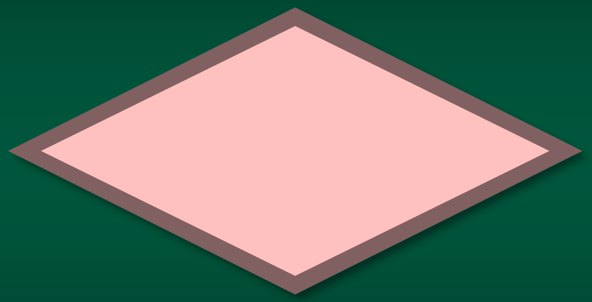
Boolean & If Statements

Chapter 4

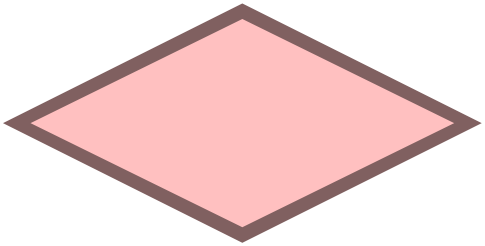


Introduction to Decision Structures

Chapter 4.1

Introduction to Decision Structures

- A *decision structure* allows a program to perform actions only under certain conditions
- Found in virtually every programming language



Different Types of Decisions

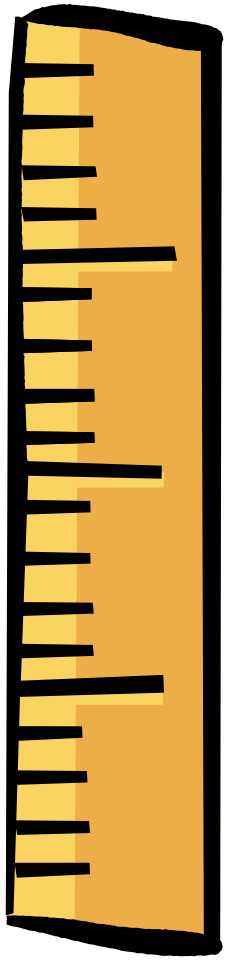
- *If Statement*
  - gives a single, optional alternative
  - very common in programming
- *If-Then-Else*
  - gives dual (two) alternatives
  - a form of the If Statement
- *Case structure*
  - gives multiple alternative decisions
  - not supported in Visual Logic

Boolean Logic

- Decision structures rely on having the computer compare data and make a decision based on the result
- For this, computers use *Boolean Logic*
  - basis of all computer (and human) logic
  - Boolean values can be either *True* or *False*
  - understanding Boolean logic is essential to knowing how to program

Relational Operators

- A *Relational Operator* determines whether a specific relationship exists between two values
- The most basic way to test two pieces of data
- Returns either True or False



Relational Operators

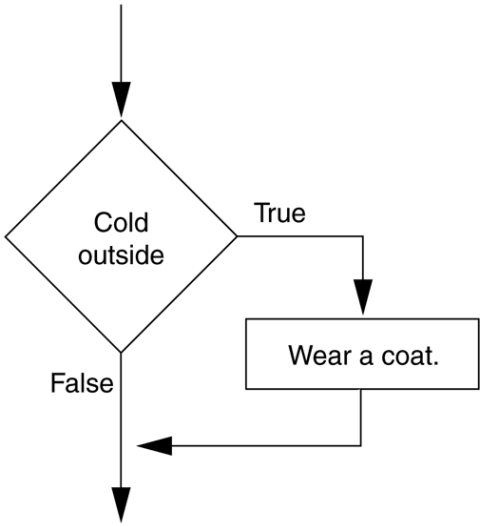
Operator	Name
==	Equal To
!=	Not Equal To
>	Greater Than
>=	Greater Than or Equal
<	Less Than
<=	Less Than or Equal

Relational Examples

`1 == 4`	False
`24 != 31`	True
`11 < 10`	False
`100 >= 100`	True

If Statements

- An action only occurs if the Boolean expression is True
- Otherwise, *nothing* will occur
- A diamond symbol is used in flowcharts



Book Pseudocode

Either True or False

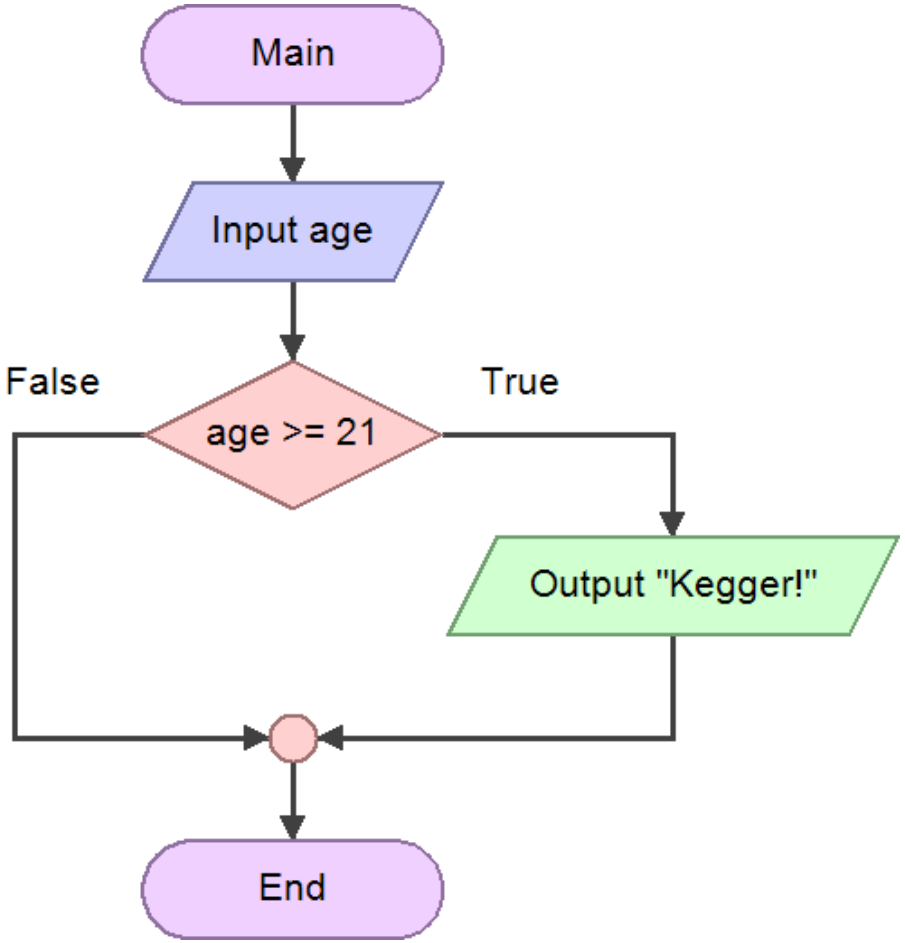
```
If Condition Then
   Statements
End If
```

Multiple statements

Example in Pseudocode

```
Declare Integer age

Input age
If age >= 21 Then
   Display "Kegger!"
End If
```

Example in a Flowchart



Example Output

```
22
Kegger!
```

Example 2 Output

20

Nothing.
The Display Statement is not executed

Example

```
Declare Integer age

Input age
If age >= 21 Then
   Display "Kegger!"
End If
```

Example 2

```
Declare Integer guess

Display "Year CSUS founded? "
Input guess
If guess == 1947 Then
   Display "Correct!"
End If
```

Increment Example Output

```
Year CSUS founded? 1992
```

Increment Example Output

```
Year CSUS founded: 1947
Correct!
```

Dual Alternative Decision Structures

Chapter 4.2

Dual Alternative Structures

- *Dual Alternative Decision Structures* selects between two different groups of statements
- If the Boolean expression is True it executes one group and, if False, the other

The Else Clause

- The If Statement has an optional "else" clause
- This denotes the group that executes in the expression is *False*
- It goes before the "End If" since it is part of the same If-Statement

Book Pseudocode

```
If Condition Then
    Statements
else
    Statements
End If
```

Processed if the Condition is false

Pseudocode

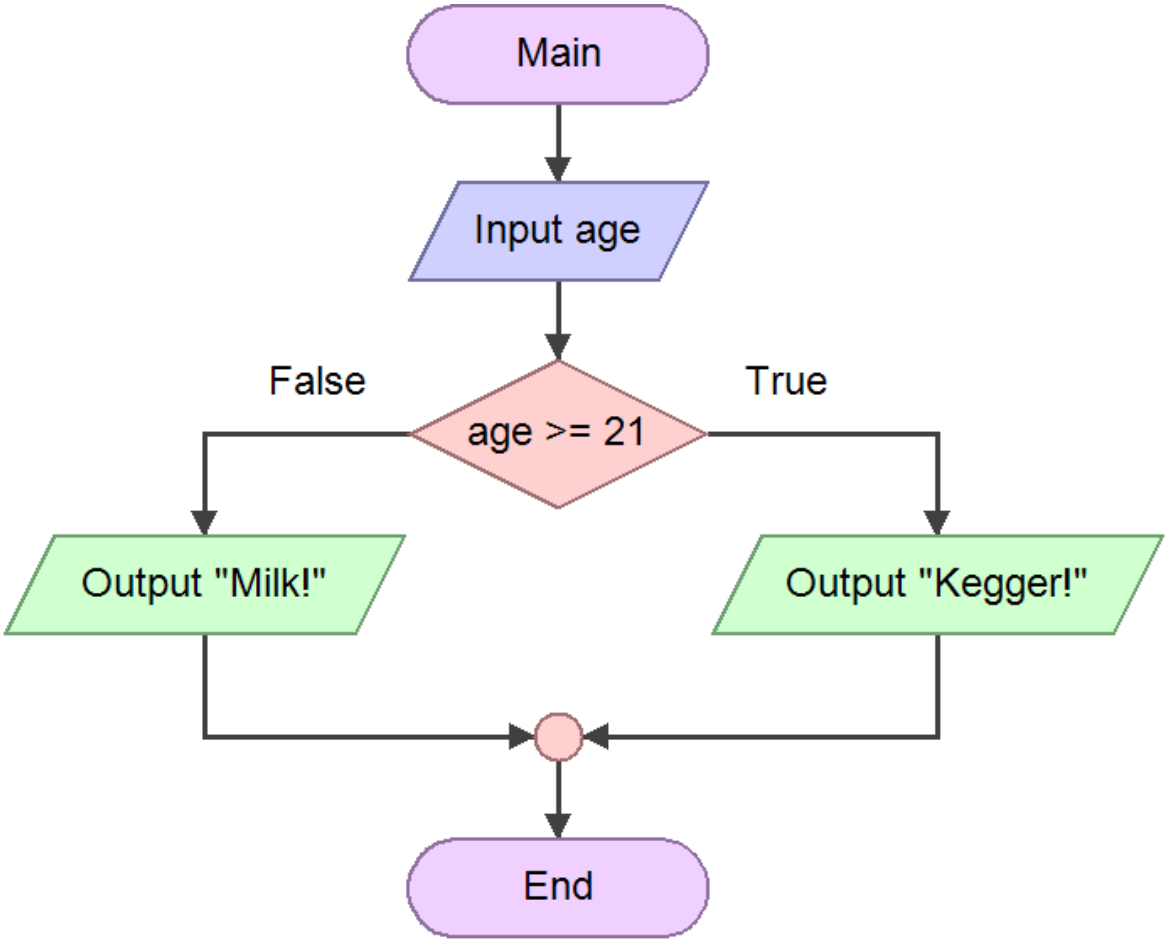
```
If condition Then
   statement
   statement
Else
   statement
   statement
End if
```

```
If temp < 40 Then
   Display "It's cold"
   Display "Get a coat!"
Else
   Display "It's warm"
   Display "Get water!"
End if
```

Dual Example in Pseudocode

```
Declare Integer age

Input age

If age >= 21 Then
    Display "Kegger!  :)"
Else
    Display "Milk!  :("
End If
```

Dual Example in a Flowchart

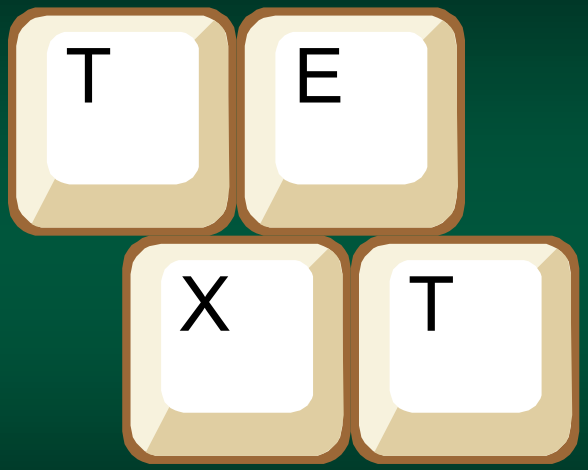


Else Example Output

```
22
Kegger! :)
```

Else Example Output

```
18
Milk! :(
```

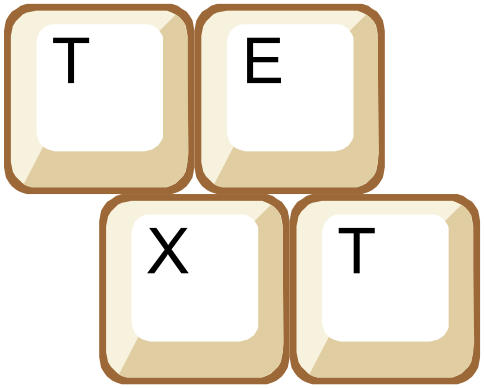


Comparing Strings

Chapter 4.3

Comparing Strings

- Most languages allow you to compare strings
- Textual data is often used to make decisions
- The same rules that apply to comparing numbers, applies to strings



String Equality

- You can test two string variables (or literals) for equality
- You can also test if one string is greater or less than another string (allows for sorting strings)

```
name1 == name2
month != "February"
```

Case Sensitivity

- String comparisons are generally *case sensitive*
- This means that uppercase and lower case letters are not the same
- Why? Let's look back at ASCII…

ASCII Chart Review

NUL	SOL	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
DLE	CD1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
sp	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

ASCII Codes

- Each character has a unique value
- The following is how "Moe" is stored in ASCII

Character	Binary	Decimal
M	01001101	77
o	01101111	111
e	01100101	101

Comparing Letters

ASCII Value

A	01000001	65
B	01000010	66
C	01000011	67
D	01000100	68
E	01000101	69
F	01000110	70

a	01100001	97
b	01100010	98
c	01100011	99
d	01100100	100
e	01100101	101
f	01100110	102

```
"A" < "a"
```

Letter Examples

`"a" == "A"`	False
`"a" > "A"`	True
`"abC" < "abc"`	True
`"dog" != "cat"`	True

Example

```
Declare String answer

Display "Do you love programming?"
Input answer

If answer = "y" Then
   Display "Most excellent!"
End If
```

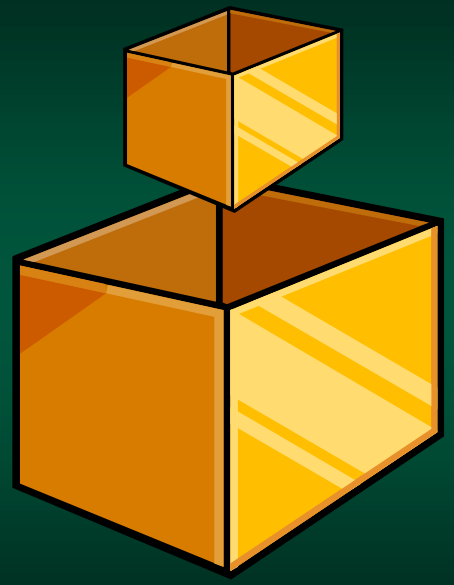
Example Output

```
Do you love programming?
y
Most excellent!
```

Example Output

```
Do you love programming?
y
```

Nothing!
"y" is not equal to "Y"

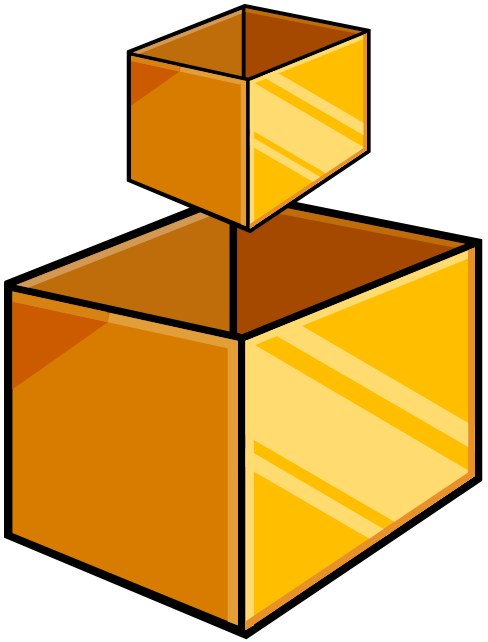


Nested Decision Structures

Chapter 4.4

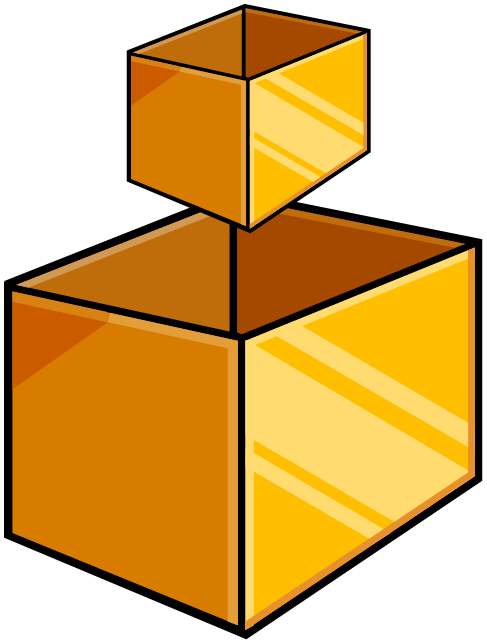
Nested Decision Structures

- In programming, control structures can be put inside other structures
- This is called *nesting* and allows complex control



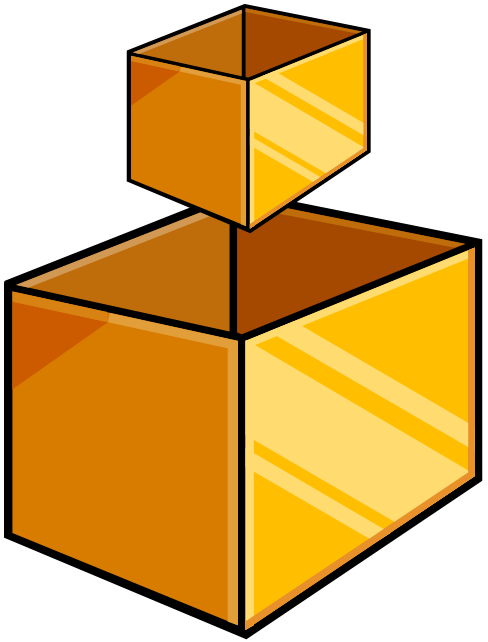
Nested Decision Structures

- By nesting…
  - you can put almost *anything* in *anything*
  - this allows you to create complex programs

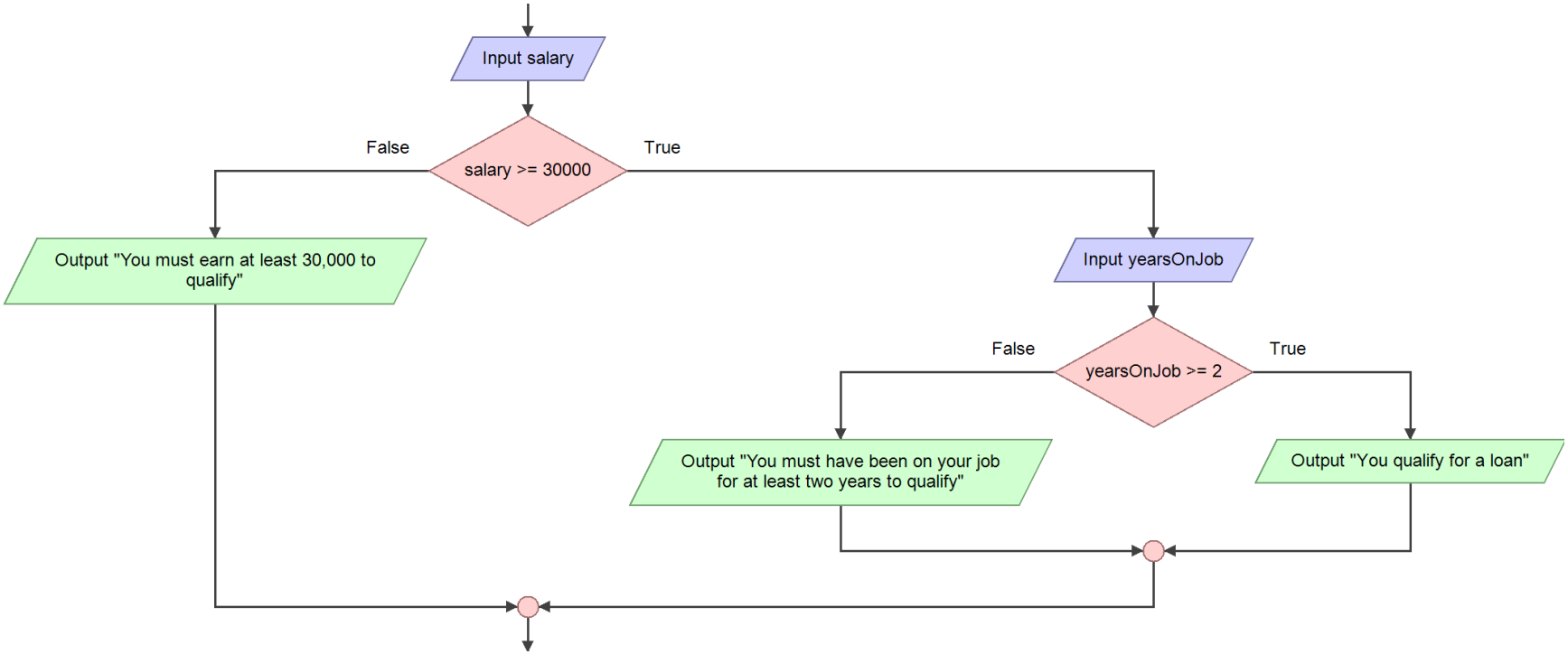


Nested If Statements

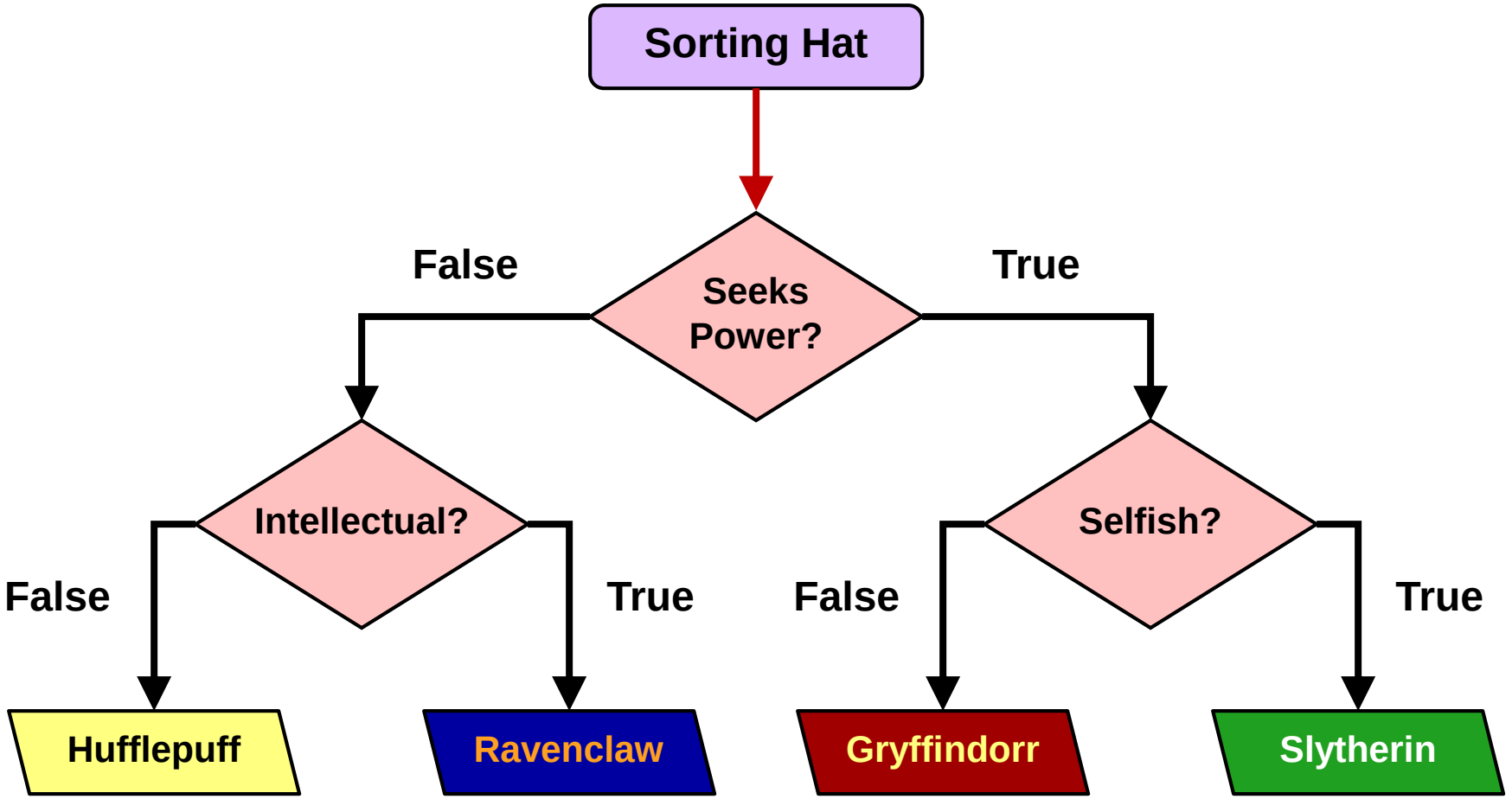
- Often If Statements are embedded in other If Statements
- This can give multiple branches in the program
- … and have a conditional statement dependent on another



Nested Decision Structures

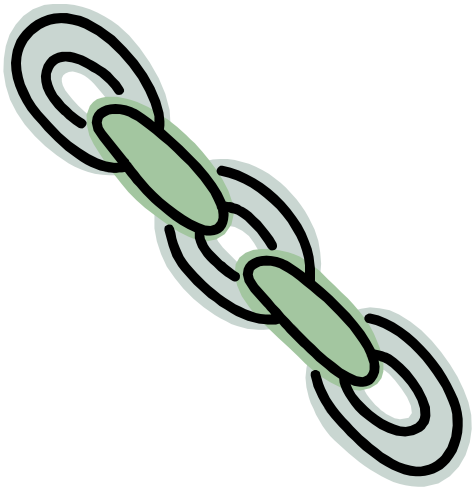


Harry Potter Example



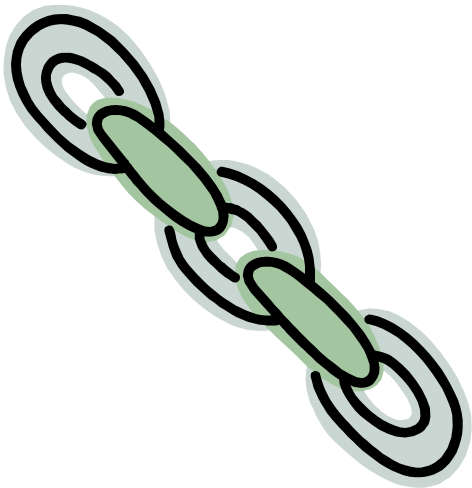
If-Else Chain

- Sometimes an If Statement is nested on the else clause of another If Statement
- It allows what is called an *"If-Else Chain"*



If-Else Chain

- It can make nested logic simpler to write
- Basically, the chain checks multiple expressions and selects the first one that is true



If-Else Chain

```
If score < 60 Then
    Display "Grade is F."
Else If score < 70 Then
    Display "Grade is D."
Else If score < 80 Then
    Display "Grade is C."
Else If score < 90 Then
    Display "Grade is B."
Else
    Display "Grade is A."
End If
```

Indenting is Vital

- You must indent each nested block of statements
- Otherwise your program quickly becomes unreadable

Logical Operators

Chapter 4.6

Logical Operators

- *Logical Operators* are used between comparisons to create complex Boolean expressions
- In Boolean logic, there are just 3 operators – which are easy to learn and master

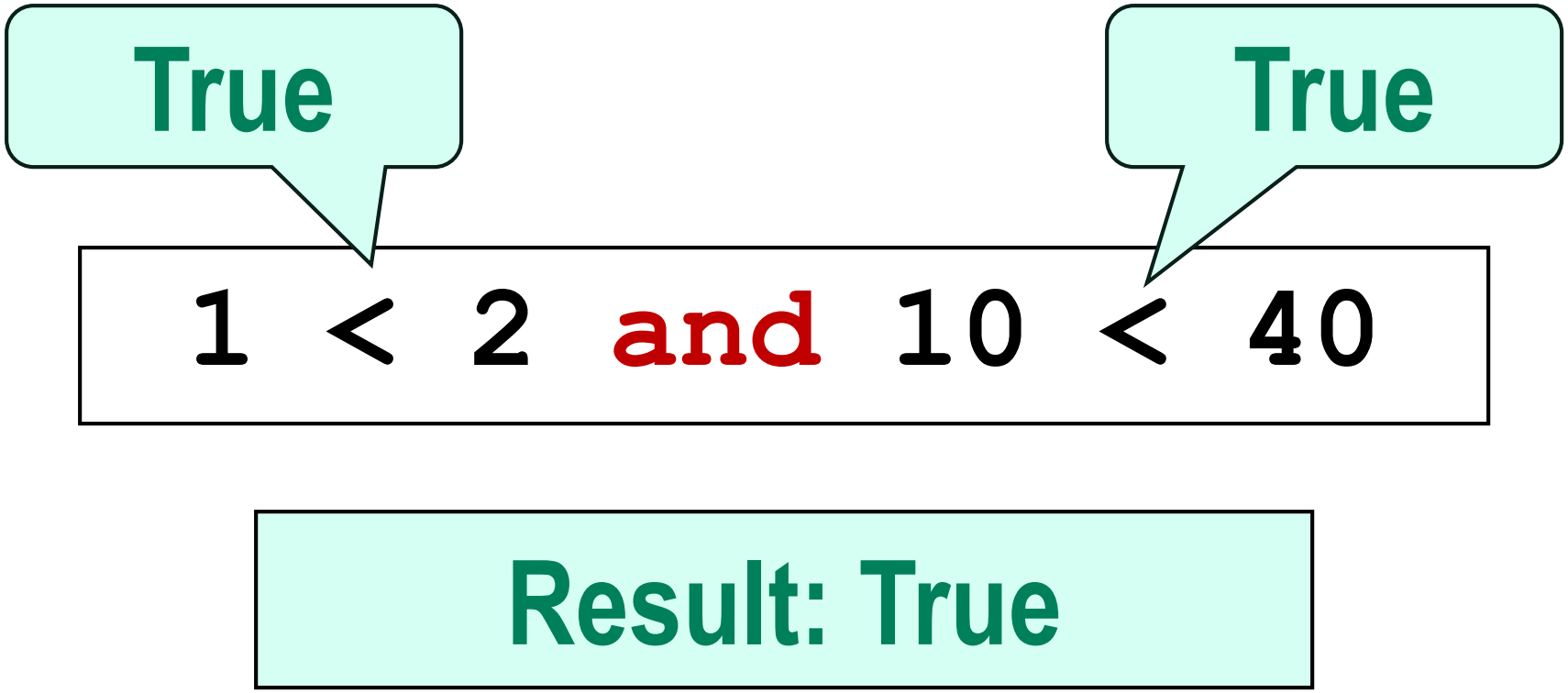
Logical Operators

Name	Rules
And	True only if both operands are True If either is False, the result is False
Or	True if either operand is True False if both operands are False
Not	True if the operand is False False if the operand is True

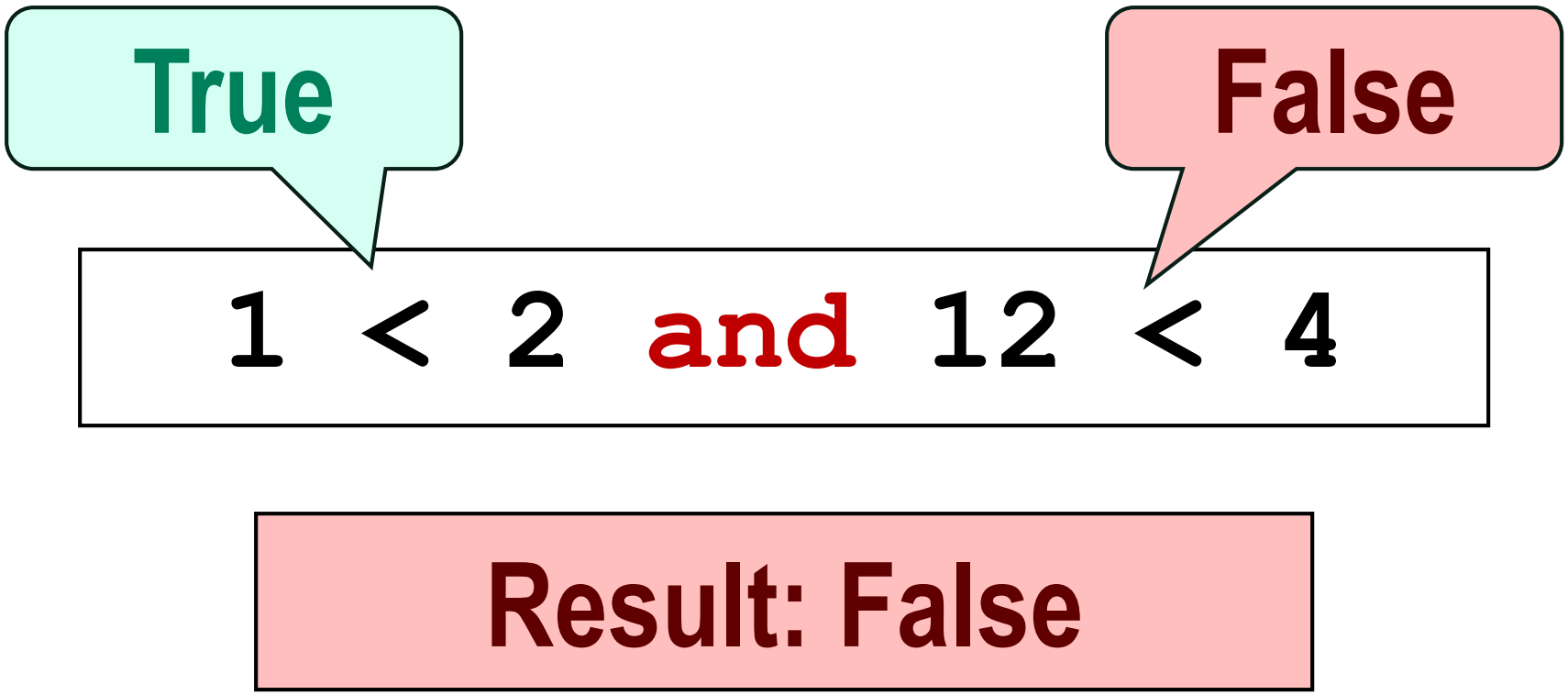
Boolean Table

p	q	not p	p or q	p and q
True	True	False	True	True
True	False	False	True	False
False	True	True	True	False
False	False	True	False	False

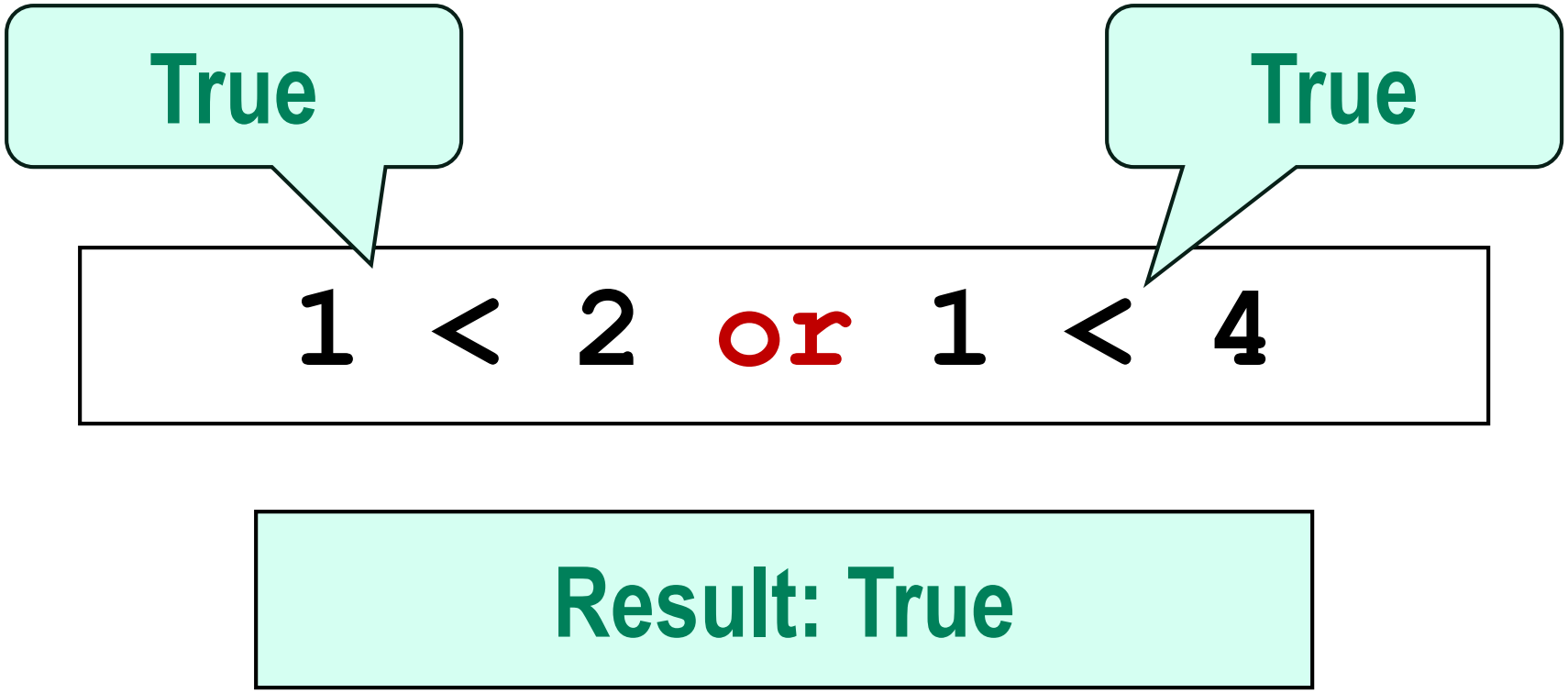
AND Example



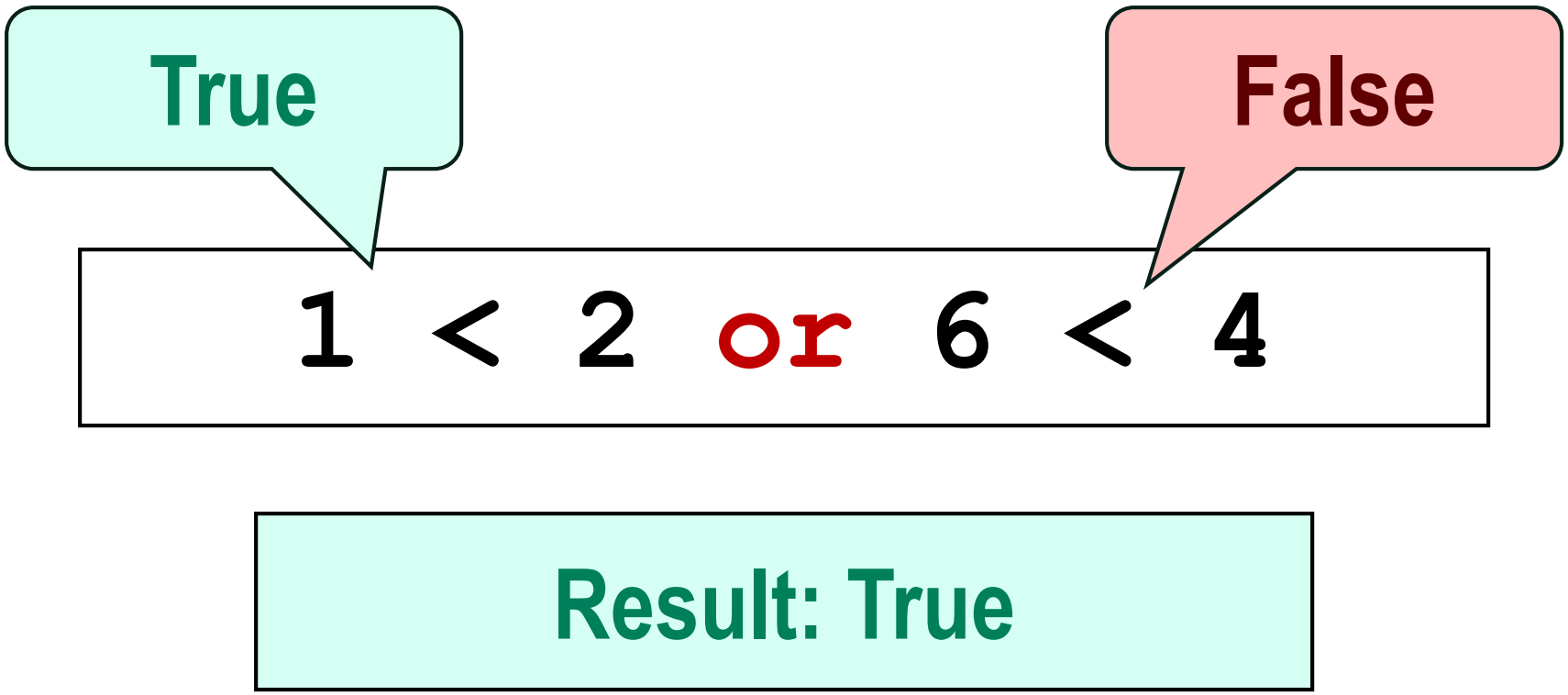
AND Example 2



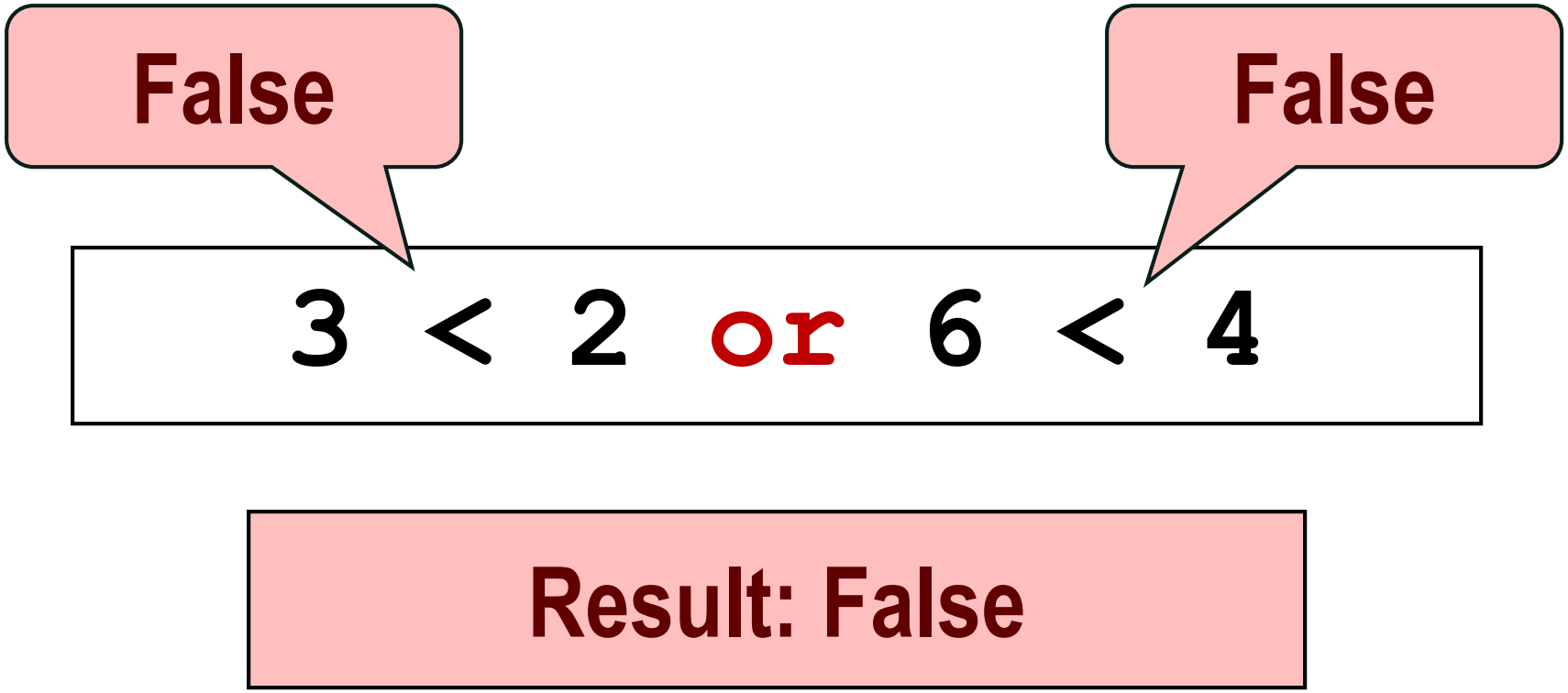
OR Example



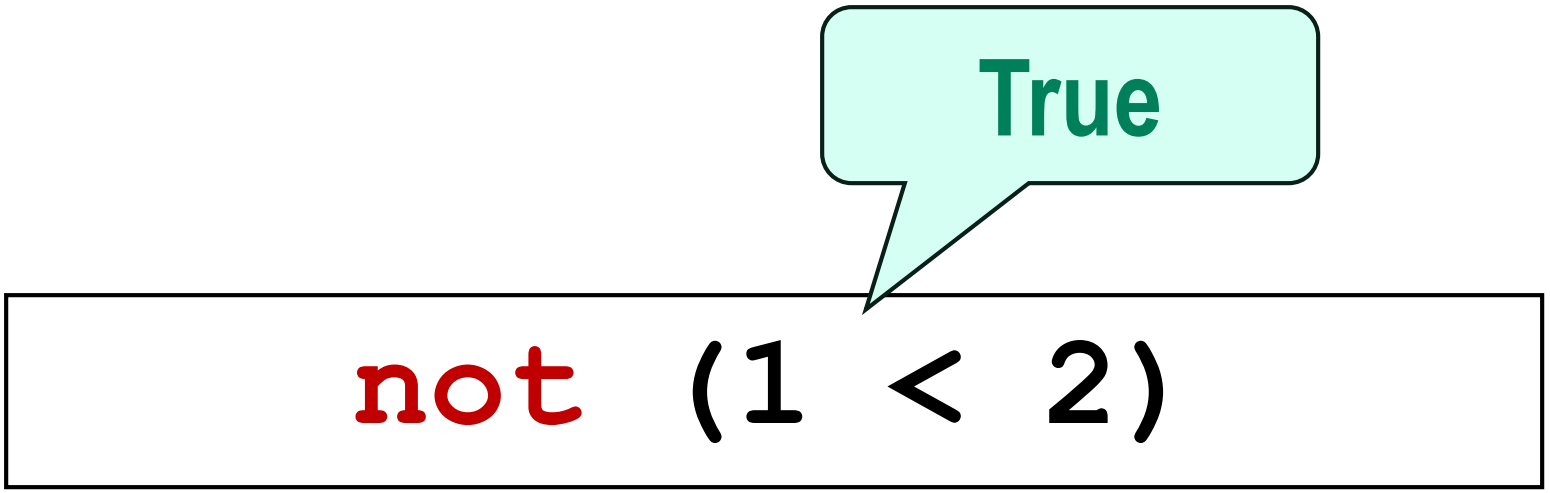
OR Example 2



OR Example 3

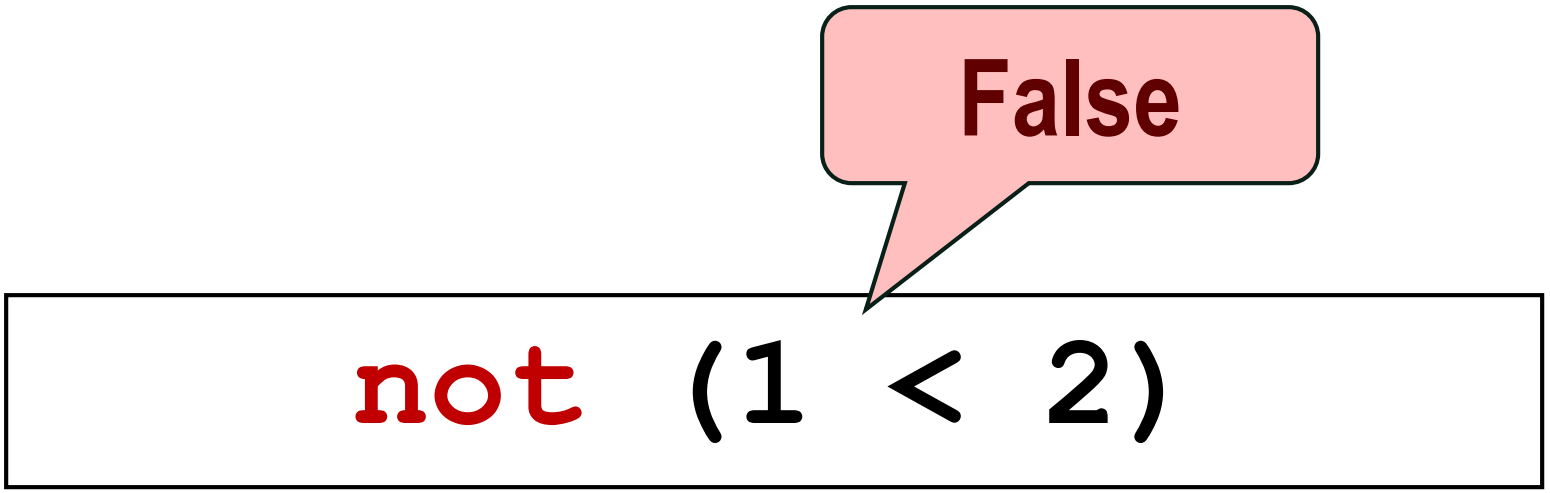


NOT Example



Result: False

NOT Example 2



Result: True

Examples

`1 < 3 and 10 < 40`	True
`1 == 3 and 10 < 40`	False
`not 1 == 2`	True
`1 > 3 or 30 < 20`	False

Typical Precedence Levels

7	- (unary) not
6	* /
5	+ -
4	== != > >= < <=
3	
2	and
1	or

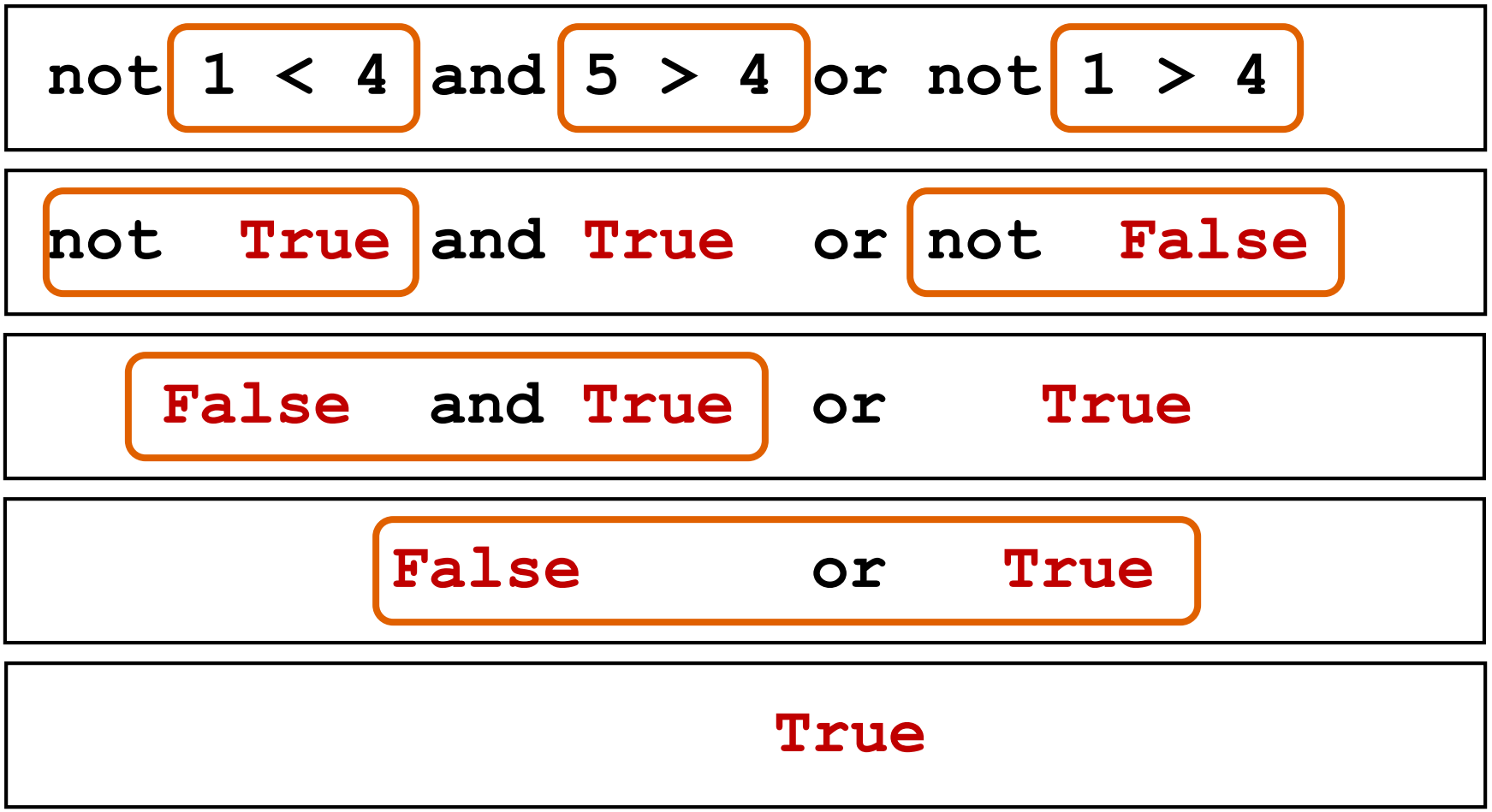
Highest

Lowest

Calculate The Result

```
2 == 4 or 1 > 2 and 5 > 4 or 1 < 3

False  or False and True  or True

False  or       False     or True

        False             or True

                          True
```

Calculate The Result 2



And Example

```
If temp < 20 AND minutes > 12 Then
    Display "Danger!"
    Display "Temperature danger zone."
End If
```

Example

```
Declare String answer

Display "Do you love programming?"
Input answer

If answer = "y" or answer = "Y" Then
    Display "Most excellent!"
End If
```

Example Output

```
Do you love programming?
y
Most excellent!
```

Range Checking

Chapter 4.6

Range Checking

- Range Checking is often used to test is between two values
- Often used to prevent invalid calculations

Inside a Range

- When checking for a number inside a range, use AND
- The following is only true if x is between 0 and 100

```
If x >= 0 AND x <= 100 Then
    Display "Inside Range"
End If
```

Outside a Range

- When checking for a number outside a range, use OR
- The following is true if x is outside the range

```
If x < 20 OR x > 40 Then
    Display "Outside range"
End If
```

Range Checking Example

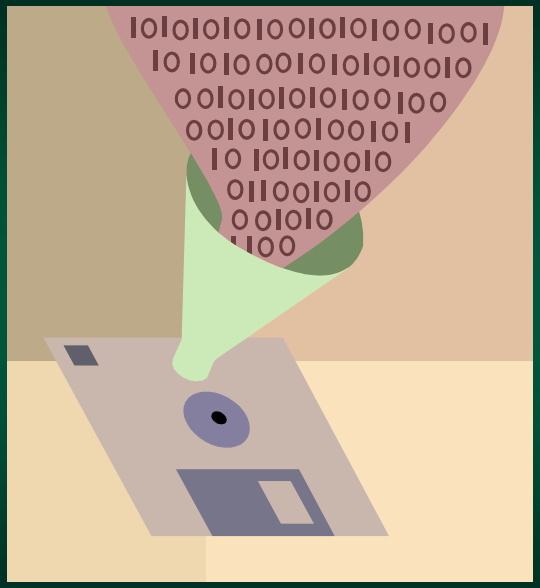
```
Declare Integer years, days
Input years

If years >= 1 and years <= 110 Then
   Set days = years * 365;
   Display "Days lived: ", days
Else
   Display "Invalid Age!"
End If
```

Range Checking Example Output

```
20
Days old: 7300
```

Range Checking Example Output

```
300
Invalid Age!
```

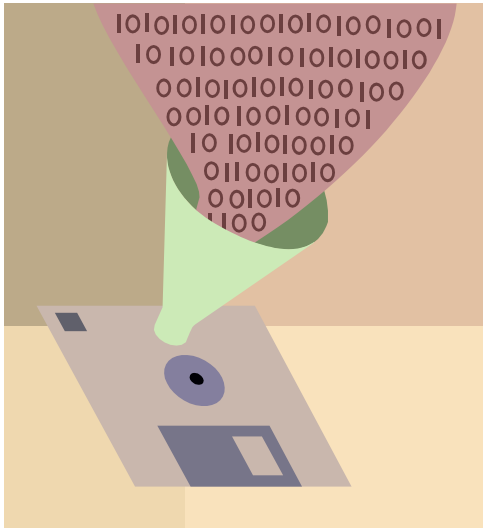


Boolean Variables

Chapter 4.7

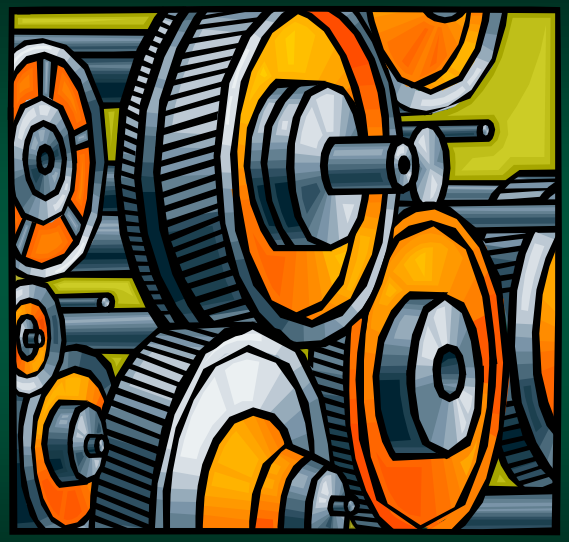
Boolean Variables

- A variable of the Boolean data type can hold one or two values: true or false
- It often holds the result of a Boolean expression – which will be used later in another expression



Boolean Variable Example

```
Declare Boolean isLunchTime

If time >= 12 and time <= 13 then
     Set isLunchTime = True
Else
     Set isLunchTime = False
End If
```

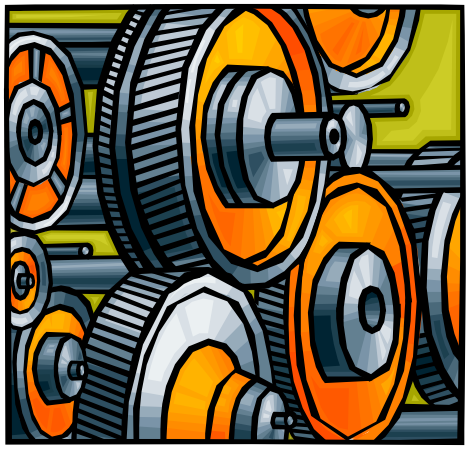


Factoring

Cleaning up code

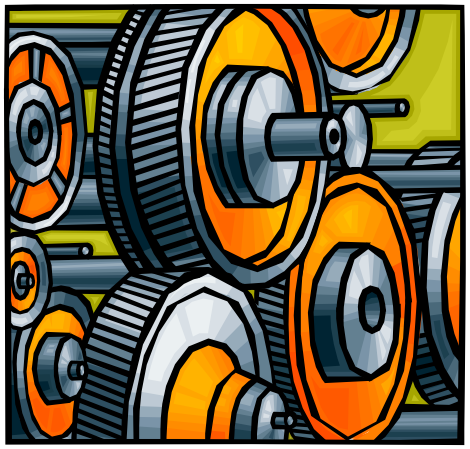
Factoring

- Often, when creating complex if statements, code is repeated in both the true branch and the false branch
- This redundant code can, sometimes, moved outside the loop



Factoring

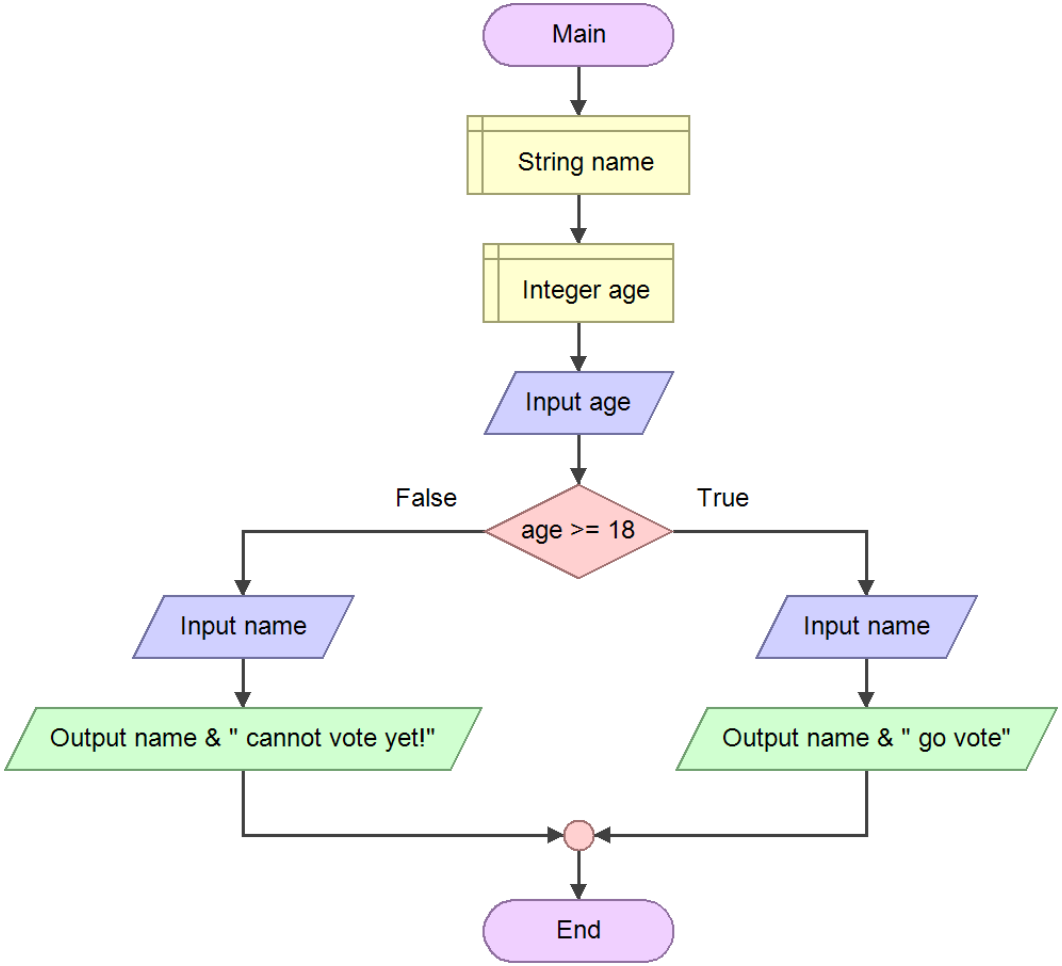
- This only works if the duplicate is at the beginning of both the True branch and the False branch
- When it is done, the code is *"factored"* out of the If Statement



Example – What's Redundant?

```
Input age
If age >= 18 then
   Input name
   Display name, " can vote"
Else
   Input name
   Display name, " cannot vote yet"
End If
```

Example – What's Redundant?



Example – Redundant Code

```
Input age
If age >= 18 then
    Input name
    Display name, " can vote"
Else
    Input name
    Display name, " cannot vote yet"
End If
```

Example – Factored Out

```
Input age
Input name
If age >= 18 then
   Display name, " can vote"
Else
   Display name, " cannot vote yet"
End If
```

Example – Factored Out