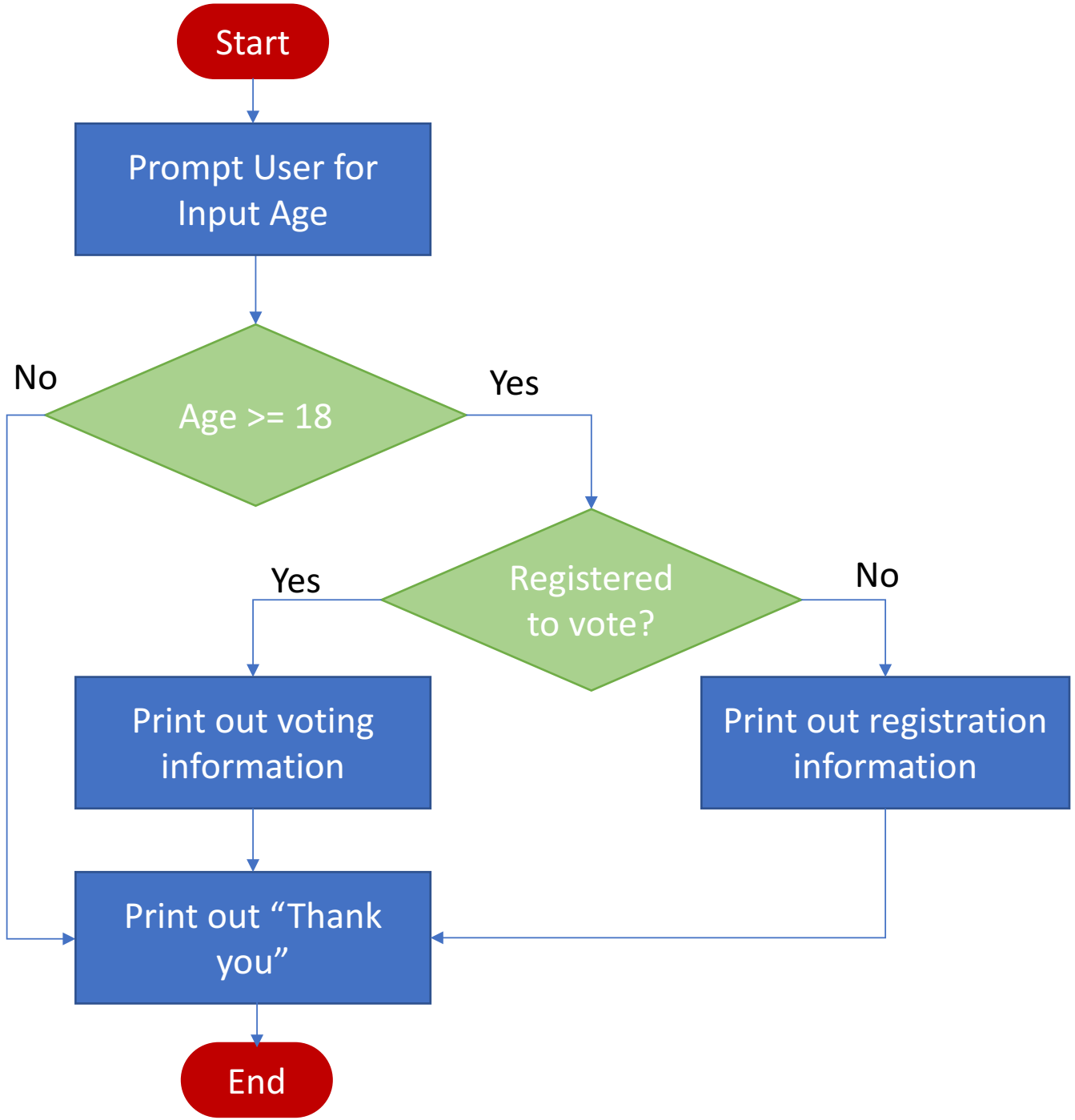
Conditionals

- An important “control structure” in programming concepts
  - **Controls the execution flow of a sequence**

Another important “control structure” in programming concepts is “Loops”

Flowchart

- Decision points
  - Conditions
  - Branches



Conditional Expressions

3.1.4. More Conditional Expressions

All the usual arithmetic comparisons may be made, but many do not use standard mathematical symbolism, mostly for lack of proper keys on a standard keyboard.

Meaning	Math Symbol	Python Symbols
Less than	<	`<`
Greater than	>	`>`
Less than or equal	≤	`<=`
Greater than or equal	≥	`>=`
Equals	=	`==`
Not equal	≠	`!=`

There should not be space between the two-symbol Python substitutes.

Notice that the obvious choice for *equals*, a single equal sign, is *not* used to check for equality. An annoying second equal sign is required. This is because the single equal sign is already used for *assignment* in Python, so it is not available for tests.

Warning: It is a common error to use only one equal sign when you mean to *test* for equality, and not make an assignment!

Tests for equality do not make an assignment, and they do not require a variable on the left. Any expressions can be tested for equality or inequality (`!=`). They do not need to be numbers! Predict the results and try each line in the *Shell*:

http://anh.cs.luc.edu/handsonPythonTutorial/ifstatements.html

If

3.1.2. Simple `if` Statements

Run this example program, suitcase.py. Try it at least twice, with inputs: 30 and then 55. As you an see, you get an extra result, depending on the input. The main code is:

```
weight = float(input("How many pounds does your suitcase weigh? "))
if weight > 50:
    print("There is a $25 charge for luggage that heavy.")
print("Thank you for your business.")
```

The middle two line are an `if` statement. It reads pretty much like English. If it is true that the weight is greater than 50, then print the statement about an extra charge. If it is not true that the weight is greater than 50, then don't do the indented part: skip printing the extra luggage charge. In any event, when you have finished with the `if` statement (whether it actually does anything or not), go on to the next statement that is *not* indented under the `if`. In this case that is the statement printing "Thank you".

The general Python syntax for a simple `if` statement is

Syntax

`if` *condition* `:`
    indentedStatementBlock

If the condition is true, then do the indented statements. If the condition is not true, then skip the indented statements.

http://anh.cs.luc.edu/handsonPythonTutorial/ifstatements.html

If-Else

3.1.3. `if-else` Statements

Run the example program, `clothes.py`. Try it at least twice, with inputs 50 and then 80. As you can see, you get different results, depending on the input. The main code of `clothes.py` is:

```
temperature = float(input('What is the temperature? '))
if temperature > 70:
    print('Wear shorts.')
else:
    print('Wear long pants.')
print('Get some exercise outside.')
```

The middle four lines are an if-else statement. Again it is close to English, though you might say "otherwise" instead of "else" (but else is shorter!). There are two indented blocks: One, like in the simple `if` statement, comes right after the `if` heading and is executed when the condition in the `if` heading is true. In the `if-else` form this is followed by an `else:` line, followed by another indented block that is only executed when the original condition is *false*. In an `if-else` statement exactly one of two possible indented blocks is executed.

A line is also shown **de**dented next, removing indentation, about getting exercise. Since it is dedented, it is not a part of the if-else statement: Since its amount of indentation matches the `if` heading, it is always executed in the normal forward flow of statements, after the `if-else` statement (whichever block is selected).

The general Python `if-else` syntax is

Syntax

```
if condition :
    indentedStatementBlockForTrueCondition
else:
    indentedStatementBlockForFalseCondition
```

These statement blocks can have any number of statements, and can include about any kind of statement.

http://anh.cs.luc.edu/handsonPythonTutorial/ifstatements.html

If-Elif

The most elaborate syntax for an `if-elif-else` statement is indicated in general below:

Syntax

```
if condition1 :
    indentedStatementBlockForTrueCondition1
elif condition2 :
    indentedStatementBlockForFirstTrueCondition2
elif condition3 :
    indentedStatementBlockForFirstTrueCondition3
elif condition4 :
    indentedStatementBlockForFirstTrueCondition4
else:
    indentedStatementBlockForEachConditionFalse
```

The `if`, each `elif`, and the final `else` lines are all aligned. There can be any number of `elif` lines, each followed by an indented block. (Three happen to be illustrated above.) With this construction *exactly one* of the indented blocks is executed. It is the one corresponding to the *first* `True` condition, or, if all conditions are `False`, it is the block after the final `else` line.

Be careful of the strange Python contraction. It is `elif`, not `elseif`. A program testing the letterGrade function is in example program `grade1.py`.

http://anh.cs.luc.edu/handsonPythonTutorial/ifstatements.html

Compound Boolean Expressions

3.1.7. Compound Boolean Expressions

To be eligible to graduate from Loyola University Chicago, you must have 120 credits *and* a GPA of at least 2.0. This translates directly into Python as a *compound condition*:

```
credits >= 120 and GPA >=2.0
```

This is true if both `credits >= 120` is true and `GPA >= 2.0` is true. A short example program using this would be:

```
credits = float(input('How many units of credit do you have? '))
GPA = float(input('What is your GPA? '))
if credits >= 120 and GPA >=2.0:
    print('You are eligible to graduate!')
else:
    print('You are not eligible to graduate.')
```

The new Python syntax is for the operator `and`:

Syntax

condition1 `and` *condition2*

The compound condition is true if both of the component conditions are true. It is false if *at least* one of the conditions is false.

http://anh.cs.luc.edu/handsonPythonTutorial/ifstatements.html

Compound Boolean Expressions

In the last example in the previous section, there was an `if-elif` statement where both tests had the same block to be done if the condition was true:

```
if x < xLow:
    dx = -dx
elif x > xHigh:
    dx = -dx
```

There is a simpler way to state this in a sentence: If x < xLow or x > xHigh, switch the sign of dx. That translates directly into Python:

```
if x < xLow or x > xHigh:
    dx = -dx
```

The word `or` makes another compound condition:

Syntax

condition1 `or` *condition2*

is true if *at least* one of the conditions is true. It is false if both conditions are false. This corresponds to *one* way the word "or" is used in English. Other times in English "or" is used to mean *exactly one* alternative is true.

http://anh.cs.luc.edu/handsonPythonTutorial/ifstatements.html

Nested If

```
res = input("Please enter your age: ")
age = int(res)
print("Your age is: " + str(age))

letter = input("Please enter an alphabet letter: ")

if (age >= 50):
  if (letter == 'a'):
     print("apple")
  elif (letter == 'b'):
     print("banana")
  elif (letter == 'c'):
     print("cantaloupe")
  elif (letter == 'd'):
     print("durian")
  else:
     print("Sorry!")
else:
  if (letter == 'a'):
     print("avocado")
  elif (letter == 'b'):
     print("blueberry")
  elif (letter == 'c'):
     print("cherry")
  elif (letter == 'd'):
     print("dragon fruit")
  else:
     print("Sorry!")
```

Nested If 2

```
res = input("Please enter your age: ")
age = int(res)
print("Your age is: " + str(age))

letter = input("Please enter an alphabet letter: ")

if (age >= 50):
  if (letter == 'a'):
     print("apple")
  elif (letter == 'b'):
     print("banana")
  elif (letter == 'c'):
     print("cantaloupe")
  elif (letter == 'd'):
     print("durian")
  else:
     print("Sorry!")
else:
  if (letter == 'a'):
     print("avocado")
  elif (letter == 'b'):
     print("blueberry")
  elif (letter == 'c'):
     print("cherry")
  elif (letter == 'd'):
     print("dragon fruit")
  else:
     print("Sorry!")
```

= ?

```
res = input("Please enter your age: ")
age = int(res)
print("Your age is: " + str(age))

letter = input("Please enter an alphabet letter: ")

if (age >= 50 and letter == 'a'):
  print("apple")
elif (age >= 50 and letter == 'b'):
  print("banana")
elif (age >= 50 and letter == 'c'):
  print("cantaloupe")
elif (age >= 50 and letter == 'd'):
  print("durian")
elif (age < 50 and letter == 'a'):
  print("avocado")
elif (age < 50 and letter == 'b'):
  print("blueberry")
elif (age < 50 and letter == 'c'):
  print("cherry")
elif (age < 50 and letter == 'd'):
  print("dragon fruit")
else:
  print("Sorry!")
```

Nested If 3

```
res = input("Please enter your age: ")
age = int(res)
print("Your age is: " + str(age))

letter = input("Please enter an alphabet letter: ")

if (age >= 50):
  if (letter == 'a'):
     print("apple")
  elif (letter == 'b'):
     print("banana")
  elif (letter == 'c'):
     print("cantaloupe")
  elif (letter == 'd'):
     print("durian")
  else:
     print("Sorry!")
else:
  if (letter == 'a'):
     print("avocado")
  elif (letter == 'b'):
     print("blueberry")
  elif (letter == 'c'):
     print("cherry")
  elif (letter == 'd'):
     print("dragon fruit")
  else:
     print("Sorry!")
```

= ?

```
res = input("Please enter your age: ")
age = int(res)
print("Your age is: " + str(age))

letter = input("Please enter an alphabet letter: ")

if (letter == 'a'):
  if (age >= 50):
     print("apple")
  else:
     print("avocado")
elif (letter == 'b'):
  if (age >= 50):
     print("banana")
  else:
     print("blueberry")
elif (letter == 'c'):
  if (age >= 50):
     print("cantaloupe")
  else:
     print("cherry")
elif (letter == 'd'):
  if (age >= 50):
     print("durian")
  else:
     print("dragon fruit")
else:
  print("Sorry!")
```

Pitfalls: Indentation

- **IMPORTANT**: Use indentations to designate “scope” or “block”

```
x = int(input("enter a number:"))

if (x > 0):
   print("You passed!")
   if (x > 90):
      print("Excellent!")
      print("Keep up the good work!")
print("Bye")
```

= ?

```
x = int(input("enter a number:"))

if (x > 0):
   print("You passed!")
   if (x > 90):
      print("Excellent!")
   print("Keep up the good work!")
print("Bye")
```

- Blank lines are ignored
- Comments on a line by themselves are ignored