

# Object-Oriented Programming, Part II

## Ruby namespace

In Ruby, the term `namespace` refers to a module the contains a group of related objects. An example of a Ruby namespace is the `Math` module.

#To retrieve a constant from the `Math` module, the scope resolution operator (`::`), should be used.

```
puts Math::PI
# => 3.141592653589793
```

#In this example, Ruby is targetting the `PI` constant from the `Math` module using the scope resolution operator, (`::`), and printing its value to the console.

## Ruby require Keyword

In Ruby, the `require` keyword is used to fetch a certain module which isn't yet presented in the interpreter. It is best practice to place this at the beginning of your code.

```
require 'date'

puts Date.today
# => 2020-04-16
```

## Ruby Module

In Ruby, a *module* contains a set of methods, constants, or classes which can be accessed with the `.` operator similarly to classes. Unlike classes, it is impossible to create instances of a Ruby module.

#A Ruby module can be created using the `module` keyword followed by the module name written in `CapitalizedCamelCase` format finalized with an `end`.

```
module MyPizza
  FAVE_TOPPING = "Buffalo Chicken"
end
```

#In this example, `myPizza` is a module that holds a constant, `FAVE_TOPPING`, set equal to the string, `Buffalo Chicken`.

## Ruby attr\_accessor Method

In Ruby, `attr_accessor`, used to make a variable both readable and writeable, is a shortcut to `attr_reader` and `attr_writer`.

```
class CollegeStudent
  attr_reader :dorm
  attr_accessor :major

  def initialize(dorm, major)
    @dorm = dorm
    @major = major
  end
end
```

#In this example, Ruby is able to only read the `@dorm` instance variable but both read and write the `@major` instance variable since it was passed to the `attr_accessor` method.