

Control Flow

CS105 : Saelee

We use test and Boolean operators to help programs make decisions

Boolean Expressions

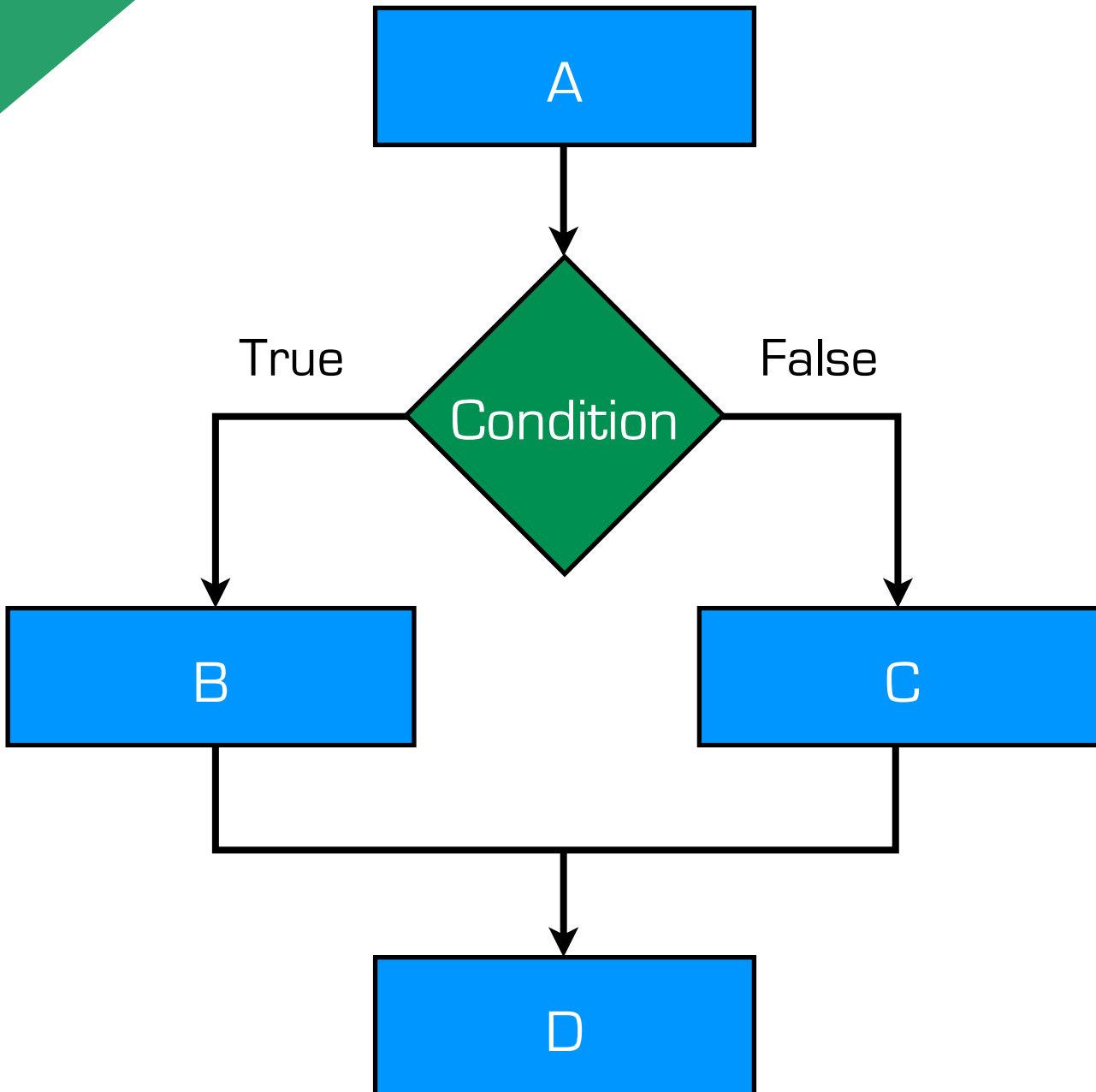
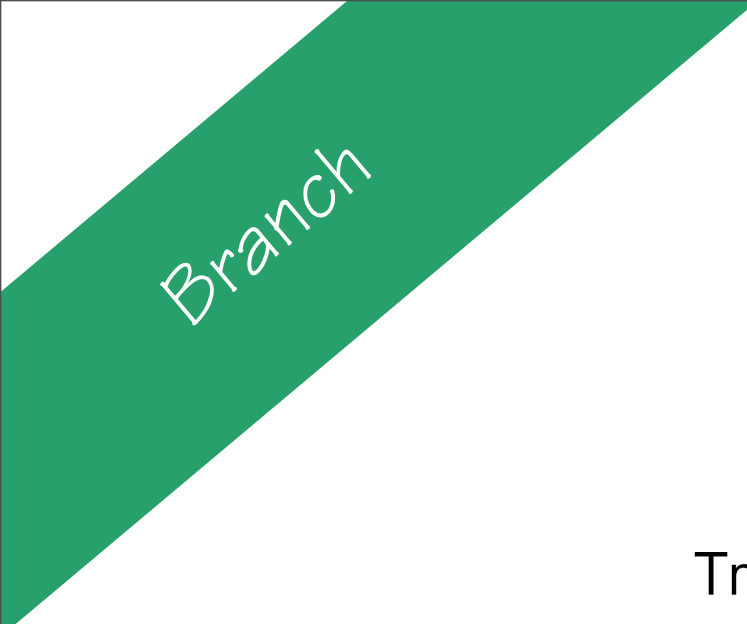
- `temp == 100`
- `divisor == 0`
- `answer == "yes"`
- `(exp - pred).abs < epsilon`

Compound Tests

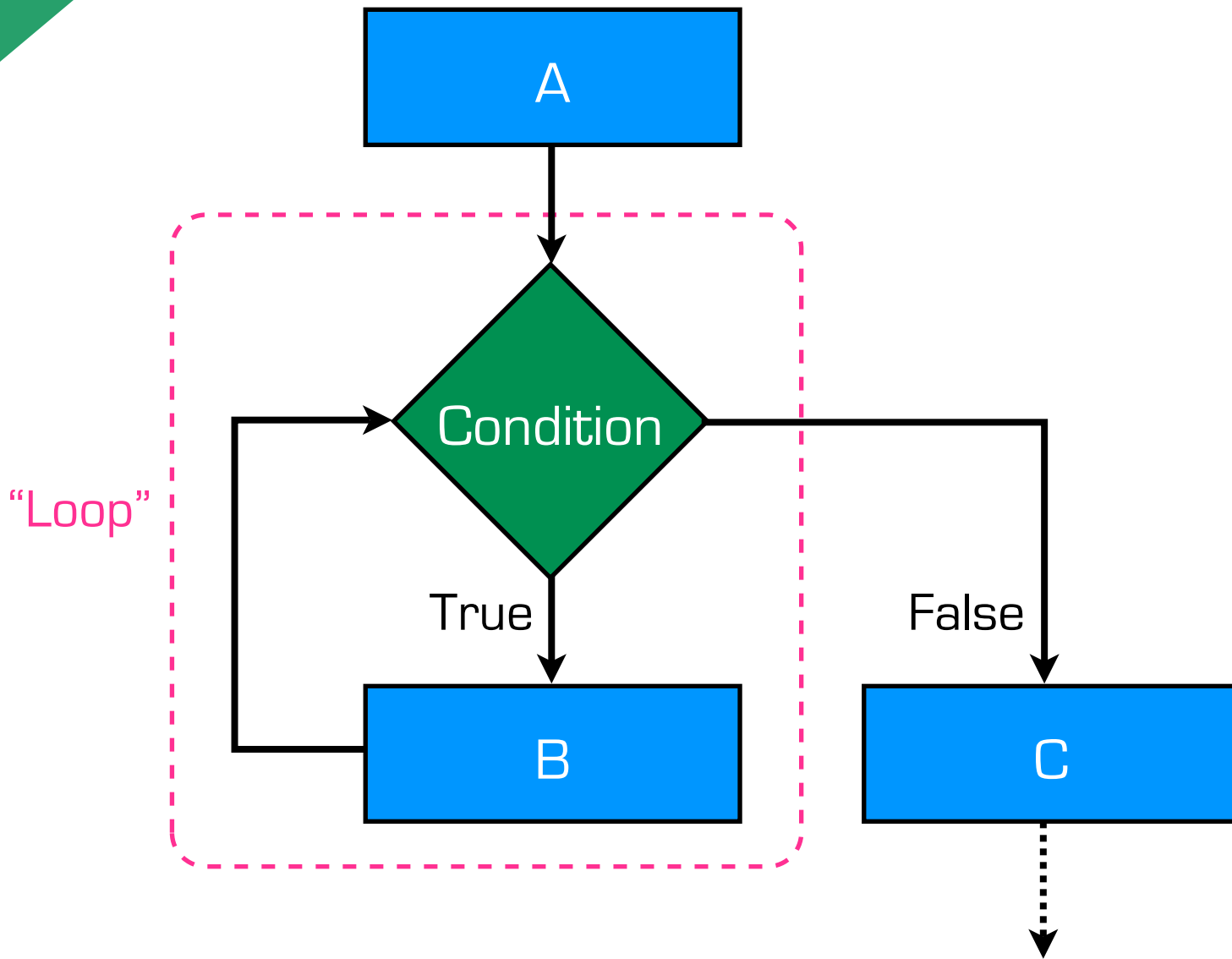
- `temp > 75 && !raining`
- `score >= 70 && score < 80`
- `price < 1500 && (size > 32 || res >= 720)`

Next Step:
Use Booleans to manipulate
control flow

Two types of control flow:
Branches and Loops



Loop



Branches: The `if` Construct

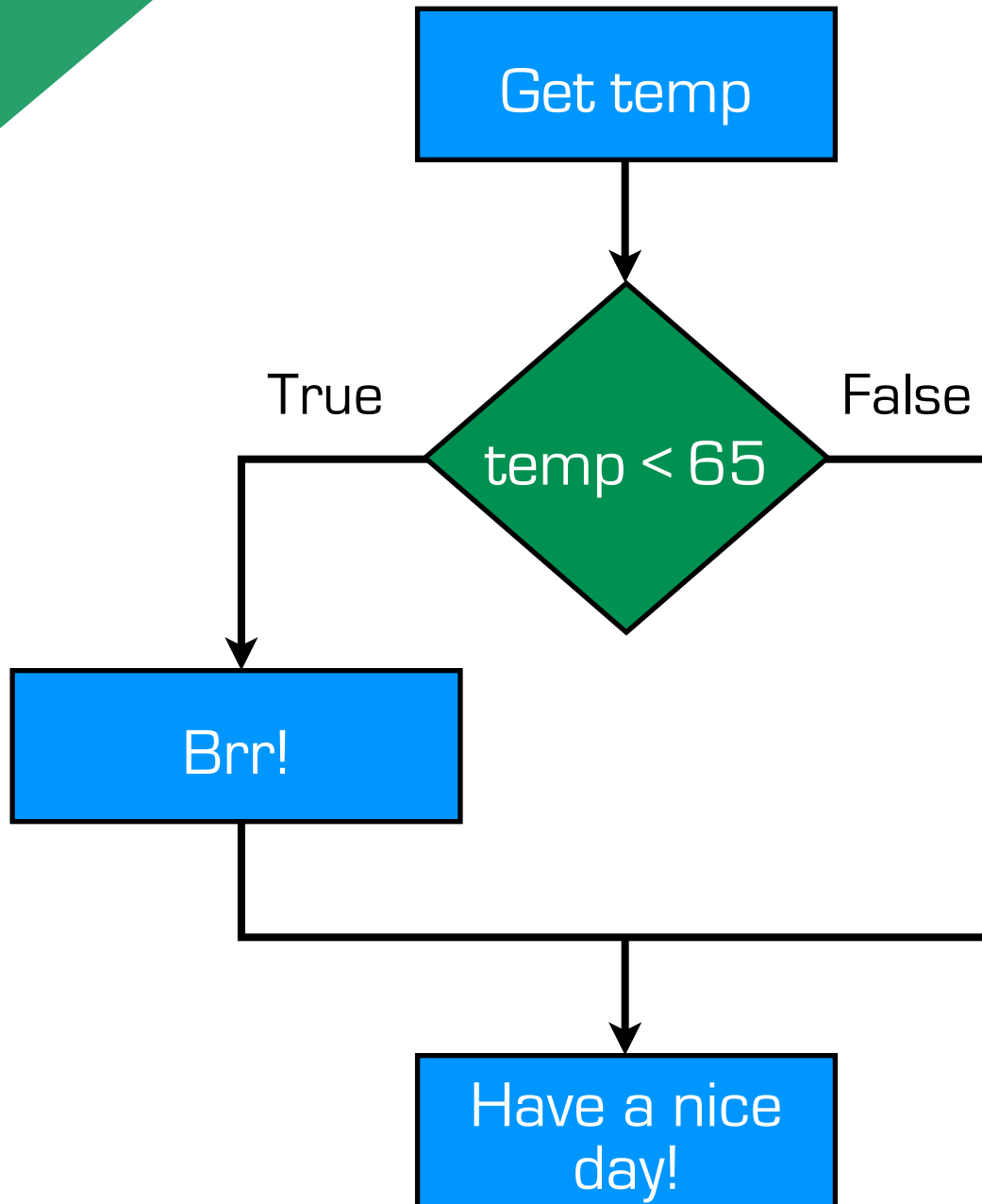
The "if" Construct

```
puts "Please enter the temperature:"  
temp = gets.to_i  
  
puts "So it's #{temp} today."  
  
if temp < 65  
  puts "Brrr! I'm staying home!"  
end  
  
puts "Have a nice day!"
```

```
Please enter the temperature:  
75  
So it's 75 today.  
Have a nice day!
```

```
Please enter the temperature:  
59  
So it's 59 today.  
Brrr! I'm staying home!  
Have a nice day!
```

Semantics



“if-else”

```
puts "Please enter the temperature:"  
temp = gets.to_i  
  
puts "So it's #{temp} today."  
  
if temp < 65  
  puts "Brrr! I'm staying home!"  
else  
  puts "Leave the coat!"  
end  
  
puts "Have a nice day!"
```

```
Please enter the temperature:  
75  
So it's 75 today.  
Leave the coat!  
Have a nice day!
```

```
Please enter the temperature:  
59  
So it's 59 today.  
Brrr! I'm staying home!  
Have a nice day!
```

“if-elsif-else”

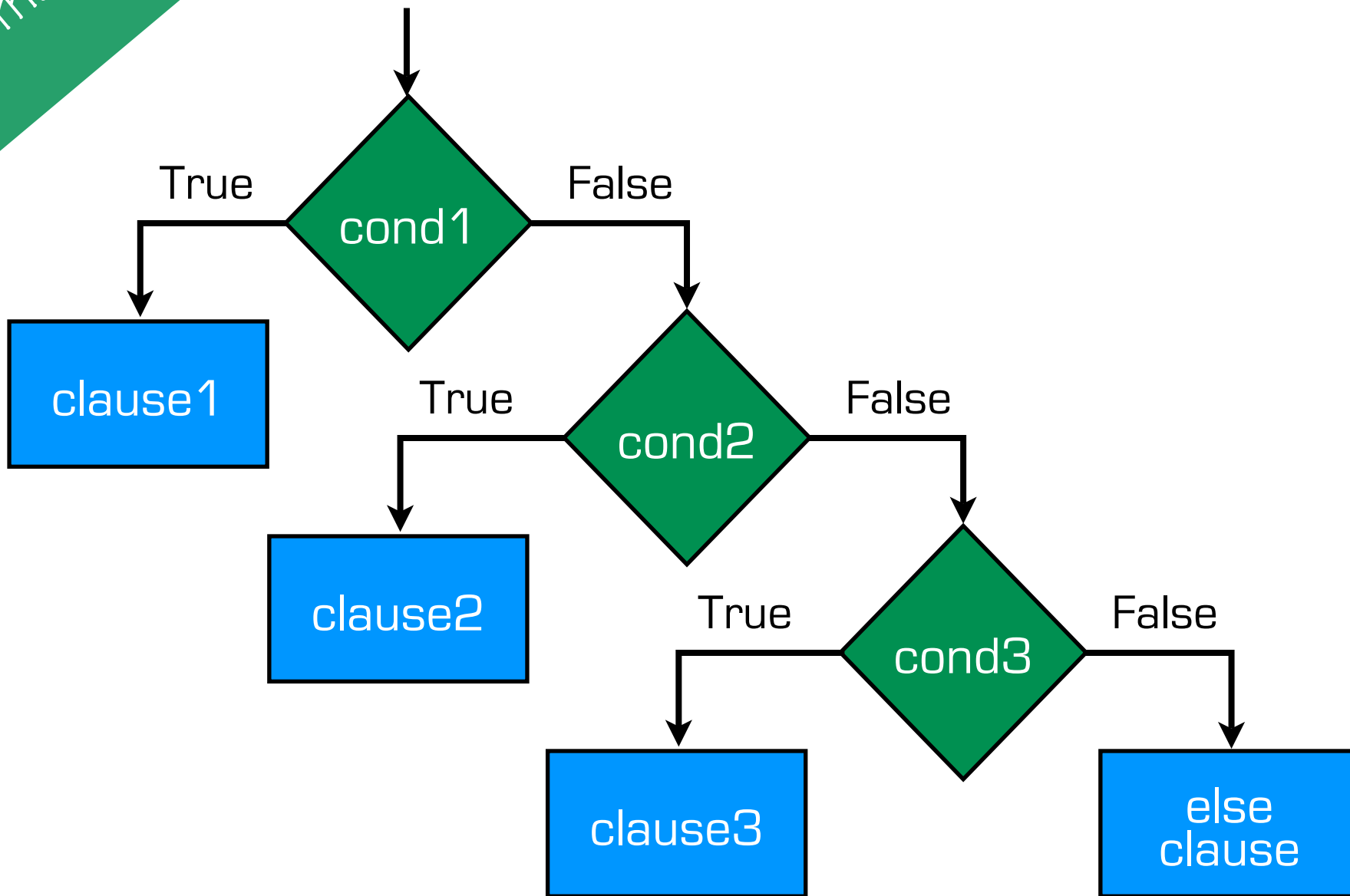
```
puts "Please enter the temperature:"  
temp = gets.to_i  
  
puts "So it's #{temp} today."  
  
if temp < 50  
  puts "Going out? Are you nuts?"  
elsif temp < 65  
  puts "Brrr! I'm staying home!"  
else  
  puts "Leave the coat!"  
end  
  
puts "Have a nice day!"
```

```
Please enter the temperature:  
40  
So it's 40 today.  
Going out? Are you nuts?  
Have a nice day!
```

General Branch Syntax

```
if cond1
  # Do this if cond1 is true
elsif cond2
  # Do this if cond1 is false
  # and cond2 is true
elsif cond3
  # Do this if cond1 and cond2 are false
  # and cond3 is true
  ...
else
  # Do this if everything else is false
end
```

Semantics



Remember:
`elsif` and `else`
clauses are optional

Nested Branches

- You can **nest** branches for more complex control flow

```
if cond1
  # Do this if cond1 is true
  if cond1_1
    # Do this if cond1 and cond1_1 are true
  else
    # Do this if cond1 is true and cond1_1 is false
  end
elseif cond2
  # Do this if cond1 is false and cond2 is true
else
  # Do this if everything else is false
end
```

On Code Style

- Indentation is not required by the Ruby interpreter, but it makes it easier for humans to see what's going on
- Keep the `if`, `elsif`, `else`, and terminating end lines at the same level of indentation
 - Indent the clauses 2 or more spaces, consistently
- A good editor should take care of indentation for you

Poor Indentation

- What on earth does this do?

```
if cond1
# Do this if cond1 is true
  if cond1_1
# Do this if cond1 and cond1_1 are true
  else
# Do this if cond1 is true and cond1_1 is false
  end
  elsif cond2
# Do this if cond1 is false and cond2 is true
  else
# Do this if everything else is false
  end
```



Programs must be written for people to read,
and only incidentally for machines to execute.



Abelson & Sussman,
Structure and Interpretation of Computer Programs

Demo



Loops:

The `while` Construct

The "while" Loop

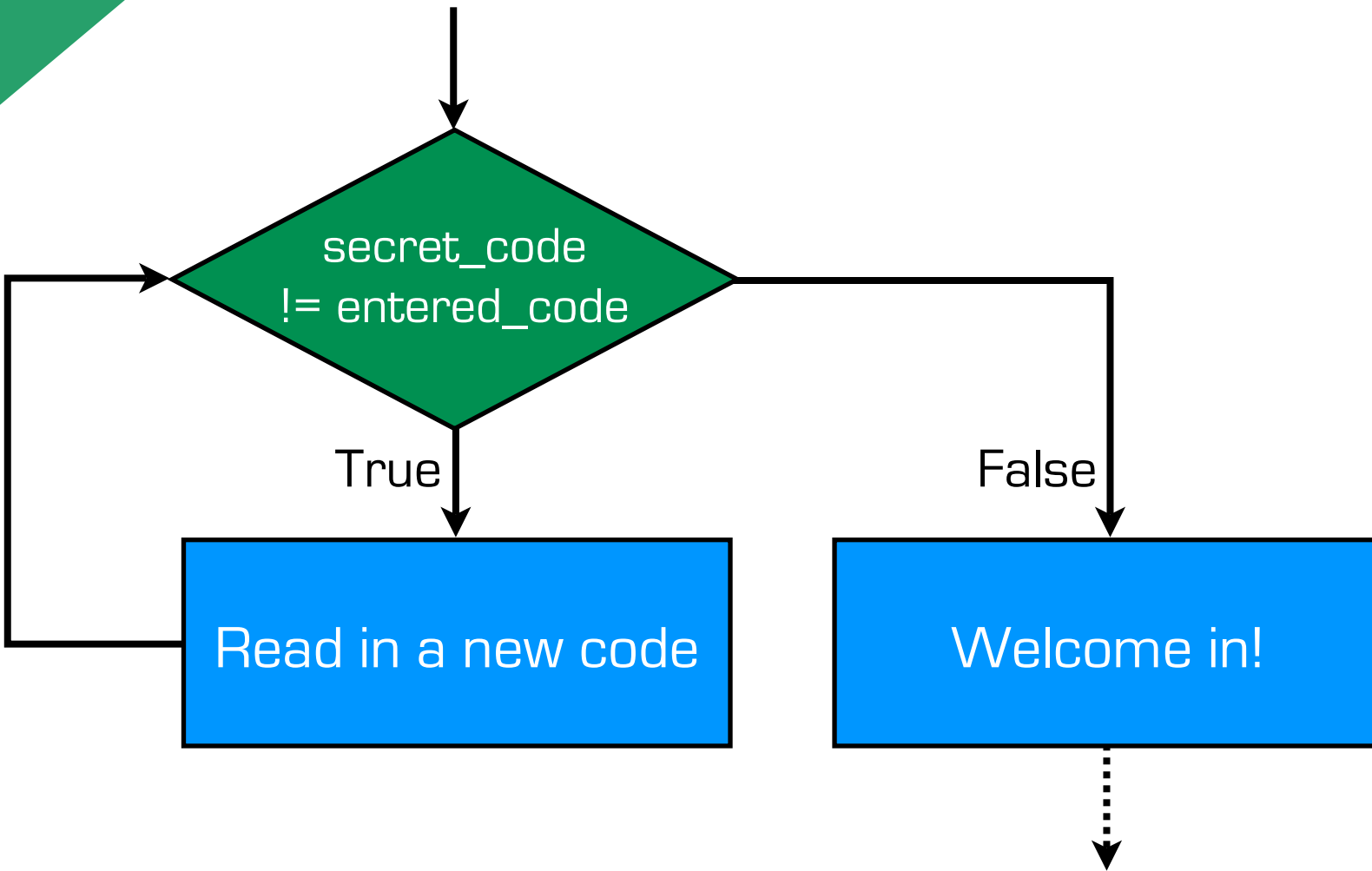
```
secret_code = 7890
entered_code = 0

while secret_code != entered_code
  puts "Please enter a secret code:"
  entered_code = gets.to_i
end

puts "Welcome in!"
```

```
Please enter a secret code:
1000
Please enter a secret code:
2321
Please enter a secret code:
7890
Welcome in!
```

Semantics



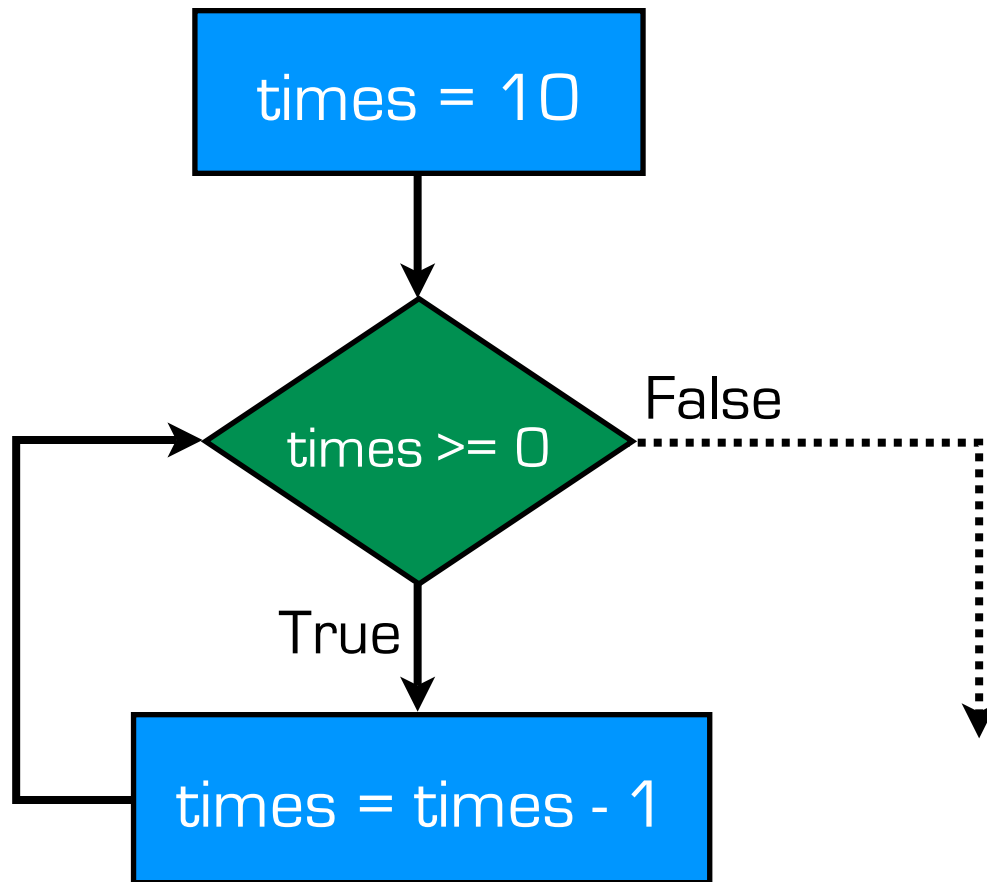
The "while" Loop

```
times = 10

while times >= 0
  puts "#{times} times left"
  times = times - 1
end
```

```
10 times left
9 times left
8 times left
7 times left
6 times left
5 times left
4 times left
3 times left
2 times left
1 times left
0 times left
```

Note: there are easier ways of doing this in Ruby



General While Loop Syntax

```
while cond  
  # Loop body  
  # Keep doing this while cond is true  
end
```

Putting It Together

```
goal = 1234
guess = -1

while goal != guess
  puts "Enter a number:"
  guess = gets.to_i

  if guess < goal
    puts "You guessed too low!"
  elsif guess > goal
    puts "You guessed too high!"
  else
    puts "Right on!"
  end
end
```

Demo

