Ruby Project 3 File

System and Data Parsing

Part 1

Create a ruby script that does the following:

1. **Writes “Hello World” to a file, 2. Reads the text file (created in step 1) and prints its contents to the screen and 3.Appends the text file (created in step 1) and adds (on a separate line) “It’s going to be a great day!”**

# If you open up the command prompt (I’m using a windows not a Mac) then

Microsoft Windows [Version 6.2.9200]

(c) 2012 Microsoft Corporation. All rights reserved.

C:\Users\Sedinirina>cd desktop

C:\Users\Sedinirina\Desktop>cd Project3

C:\Users\Sedinirina\Desktop\Project3>irb --simple-prompt

>> file = File.new('test.txt', 'w') # Here we are going to create a file that has Hello word in it, and this file is on write mode

=> #<File:test.txt>

>> file.puts "Hello world"

=> nil

>> file.close # this file won’t close unless you close the file by using file.close

=> nil

>> file = File.new('test.txt', 'r') # Here we are going to open the file in read and the append these words and the close it

=> #<File:test.txt>

>> file.gets

=> "Hello world\n"

>> file.close

=> nil

>> file = File.new('test.txt','a')

=> #<File:test.txt>

>> file << "It's going to be great day!"

=> #<File:test.txt>

>> file.close

=> nil

>>

**4. Your script should create a new directory called “my\_new\_directory”**

C:\Users\Sedinirina\Desktop>cd sedy # First, you have to go where you want to create the new directory

C:\Users\Sedinirina\Desktop\sedy>irb --simple-prompt

>> Dir.mkdir("my\_new\_directory")

=> 0

>>

**5. Your script should check to see if a file exists and if it does print “Yes! this file exists!”**

if Dir["C:\Users\Sedinirina\Desktop\sedy\my\_new\_directory"] != nil

puts 'Good Job! The file exists!’

else

puts 'Sorry! There is no such directory'

end

so the result will look like this :

Good Job the file exists!

=> nil

# And there is another way of knowing if the directory exists, by using File.exists?

>> File.exists? File.expand\_path('C:\Users\Sedinirina\Desktop\sedy')

=> true

**7. Name your script “basic\_io.rb”**

# to name a file you just need to use File.rename(“oldfile.txt”,”newfile.rb”)

# And in this case, you are not supposed to be reading the file or append something or any kind of dealing inside the file but this way will work outside of the file , it means you use file.close :

>> file = File.new('test.txt', 'w')

=> #<File:test.txt>

>> file.puts "Hello world"

=> nil

>> file.close

=> nil

>> file = File.new('test.txt', 'r')

=> #<File:test.txt>

>> file.gets

=> "Hello world\n"

>> file.close

=> nil

>> file = File.new('test.txt','a')

=> #<File:test.txt>

>> file << "It's going to be great day!"

=> #<File:test.txt>

>> file.close

=> nil

>>File.rename(“test.txt”,”basic\_io.rb”)

=> 0 # So here if you look on the my\_new\_directory then you will see that the “test.txt” has been changed to “basic\_io.rb”

**Part 2:**

**1. Now we are going to write the following to a file: (and save the filename as makes\_models.csv)**

# Right, let’s take a look on this Part 2, and we will use a text editor called Komodo Edit 7

C:\Users\Sedinirina\Desktop\sedy>irb --simple-prompt

>> file = File.new('makes\_models.csv', 'w')

=> #<File:makes\_models.csv>

>> file.puts "Make, Model"

=> nil

>> file.puts "Chevrolet, Malibu"

=> nil

>> file.puts "Ford, Mustang"

=> nil

>> file.puts "Ford, Taurus"

=> nil

>> file.print "Chevrolet, Cobalt\n"

>> file.close

=> nil

>>

**Now we are going to read the file and prints the following to the screen: “Make: [MAKE\_NAME] Model: [MODEL]” (replace MAKE\_NAME and MODEL with actual content)**

C:\Users\Sedinirina\Desktop\sedy\my\_new\_directory>irb --simple-prompt

>> file = File.open("makes\_models.csv", "r")

=> #<File:makes\_models.csv>

>> while line = file.gets

>> next if file.lineno == 1

>> line = line.split(",")

>> puts "Make : #{line[0]} -- Model: #{line[1]}"

>> end

Make : Chevrolet -- Model: Malibu

Make : Ford -- Model: Mustang

Make : Ford -- Model: Taurus

Make : Chevrolet -- Model: Cobalt

=> nil

>>

**3. Here our script should print out how many lines it processed (i.e. processed 5 lines). This value**

**should come from our program and not “hard coded” ( Hey Steve! I know something is wrong here please help )**

C:\Users\Sedinirina\Desktop\sedy\my\_new\_directory>irb --simple-prompt

>> file = File.open("makes\_models.csv", "r")

=> #<File:makes\_models.csv>

>> while line = file.gets

>> next if file.lineno == 1

>> line = line.split(",")

>> puts "Make : #{line[0]} -- Model: #{line[1]}"

>> end

Make : Chevrolet -- Model: Malibu

Make : Ford -- Model: Mustang

Make : Ford -- Model: Taurus

Make : Chevrolet -- Model: Cobalt

=> nil

>> line\_num=0

=> 0

>> File.open("make\_models.csv", "r") .each do |line|

?> puts " processed #{line\_num += 1} Lines"

>> end

processed 1 Lines

processed 2 Lines

processed 3 Lines

processed 4 Lines

processed 5 Lines

=> #<File:makes\_models.csv>

**4. Your script should then print out how large the file was that you processed.**

C:\Users\Sedinirina\Desktop\sedy\my\_new\_directory>irb --simple-prompt

>> file = File.open("makes\_models.csv", "r")

=> #<File:makes\_models.csv>

>> while line = file.gets

>> next if file.lineno == 1

>> line = line.split(",")

>> puts "Make : #{line[0]} -- Model: #{line[1]}"

>> end

Make : Chevrolet -- Model: Malibu

Make : Ford -- Model: Mustang

Make : Ford -- Model: Taurus

Make : Chevrolet -- Model: Cobalt

=> nil

>> line\_num=0

=> 0

>> File.open("make\_models.csv", "r") .each do |line|

?> puts " processed #{line\_num += 1} Lines"

>> end

processed 1 Lines

processed 2 Lines

processed 3 Lines

processed 4 Lines

processed 5 Lines

=> #<File:make\_models.csv>

>> File.size("make\_models.csv") # I’m lost here I need help

=> 78

# this is what I thought but it doesn t work so let me just give it to you so that you can correct me

max\_lines = 5 # it took time but still doesn t work so this kind of my first time to deal with this file size

File.open("make\_models.csv", "r")

while line = file.gets

next if file.lineno == 1

puts "Lines processed: #{max\_lines - 1}"

print "File Size: #{File.size("make\_models.csv")} MB"

end

end

**Part 3:**

1. **right now we are going to make a copy of our script from Part 2 and name it “ford\_make\_models.rb”**

C:\Users\Sedinirina\Desktop\sedy\my\_new\_directory>irb --simple-prompt

>> File.rename( "make\_models.rb" , "ford\_make\_models.rb" )

=> 0

>>

1. **2. Modify your script to store your make/models in a hash**

C:\Users\Sedinirina\Desktop\sedy\my\_new\_directory>irb --simple-prompt

store = {"Chevrolet"=>["Malibu","Cobalt"], "Ford"=>["Mustang","Taurus"]}

store.each do |keys, values|

puts "Maker: #{keys} -- Models: #{values.upcase}"

end

Maker: Chevrolet -- Models: ["Malibu", "Cobalt"]

Maker: Ford -- Models: ["Mustang", "Taurus"]

=> {"Chevrolet"=>["Malibu", "Cobalt"], "Ford"=>["Mustang", "Taurus"]}

>>

1. **3-Remove any items from the hash that are not a “Ford” make**

C:\Users\Sedinirina\Desktop\sedy\my\_new\_directory>irb --simple-prompt

>> store = {"Chevrolet"=>["Malibu","Cobalt"], "Ford"=>["Mustang","Taurus"]}

=> {"Chevrolet"=>["Malibu", "Cobalt"], "Ford"=>["Mustang", "Taurus"]}

>> filter\_key = "Ford"

=> "Ford"

>> store.keys.each do | key |

?> next if key == filter\_key

>> store.delete key

>> end

=> ["Chevrolet", "Ford"]

>> store

=> {"Ford"=>["Mustang", "Taurus"]}

>>

**4-Create a file (called fords.csv) and write a file that only contains “Ford” makes/models**.

C:\Users\Sedinirina\Desktop\sedy\my\_new\_directory>irb --simple-prompt

>> file = File.new('fords.csv', 'w')

=> #<File:fords.csv>

>> file.puts "Make, Model"

=> nil

>> file.puts "Ford, Mustang"

=> nil

>> file.prints "Ford, Taurus\n"

=> nil

>> file.close

=> nil

>>