**Ruby Documentation**

Ruby as default is included in MacOS

**Data type**

string 🡪 common string 🡪 name = “Asep Budi”

number 🡪 can be int or float 🡪 age = 7.5, num = -8

boolean 🡪 common Boolean 🡪 isMale = true

nil 🡪 no value

Array 🡪 common array 🡪 myArr = Array.new

# 🡪 for commenting

**Print**

puts “text” 🡪 like println

name = “asep”

puts (“My name is “ + name)

puts “New \n Line”

puts 1 + 3

print “text” 🡪 like print

**String method**

phrase = “Asep Budi”

phrase.upcase() 🡪 capitalize all text

phrase.downcase() 🡪 downcase all text

phrase.strip() 🡪 remove space before and after text

phrase.length() 🡪 length of the text

phrase.include? “Budi” 🡪 return true, check if input in text

phrase[i] 🡪 return the i-th char

phrase[I,j] 🡪 return the i-th to (not include) j-th char

phrase.index(“A”) 🡪 return 0, index of input char

phrase.to\_i 🡪 convert char to int

phrase.to\_f 🡪 convert char to float

**Number method**

num = 8

num.abs() 🡪 absolute number

num.round() 🡪 rounding nearest

num.ceil() 🡪 rounding up

num.floor() 🡪 rounding down

Math.sqrt(num) 🡪 using Math library

Puts 10 / 7 🡪 return 1, to get int output, use int input

Puts 10 / 7.0 🡪 return 1.42, to get float output, use float input

num.to\_s 🡪 convert number to string

**Take input from terminal**

Single input:

inputVar = gets

Put (“you inputting “ + inputVar)

Multiple input:

inputVar1 = gets.chomp()

inputVar2 = gets.chomp()

Put (“you inputting “ + inputVar1 + “ and “ + inputVar2)

**Array**

myArr = Array.new

myArr= Array[“Asep”, 1,true]

myArr[0] 🡪 return “Asep”

myArr[-1] 🡪 return true

myArr[0, 2] 🡪 return the i-th to (not include) j-th element

myArr.length() 🡪 return length of array

myArr.include? “Asep” 🡪 return true, check if input in array

myArr.reverse 🡪 reverse the array, void method with no return value

myArr.sort() 🡪 sorting array

**Hash**

myHash = {

“name” => “Asep”,

“age” => 18,

:address => “bandung”,

1 => “satu”

}

myHash[“name] 🡪 return “Asep”

myHash[:address] 🡪 return “bandung”

myHash[1] 🡪 return “satu”

**Function/method**

Without return value

def myFunc(name=”default name”, age=-1)

puts (“My name is ” + name + “, age “ + age.to\_s)

end

myFunc(“Asep”, 18)

With return value

def myFunc(param1, param2)

return param1 \* param1, param2 // we thought it return tuple..

end

myFunc (3)[1] // the output is not tuple, but array!

**Conditional**

If-else clause, Boolean operation -> or, and, !

if isMale and isTall

puts “male and tall”:

elsif isMale and !isTall

puts “male and not tall”

else

puts “not male and not tall”

end

Case clause:

case day

when “1”

num\_name = “one”

when “2”

num\_name = “two”

when “mon”

num\_name = “three”

end

**Loop**

While loop:

index = 1

while index <= 5

puts index

index += 1

end

For loop:

index = [1, 2, 3, 4, 5]

for i in index

puts i

end

index = [1, 2, 3, 4, 5]

index.each do |i|

puts i

end

for i in 0..5

puts i // i will be 0 1 2 3 4 5

end

6.times do |i|

puts I // i will be 0 1 2 3 4 5

end

**Opening File**

File.open(“folder/file\_name.txt”) 🡪 input is file directory and read mode

End

File.open(“folder/file\_name.txt”, “r”) do |file| 🡪 “file” variable store the File

puts file.readchar() 🡪 read per char

puts file.readline() 🡪 read per line, returned as array

end

file = File.open(“folder/file\_name.txt”)

file.close()

**Handling Error**

begin // every code that possibly return error, begin with “begin”

num = 10 / 0

rescue ZeroDivisionError => e

puts e

end

**OOP**

Without constructor:

class Book

attr\_accessor :title, :pages

end

book1 = Book.new()

book1.title = “Harry Potter”

book1.pages = 1000

With constructor:

class Book

attr\_accessor :title, :pages

def initialize(title, pages) // this initialize method always called when instantiation

@title = title

@pages = pages

end

end

book1 = Book.new(“Harry Potter”, 1000)

With methods:

class Student

attr\_accessor :name, :major, :gpa

def initialize(name, major, gpa)

@name = name

@major = major

@gpa = gpa

end

def has\_honors

if @gpa >= 3.5

return true

end

return false

end

end

student1 = Student.new(“Asep”, “Art”, 2.7)

puts student1.has\_honors

Inheritance:

class StudentMale < Student

end

**Modules**

In module/library file:

Module Tools=

def sayhi(name)

puts “hello #{name}”

end

def sayhi(name)

puts “hello #{name}”

end

end

In main file:

include Tools

Tools.sayhi(“Asep”)

To include all information from a file, in main file:

require\_relative “useful\_tools.rb”

Include Tools

Tools.sayhi(“Asep”)