```
###

	The programs in this document introduce:

		• Methods

###
```

```
######################################## P1
# A method is just a function attached to an object.

cat = {}
cat.meow = -> print("mewwww!")

cat.meow()  # methods are "called" just like functions, because they are functions.

# If you like analogies...   variable:property :: function:method

```

```
######################################## P2
# Methods can have parameters that accept arguments, just like normal functions.

cat = {}
cat.eat = (food)->
    print("I pooped "+ food)

cat.eat("kibbles")
```

```
########################################## P3
# We can define a method in an Object Literal.
# But this gets messy...

cat = {eat: (food)->  print("I pooped "+ food)} # oof
cat.eat("kibbles")
```

```coffeescript
######################################## P4
# When a method runs, it can refer to the object it is attached to.
# In JavaScript and CoffeeScript, we use the keyword 'this'.

cat1 = {}
cat1.name  = "meowz"
cat1.speak = ->
    print(this.name)    # when this code runs, 'this' refers to the 'cat1' object

cat2 = {}
cat2.name = "mittens"
cat2.speak = ->
    print(this.name)    # when this code runs, 'this' refers to the 'cat2' object


cat1.speak()
cat2.speak()
```

```coffeescript
######################################## P4b
# We can use existing functions as methods
# In other words, we can attach existing functions to objects.

genericSpeak = -> print(this.name)

cat1 = {name:"meowz",speak:genericSpeak}    # NOTE that we are NOT calling the genericSpeak function here.
cat2 = {name:"mittens",speak:genericSpeak}  # We are only refering to it.
cat3 = {name:"violet",speak:genericSpeak}   # The property 'speak' will contain the function 'genericSpeak'.

cat1.speak() # 'cat1.speak' refers to the genericSpeak function code
cat2.speak() # 'cat2.speak' also refers to the genericSpeak function code
cat3.speak() # same.
```

```
########################################## P5
# Using the 'this' keyword, a method can modify the variables of the object it belongs to.

cat = {hairballs:10}
cat.hork = ->
    this.hairballs = this.hairballs - 1
    print(this.hairballs + " remaining")


cat.hork() # notice how the method call has the same GENERAL effect
cat.hork() # but the SPECIFIC results are different
cat.hork()
cat.hork()
```

```
######################################## P6
# JavaScript (and CoffeeScript) have many built in functions.
# They are often organized by being attached to an Object (of sorts).
# For example, many mathematical functions are attached to the built in 'Math' Object.

print(Math.round(3.14)) # round a number to the closest whole number
print(Math.floor(3.6))  # round a number down
print(Math.ceil(3.1))   # round a number up (think 'ceiling')
print(Math.cos(0))      # Cosine of an angle
print(Math.random())    # Returns a number between 0 and 1

# You can find a full list of mathematical functions by googling "javacript math object"
```

```
########################################
######################################## Using CoffeeScript shortcuts
########################################


######################################## P7
# You can also define a method using CoffeeScript's Object Literal Shorthand.

cat =                   # variable declaration
    name:"mittens"      # property value pair
    age:10              # property value pair
    eat:(food)->        # method declaration
        print "I pooped " + food    # method body

print cat.name          # If we call a method with an argument, we can omit the parenthesis
print cat.age
cat.eat "kibbles"
```

```
######################################## P8
# In CoffeeScript, "this." can be replaced with "@".
# Note that the dot after "this" is also replaced.
# The shortcut is very common in CoffeeScript code.

cat =
    hairballs: 10
    hork: ->
        @hairballs = @hairballs - 1
        print @hairballs + " remaining"


cat.hork()
cat.hork()
cat.hork()
cat.hork()

########################################
######################################## End
########################################
```