File:

Every day we create, open, read, write, edit, rename, delete, close in some files,

In ruby, without software, we can do above things easily with file instance\_methods.

File Methods

new, open, close, delete, rename, puts, syswrite, write, read, readlines

Note: #You must open file before reading and after read close file.

|  |  |
| --- | --- |
| Mode | Description |
| r | Read-only mode. |
| r+ | Read-write mode. |
| w | Write-only mode. |

f = File.new("file.rb", "w")

f.close

my\_file = File.new("C://sites/data.rb", "w")

# you must close file after , creating, opening

ff = File.open("file1.rb", "w")

ff.close

Rename File

File.rename("old\_file.rb", "new\_file.rb")

Delete File

File.delete("file\_name.rb")

#write content in file using puts

f = File.new("file.rb", "w")

f.puts('something')

f.close

f1 = File.open('file1.rb', 'w') do |line|

line.puts('I am srinivas and ror trainer')

end

# The difference is that the File.open method can be associated with a block.

Read Multiple Line file:

fname = "srinivas.rb"  
File.readlines(fname).each do |line|  
 puts line  
end

Check file is exist or not:

File.exists?("temp.txt")=>true

File expand\_path:

puts "The value of `\_\_FILE\_\_` (the name of this Ruby file) is:"

puts "=> #{\_\_FILE\_\_}"

puts "The absolute path of this Ruby file is:"

puts "=> #{File.expand\_path(\_\_FILE\_\_)}"

puts "The relative path from this file to `args.rb` is:"

puts "=> #{"./args.rb"}"

puts "Does `File.exist?` return true for the relative path `./args.rb`?"

puts "=> #{File.exist?("./args.rb")}"

puts "The absolute path to `args.rb` is:"

puts "=> #{File.expand\_path("../args.rb", \_\_FILE\_\_)}"

puts "Does `File.exist?` return true for the absolute path of `args.rb`?"

puts "=> #{File.exist?(File.expand\_path("../args.rb", \_\_FILE\_\_))}"

puts "The tilde character '~' represents your home directory."

puts "`File.expand\_path('~/')` is:"

puts "=> #{File.expand\_path('~/')}"

puts "A single period '.' represents the current directory."

puts "`File.expand\_path('.')` is:"

puts "=> #{File.expand\_path('.')}"

puts "Two periods '..' represents the parent directory."

puts "`File.expand\_path('..')` is:"

puts "=> #{File.expand\_path('..')}"