In this lecture, we will discuss…

- Flow of Control
  - if / elsif / else
  - case
  - until / unless?
  - while / for
- What is true and what is false?
- What in the world is === ?

Flow of Control

- `if, unless, elsif, else`
- No parentheses or curly braces
- Use `end` to close flow control block

```ruby
a = 5 # declare a variable

if  a == 3
  puts "a is 3"
elsif a == 5
  puts "a is 5"
else
  puts "a is not 3 or 5"
end

# => a is 5
```

```ruby
a = 5

unless a == 6
  puts "a is not 6"
end

# => a is not 6
```

Flow of Control

- `while, until`

```ruby
a = 10

while a > 9
  puts a
  a -= 1
  # same as a = a - 1
end

# => 10
```

```ruby
a = 9

until a >= 10
  puts a
  a += 1
end

# => 9
```

Flow of Control: Modifier Form

- `if, unless, while, until` – on the same line as the statement

```ruby
# if modifier form

a = 5
b = 0

puts "One liner" if a == 5 and b == 0
# => One liner
```

```ruby
# while modifier form

times_2 = 2
times_2 *= 2 while times_2 < 100
puts times_2 # => ?
```

True / False

- `false` and `nil` objects are false

- **Everything else** is true!

```
puts "0 is true" if 0 # => 0 is true
puts "false is true?" if "false" # => false is true?
puts "no way - false is false" if false # => NOTHING PRINTED
puts "empty string is true" if "" # => empty string is true
puts "nil is true?" if "nil" # => nil is true?
puts "no way - nil is false" if nil # => NOTHING PRINTED
```

```
warning: string literal in condition
```

Triple Equal

- Triple Equal: ===
- “Equal” in its own way
  - Sometimes it’s not about being exactly equal

```
if /sera/ === "coursera"
  puts "Triple Equals"
end
# => Triple Equals

if "coursera" === "coursera"
  puts "also works"
end
# => also works

if Integer === 21
  puts "21 is an Integer"
end
# => 21 is an Integer
```

Case Expressions

- Two “flavors”
  1. Similar to a series of “`if`” statements
  2. Specify a target next to `case` and each `when` clause is compared to target

- === is called the case equality operator because it is used in precisely this case!

- **No fall-through logic**

Case Expressions

```ruby
age = 21

case # 1ST FLAVOR
  when age >= 21
    puts "You can buy a drink"
  when 1 == 0
    puts "Written by a drunk programmer"
  else
    puts "Default condition"
end
# => You can buy a drink
```

```ruby
name = 'Fisher'
case name # 2nd FLAVOR
  when /fish/i then puts "Something is fishy here"
  when 'Smith' then puts "Your name is Smith"
end

#=> Something is fishy here
```

For Loop

- Hardly used
- `each` / `times` preferred

```
for i in 0..2
  puts i
end

# => 0
# => 1
# => 2
```

Summary

- Lots of options for flow of control
- Modifier form is an interesting way to be very expressive
- Non-nil and non-false values are always true

What's next?

- Functions / Methods