**WDI - Day 3 Notes**

**AGENDA**

**1)**

----------------------------------------------------------------------------------------------------------------------

**1)**

Should make sub-problems out of larger problems

Division

* Dividing by zero (must use float to get 🡪 infinity)
* can use guard clause

def divide (num1, num2)

return infinity if num2 == 0

else num1.to\_f / num2.to\_f

end

def main\_loop

def process\_input

choice = **get\_user\_choice**

puts “You chose: # {choice}”

case choice

when “a”

puts “addition!”

num\_hash = gets.nums

prints “your answer is: “

puts add(num\_hash[:num1],num\_hash[:num2])

when “s”

puts “subtraction!”

num\_hash = gets.nums

prints “your answer is: “

puts subtract(num\_hash[:num1],num\_hash[:num2])

end

puts “do you want to try again? (y)es or (n)o ”

answer = gets.chomp

if answer != “n”

**ask**

end

end

def **get\_user\_choice**

print “enter your choice: (a)dd, (s)ubtract?

input = gets.chomp

return input

end

def get\_numbs

puts “enter number 1: “

num1 = gets.chomp

puts “enter number 2: “

num2 = gets.chomp

return {

: num1 => num1,

:num2 => num2

}

end

When you think about function: use the function flow chart:

Argument 🡪 function 🡪 return

(input)

return stores value 🡪 gets value out of function

* return can only return one thing
* that thing can be a number of things 🡪 hash or array

**rspec**

rspec allows you to check individual functions without adding them to the end of your program and having to run through entire program in ruby.

**LOOPS**

To execute a block of code several times

For Loops

for x in (1..10)

puts x += 1

end

x = variable

(range)

.. inclusive … not inclusive

puts x =1 for range (1..10)

String Concatenate in For Loops

for x in (1..10)

puts “This is the number #{x}” string concatenate

end

While Loops

x = 1

while (x < 11) while loop

puts x

**x += 1**

end

Arrays in Loops

array = ["wednesday", "macbook", "coffee", "lunch"]

for x in (0..3)

puts array[x]

end

array = ["wednesday", "macbook", "coffee", "lunch"]

for x in (0..array.length)

puts array[x]

end

array = ["wednesday", "macbook", "coffee", "lunch"]

for index in (0..array.length)

puts array[index]

end

array = ["wednesday", "macbook", "coffee", "lunch"]

for item in (array)

puts item

end

Using.each with an array to Loop

array = ["wednesday", "macbook", "coffee", "lunch"]

array.each do |item|

puts item

end

hash = {Monday: “52 degrees”, Tuesday: “60 degrees”, Wednesday: “56 degrees”}

hash.each do |key, value|

puts key

puts value

end

hash.each do |k, v|

puts k

puts va

end

Hash you do not need an order number to return value because you have the key value (which keys into the variable)

array = ["wednesday", "macbook", "coffee", "lunch"]

array.each\_with\_index do |item, index|

puts “#{item} is at position #{index} in this array”

end

Loading files in other files:

require\_relative '../helper\_functions.rb' 🡪

(method) (relative location/file)

* Can not export local variables out of the file.

$x 🡪 global variables

* you can export these

Anonymous Function

[“Green “, “Red”, “Blue”].each do |color|

# any code goes here

end

|\_\_\_\_\_

^Anonymous function 🡪 not defined 🡪 not re-callable 🡪 inherits the scope above it (essential for Java programming)

def sugary

[“green”, “red”, “blue”].each { |c|

puts c

}

end

sugary

Matrix

$matrix = [[“1a”, “1b”, “1c”],

[“2a”, “2b”, “2c”]

[“2a”, “2b”, “2c”]

]

def print\_matrix

$matrix.each do |rows|

rows.each do |col|

puts col

end

end

end

Pry command:

$matrix = [0] [1]

row column

Hash vs. Array 🡪 fork vs. collection

.class 🡪 convert to class

janes\_favorite\_colors = [“red”, “blue”, “green”]

omars\_favorite\_colors = [“purple”, “yellow”, “red”]

.index 🡪 allows you to look in array and tell you the corresponding order number

janes\_favorite\_colors & omars\_favorite\_colors

🡪 intersection of two arrays

janes\_favorite\_colors | omars\_favorite\_colors

🡪 merges two arrays

janes\_favorite\_colors - omars\_favorite\_colors

🡪 mutually exclusive or extracts any overlap

<< push

**Hashes and Arrays**