## Fancy flatten

class Array

def my\_flatten

self.inject([]) do |acc, ele|

if ele.is\_a?(Array)

acc.concat(ele.my\_flatten)

else

acc.push(ele)

end

end

end

end

arr = [1,[2,3],[[4]]]

p arr.my\_flatten

## Build a hash

data = [[:student, "Darren Eid"], [:course, "Advanced Potions"], [:grade, "D-"]]

data\_object = data.inject({}) do |acc, ele|

acc[ele.first] = ele.last

acc

end

p data\_object

## Filter

arr = [10, 20, 30, 5, 7, 9, 3]

p (arr.inject([]) do |acc, ele|

acc << ele if ele.odd?

acc

end)

## General Algorithms

def factors(n)

(1..Math.sqrt(n)).inject([]) do |acc, ele|

q, r = n.divmod(ele)

r == 0 ? acc += [ele, q] : acc

end

end

def common\_factors(\*nums)

end

# p factors(65)

## #inject

# Enumerable method

# accumulator and current element

# accumulator is set to return value of block

# set initial value of accumulator

# will default to first value

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

p arr.inject(&:+)

# at each step, acc + ele

arr.inject do |acc, ele|

acc + ele

end