Manejo de Archivos en `ruby`

> **Objetivo**: Persistir datos más de la inherencia de la aplicación.

Archivos

Para abrir y leer archivos en `ruby`, utilizamos la clase `file` https://ruby-doc.org/core-2.4.1/File.html.

```ruby
File.open(nombre_archivo, intención)
```

Leyendo un archivo

```ruby
file = File.open('sample.txt', 'r')
contents = file.read
puts contents
file.close
```

El método `read` devuelve un string con todo el contenido dentro del archivo.

```ruby
File.open('sample.txt', 'r') { |file| puts file.read}
```

En el formato de bloque, el archivo se cierra de forma automática.

Leyendo una tabla como archivo

Producto	Precio
1	100
2	210
3	3000

Estrategia ↦ contar filas (cantidad de observaciones dentro del archivo).

1. Cambiar linea 2 por `file.read.split("\n")` para contar el quiebre de línea
2. Para mostrar la cantidad elementos en el array. `file.read.split("\n").length`

```ruby
file = File.open('sample.txt', 'r')
puts file.read
file.close
```

Readlines

> Automatiza el contenido y lo separa por líneas

se cambia el `read.split` por `.readlines`, dad que es más conciso.

```ruby
lines = []
File.open('sample.txt', 'r') { |file| lines = file.readlines }
puts lines.count
```

Promedio

```ruby
file = File.open('sample.txt', 'r')
data = file.readlines
file.close
print data

print data[0] # Producto 100

# Separando por espacio

print data[0].split(' ')

suma = 0
data.each do |prod|
    suma += prod.split(' ').last.to_i
end

puts suma/ data.length
```

Promedio con `inject`

```ruby
file = File.open('sample.txt', 'r')
data = file.readlines
file.close

suma = data.inject(0) { |acc, prod| acc + prod.split(' ').last.to_i }
puts suma / data.length
```

Scope y bloques en archivos

¿Qué está mal en el siguiente código?

```ruby
File.open('sample.txt', 'r') { |file| a = file.read }
puts a
```

> Se declara `a` dentro del bloque. Se puede dar un valor inicial fuera del bloque.

Desafío Suma

Con el siguiente archivo, sumar los valores de la tercera columna.

```
199403 73.35 71.45 72.50 36,065 5,844
199403 72.70 71.30 71.85 36,258 7,048
199403 73.10 71.40 71.55 35,938 6,608
199403 72.20 70.55 72.10 35,921 6,710
199403 72.70 70.65 72.50 35,468 9,432
199403 73.35 72.60 73.05 34,952 4,911
199403 73.35 71.60 71.70 35,563 5,455
199403 73.05 71.40 72.95 35,144 7,552
199403 73.80 72.60 73.45 34,671 6,055
199403 74.40 72.75 73.95 34,849 6,538
199403 74.60 73.25 73.70 34,517 5,785
199403 74.90 73.65 74.65 34,216 6,496
199403 74.10 72.00 72.50 33,792 9,940
```

Solución

```ruby
file = File.open('suma.txt', 'r')
data = file.readlines
file.close

suma = data.inject(0) { |catch, sum | catch + sum.split("\t")[3].to_i }
puts suma
```

Listado de palabras

> Crear un método para ver si una palabra se encuentra en un listado.

```
preliminarily
twain
unenlightened
dupr
abomasum
huntaway
father
servantless
anthologist
equidistant
pastelist
dispellable
polypoid
preforgiving
```

Solución

```ruby

f = File.open('palabras.txt', 'r')
data = f.read
f.close

# paso 1, ver si existe palabra
data.include?(palabra)

#Modularizando método

def find_word(file, word)
    f  = File.open(file, 'r')
    data = f.read
    f.close
    data.include?(word) ? puts "La palabra #{word} está en el archivo #{file}" : puts "La palabra #{word}
end
```

Solución Gonzalo

Las palabras vienen con `\n`, para eso se utiliza `.chomp`

```ruby
def scanner(filename, word)
    file = File.open(filename, 'r')
    data = file.readlines.map { |e| e.chomp }
    # data  = file.readlines.map(&:chomp) (manera alternativa)
    file.close
    data.include?(word)
end
```

Múltiples Líneas

> Lineas que contienen distintos atributos de una observación.

```
Johnson Trail
Wed, 21 Dec 2005
Spyglass Entertainment
Action
bad, good, bad, good, bad, bad, good, good, bad, bad, bad, bad
Gregorio Road
Wed, 28 Jun 2006
Embassy Pictures
Documental
good, good, good, good, bad, good, bad, good, bad, bad, bad, bad
Tremayne Way
Thu, 31 Aug 2006
Spyglass Entertainment
Musical
good, bad, bad, good, good, good, good, good, good, bad, good, bad
Nicolas Parkway
Mon, 22 Sep 1997
Five & Two Pictures
Drama
good, good, bad, bad, good, good, bad, good, bad, bad, bad, good
Alfonzo Circles
Sun, 26 Feb 2012
Walt Disney Pictures
Drama
bad, bad, good, bad, bad, good, good, good, bad, bad, good, good
Windler Turnpike
Wed, 04 Dec 2002
Metro Goldwyn Mayer
Thriller
good, good, good, good, good, bad, good, good, good, good, bad, bad
```

El primer paso es agrupar las observaciones de las películas. Se sabe que cada película contiene 5 atributos (nombre, fecha, estudio, categoría y opiniones. Se reordenará mediante el siguiente método:

```ruby
def show(movie)
    puts "#{movie[0]}: #{movie[1]}"
end

f = File.open "movies.txt"
data = file.readlines
file.close

data.each_slice(5) do |movie|
    show(movie)
end
```

Splat

El código se puede optimizar mediante el operador `*`, que permite pasar un arreglo y cada uno de sus elementos como un argumento.

```
def show(movie)
    puts "#{movie[0]}: #{movie[1]}"
end

def parse_movies(name, date, studio, category, votes)
    puts name
    puts date
    puts studio
    puts category
    puts votes
end

f = File.open "movies.txt"
data = file.readlines
file.close

data.each_slice(5) do |slice|
    parse_movies(*slice)
end
```

Contando votos

Hasta ahora, los votos son string

```ruby
def show(movie)
    puts "#{movie[0]}: #{movie[1]}"
end

def parse_movies(name, date, studio, category, votes)
    puts name
    puts date
    puts studio
    puts category
    votes = votes.split(', ').map(&:chomp)
    puts "Buenos: #{votes.count('good')} / Malos: #{votes.count('bad')}"
end

f = File.open "movies.txt"
data = file.readlines
file.close

data.each_slice(5) do |slice|
    parse_movies(*slice)
end
```

Película más votada

```ruby
votes = []
movies.each_slice(5) do |slice|
    votes << slice[4].split(", ").map(&:chomp)
end

puts votes.map{ |e| e.count('good') }

```

Desafío Pokemon

> Leer el archivo y mostrar en pantalla los pokemones tipo fuego

```
#001, Bulbasaur, Grass
#002, Ivysaur, Grass
#003, Venusaur, Grass
#004, Charmander, Fire
#005, Charmeleon, Fire
#006, Charizard, Fire
#007, Squirtle, Water
#008, Wartortle, Water
#009, Blastoise, Water
#010, Caterpie, Bug
#011, Metapod, Bug
#012, Butterfree, Bug
#013, Weedle, Bug
#014, Kakunaa Bug
#015, Beedrill, Bug
#016, Pidgey, Normal
#017, Pidgeotto, Normal
#018, Pidgeot, Normal
#019, Rattata, Normal
#020, Raticate, Normal
#021, Spearow, Normal
#022, Fearow, Normal
#023, Ekans, Poison
#024, Arbok, Poison
#025, Pikachu, Electric
#026, Raichu, Electric
```

```ruby
f = File.open('pokemones.txt', 'r')
data = f.readlines
file.close

data.each do |poke|
    puts pokemon.split(', ').map(&:chomp) if data[2] == 'Fire'
end

```

Escribiendo archivos

```ruby
f = File.open('lorem.txt', 'w')
f.puts 'Lorem ipsum dolor sit amet'
f.close
```

```ruby
File.open('sample.txt', 'w'){ |file| file.puts 'lorem ipsum' }
```

```ruby
content = gets.chomp

file = File.open('secreto.txt', 'w')
file.puts content
file.close
```

Integración

> **Objetivo:** El usuario ingresa un producto y precio, se debe cambiar el precio del producto ingresado en el archivo.

```
Product1    100
Product2    150
Product3    350
```

```ruby
file = File.open('products.txt', 'r')
products = file.readlines
file.close

print products

products = products.map{ |e| e.split(' ') }.to_h
products[products_name] = price

file = File.open('products.txt', 'w')
products.each do |key, value|
    file.puts "#{key} #{value}"
```

Append

> Añadir, no borrar.

```ruby
file = File.open('products.txt', 'a')
```

File exists

```ruby
if File.exists?(filename)
    puts "#{filename} exists"
end
```

Excepciones

> Se levantan cuando sucede algo inesperado:
>
> 1. Archivos inexistentes
> 2. Métodos inexistentes
> 3. Among others...

Las excepciones se levantan de forma automática. Ruby las define en el prompt `NameError:`. Podemos definir excepciones de forma manual por medio de `raise`:

```ruby
def inverse(x)
    raise ArgumentError, "Argument is not numeric" unless x.is_a? Numeric 1.0 / x
end

puts inverse(2)
puts inverse('not a number')
```

Tipos de excepciones en ruby

- fatal *usada de forma interna por Ruby*
- NoMemoryError
- ScriptError
  - LoadError
  - NotImpplementedError
  - SyntaxError
- SecurityError
- SignalException
  - Interrupt
- StandardError
  - ArgumentError
  - FiberError
  - IndexError
    - KeyError

        - StopIteration
    - IOError
    - LocalJumpError
    - NameError
        - NoMethodError
    - RangeError
        - FloatDomainError
    - RegexpError
    - RuntimeError
    - SystemCallError
        - *excepciones dependientes del sistema* (`Errno:###`)
    - ThreadError
    - TypeError
    - TypeError
    - ZeroDivisionError
- SystemExit
- SystemStackError

Manejando excepciones

> Utilizar `rescue`, permite continuar si ocurre una excepción.

```
def x
        2 + 2
        raise
rescue
        puts "hola"
end

x
```

Las excepciones resultan de utilidad para los archivos en ruby:

```ruby
begin
    File.open('asdf.rb', 'r') do |f1|
        while line = f1.gets
            puts line
        end
    end

    File.open('test.rb', 'w) do |f2|
        f2.puts "A la grande le puse cuca"
    end
rescue Exception => msg
    puts msg
end
```

Antipatrones

> Práctica de código abusiva que resulta intuitiva la primera vez. ↦ `begin rescue all`