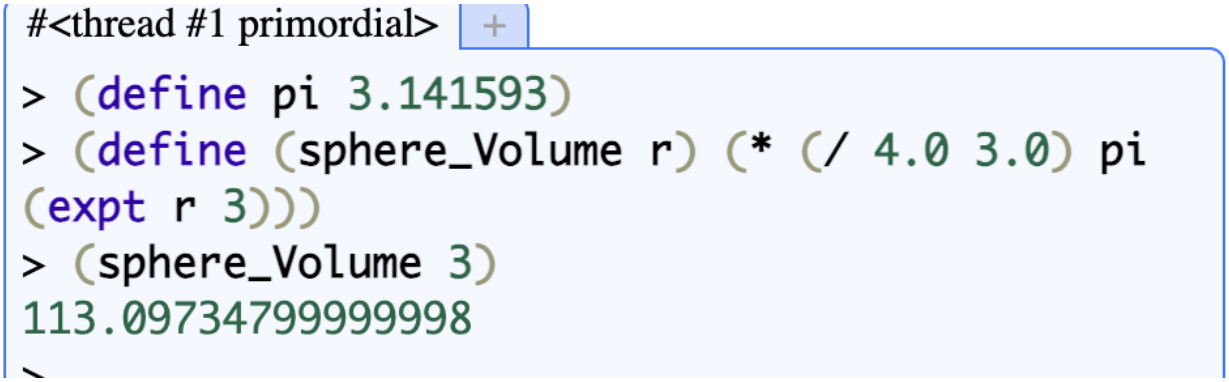
Assignment #4 Functional Programming and Scripting Languages

Question 1: Given the volume formula of a sphere

a. Scheme Sources code

```
> (define pi 3.141593)
> (define (sphere_Volume r) (* (/ 4.0 3.0) pi (expt r 3)))
> (sphere_Volume 3)
```

Image of the output generated when the radius is set to 3.

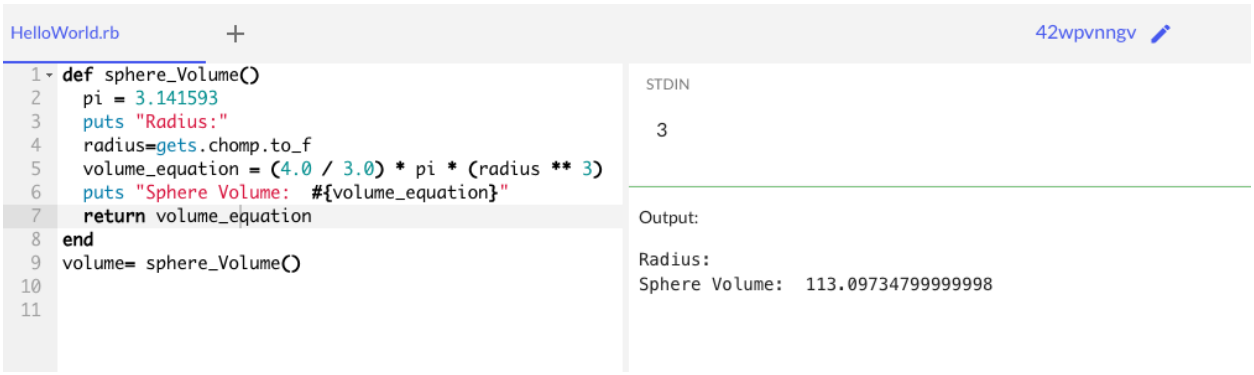


b. Ruby Source code

```
def sphere_Volume()
  pi = 3.141593
  puts "Radius:"
  radius=gets.chomp.to_f
  volume_equation = (4.0 / 3.0) * pi * (radius ** 3)
```

```
  puts "Sphere Volume:  #{volume_equation}"
  return volume_equation
end
volume= sphere_Volume()
```

Image of the output generated when the radius is set to 3.

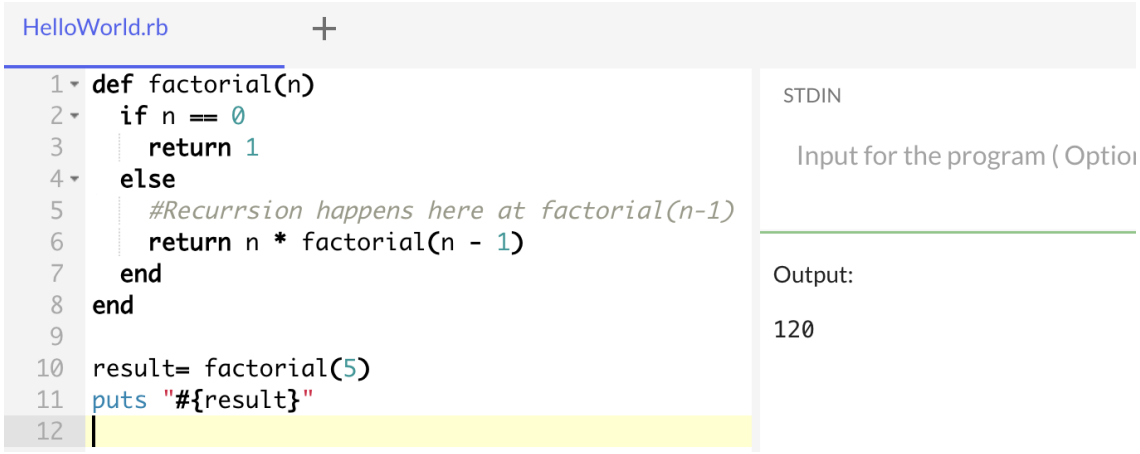


Question 2:

a. Recursion Source code and the Output

```
def factorial(n)
  if n == 0
    return 1
  else
    #Recursion happens here at factorial(n-1)
    return n * factorial(n - 1)
  end
end

result= factorial(5)
puts "#{result}"
```



b. Loop Source code and the Output

```
def factorial_loop(n)
  if n == 0
    return 1
  end

  result = 1
  i = n

  while i > 1
    result *= i
    i -= 1
  end

  return result
end

result = factorial_loop(1)
puts result
```

HelloWorld.rb +

```ruby
def factorial_loop(n)
  if n == 0
    return 1
  end

  result = 1
  i = n

  while i > 1
    result *= i
    i -= 1
  end

  return result
end

result = factorial_loop(5)
puts result
```

STDIN

Input for the program (Optio

Output:

```
120
```

c. Class

```ruby
class FindFactorial
  def initialize(factorial_number)
    @factorial_number = factorial_number
  end

  def factorial()
    n = @factorial_number
    if n == 0
      return 1
    end

    result = 1
    i = n

    while i > 1
      result *= i
```

```
      i -= 1
    end

    return result
  end
end

find = FindFactorial.new(5)
result = find.factorial
puts result
```


```ruby
class FindFactorial
  def initialize(factorial_number)
    @factorial_number = factorial_number
  end

  def factorial()
    n = @factorial_number
    if n == 0
      return 1
    end

    result = 1
    i = n

    while i > 1
      result *= i
      i -= 1
    end

    return result
  end
end

find = FindFactorial.new(5)
result = find.factorial
puts result
```

Question 3:

Scripting Language

a. **Error handling**

   Scripting languages have inbuilt datasets that can catch errors. There are also libraries that handle missing values and incorrect formats. Functional programming is also good for catching errors. In functional programming, the code is executed sequentially, meaning if you encounter an error, you must fix it before proceeding to the next line of code. Object-oriented programming has embedded error handling, but it can make the code harder to read.

b. **Suitability for Handling Complex Data Structures**

   Scripting languages excel at handling complex data structures because they have built-in data types such as dictionaries and lists. Additionally, some scripting languages are dynamically typed, meaning they can mix different data types within data structures. Functional programming is satisfactory, although it can handle complex data structures; it doesn't have built-in libraries to read complex datasets. Object-oriented programming is the best when it comes to handling complex data structures because of polymorphism, but it increases the size of the dataset.

   c. Performance

   Scripting languages are slow when it comes to performance, but they have built-in libraries that optimize the background, resulting in better performance. Functional programming varies in performance depending on the dataset given. Object-oriented programming is the best regarding performance because it can handle any dataset due to the vast libraries that address all performance issues.