def create\_array\_of\_numbers(num)

my\_array = []

100.times do

if num % 3 == 0 && num % 5 == 0

my\_array << "Greene"

elsif num % 3 == 0

my\_array << "County"

elsif num % 5 == 0

my\_array << "Greene County"

elsif num == 77

my\_array << "Snowmageddon"

else

my\_array << num

end

num = num + 1

end

my\_array

end

my\_returned\_array = create\_array\_of\_numbers(1)

puts my\_returned\_array

def create\_array\_of\_numbers(num)

my\_array = []

100.times do

my\_array << num

num = num + 1

end

my\_array

end

my\_returned\_array = create\_array\_of\_numbers(1)

puts my\_returned\_array

def one

1

end

def greene\_county\_array(num)

my\_array = []

100.times do

if num % 3 ==0 && num % 5 == 0

my\_array<<"Greene County"

elsif num % 3 == 0

my\_array<<"Greene"

elsif num == 5

my\_array<<"County"

elsif num == 20

my\_array<<"County"

else

my\_array<<num

end

num=num+1

end

my\_array

end

def create\_array\_of\_numbers(num)

my\_array = []

100.times do

if num % 3 == 0 && num % 5 == 0

my\_array << "Greene County"

elsif num % 3 == 0

my\_array << "Greene"

elsif num % 5 == 0

my\_array << "County"

else

my\_array << num

end

num = num + 1

end

my\_array

end

my\_returned\_array = create\_array\_of\_numbers (1)

puts my\_returned\_array

puts "The first element in the array is #{my\_returned\_array[0]}"

greene\_counts = my\_returned\_array.count("Greene")

puts "There are #{greene\_counts} numbers divisible by only 3"

def create\_greene\_county\_array()

array = []

num = 1

100.times do

if num % 3 == 0 && num % 5 == 0

array << "Greene County"

elsif num % 3 == 0

array << "Greene"

elsif num % 5 == 0

array << "County"

else

array << num

end

num = num + 1

end

array

end

def create\_fizz\_buzz\_array(gcarray)

gcarray.map do |x|

if x == "Greene County"

"Fizz Buzz"

elsif x == "Greene"

"Fizz"

elsif x == "County"

"Buzz"

else

x

end

end

end

require "minitest/autorun"

require\_relative "fizz\_buzz.rb"

class TestFizzBuzz < Minitest::Test

def test\_array\_has\_100\_elements

my\_returned\_array = create\_greene\_county\_array()

assert\_equal(100, my\_returned\_array.count)

end

def test\_first\_element\_in\_array\_is\_one

my\_returned\_array = create\_greene\_county\_array()

assert\_equal(1,my\_returned\_array.first)

end

def test\_second\_element\_in\_array\_is\_two

my\_returned\_array = create\_greene\_county\_array()

assert\_equal(2,my\_returned\_array[1])

end

def test\_98\_element\_in\_array\_is\_98

my\_returned\_array = create\_greene\_county\_array()

assert\_equal(98,my\_returned\_array[-3])

end

def test\_3\_is\_greene

my\_returned\_array = create\_greene\_county\_array()

assert\_equal("Greene",my\_returned\_array[2])

end

def test\_5\_is\_county

my\_returned\_array = create\_greene\_county\_array()

assert\_equal("County",my\_returned\_array[4])

end

def test\_all\_numbers\_divisible\_by\_3

my\_returned\_array = create\_greene\_county\_array()

assert\_equal("Greene",my\_returned\_array[5])

assert\_equal("Greene",my\_returned\_array[-2])

assert\_equal("Greene",my\_returned\_array[32])

end

def test\_15\_is\_greene\_county

my\_returned\_array = create\_greene\_county\_array()

assert\_equal("Greene County",my\_returned\_array[14])

end

def test\_99\_is\_greene

my\_returned\_array = create\_greene\_county\_array()

assert\_equal("Greene",my\_returned\_array[-2])

end

def test\_new\_array\_contains\_100\_elements

greene\_county\_array = create\_greene\_county\_array()

fizz\_buzz\_array = create\_fizz\_buzz\_array(greene\_county\_array)

assert\_equal(100,fizz\_buzz\_array.count)

end

def test\_first\_element\_in\_array\_is\_1

greene\_county\_array = create\_greene\_county\_array()

fizz\_buzz\_array = create\_fizz\_buzz\_array(greene\_county\_array)

assert\_equal(1,fizz\_buzz\_array.first)

end

def test\_3\_is\_fizz

greene\_county\_array = create\_greene\_county\_array()

fizz\_buzz\_array = create\_fizz\_buzz\_array(greene\_county\_array)

assert\_equal("Fizz",fizz\_buzz\_array[2])

end

def test\_5\_is\_buzz

greene\_county\_array = create\_greene\_county\_array()

fizz\_buzz\_array = create\_fizz\_buzz\_array(greene\_county\_array)

assert\_equal("Buzz",fizz\_buzz\_array[4])

end

def test\_if\_number\_divisible\_by\_3

greene\_county\_array = create\_greene\_county\_array()

fizz\_buzz\_array = create\_fizz\_buzz\_array(greene\_county\_array)

assert\_equal("Fizz",fizz\_buzz\_array[5])

assert\_equal("Fizz",fizz\_buzz\_array[-2])

assert\_equal("Fizz",fizz\_buzz\_array[32])

end

def test\_if\_number\_divisible\_by\_5

greene\_county\_array = create\_greene\_county\_array()

fizz\_buzz\_array = create\_fizz\_buzz\_array(greene\_county\_array)

assert\_equal("Buzz",fizz\_buzz\_array[4])

assert\_equal("Buzz",fizz\_buzz\_array[9])

assert\_equal("Buzz",fizz\_buzz\_array[19])

end

end