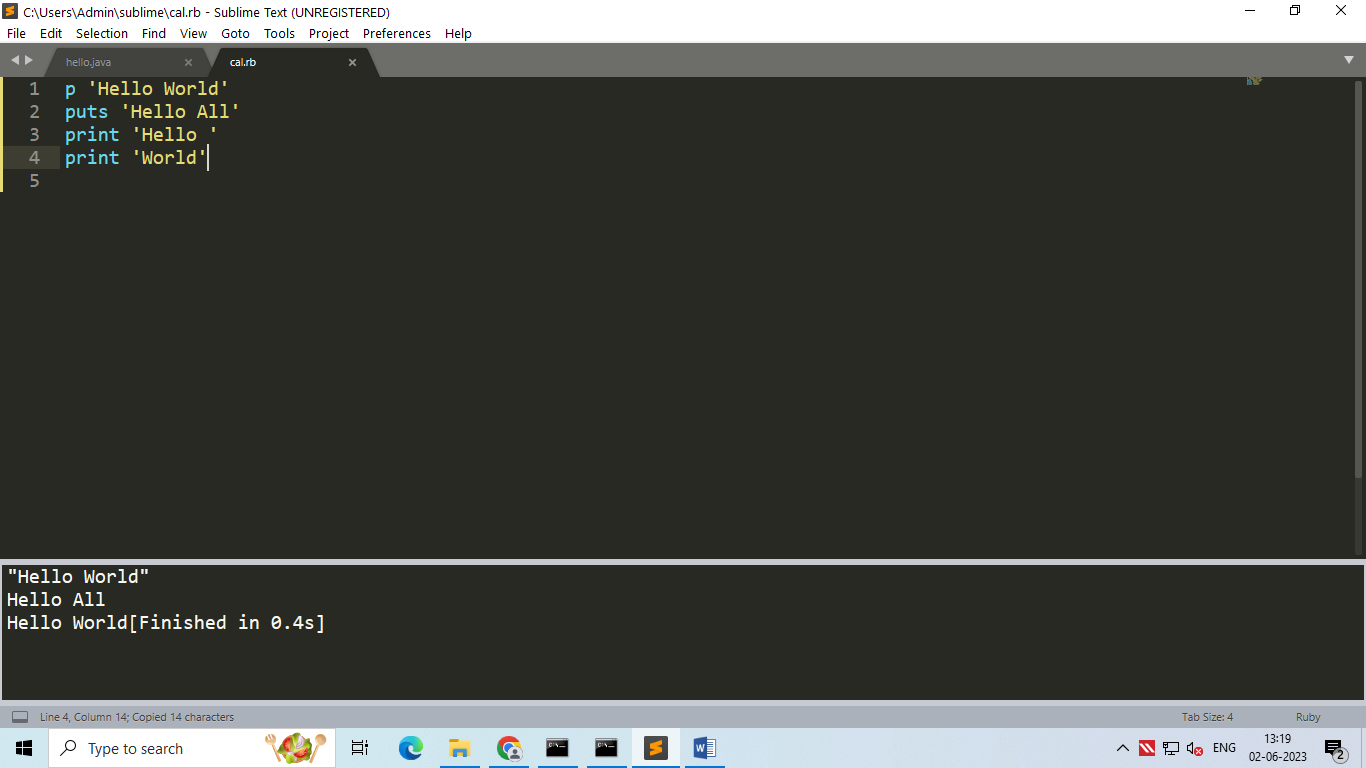
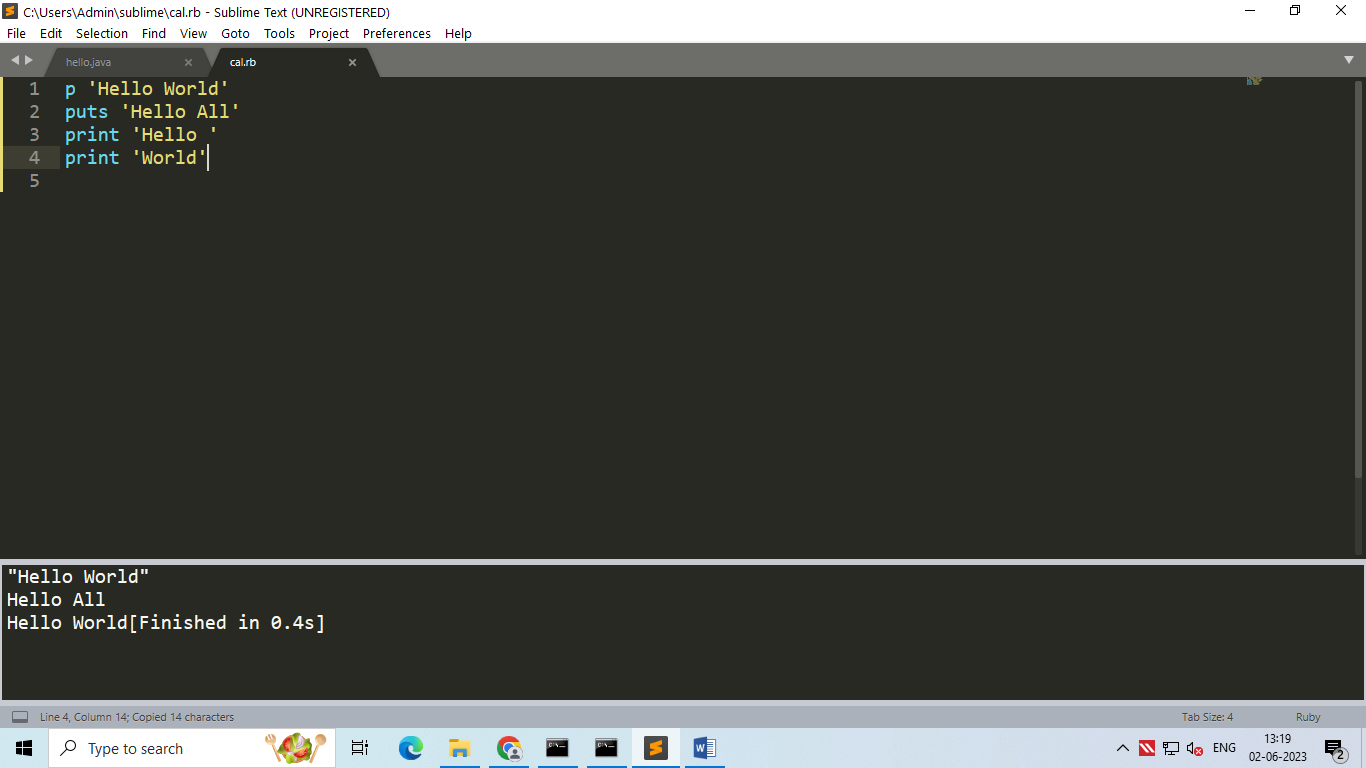
RUBY

1. Difference between p, puts, print :

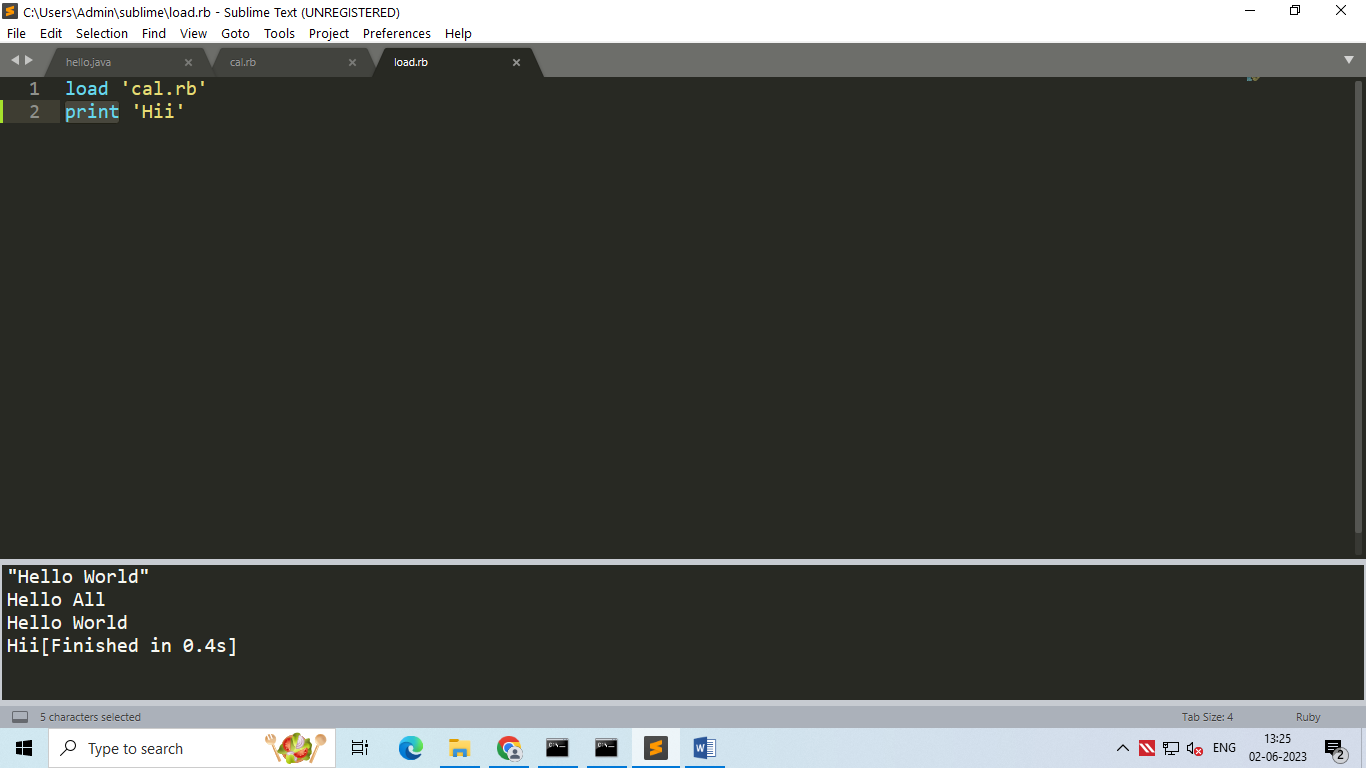


O/P:

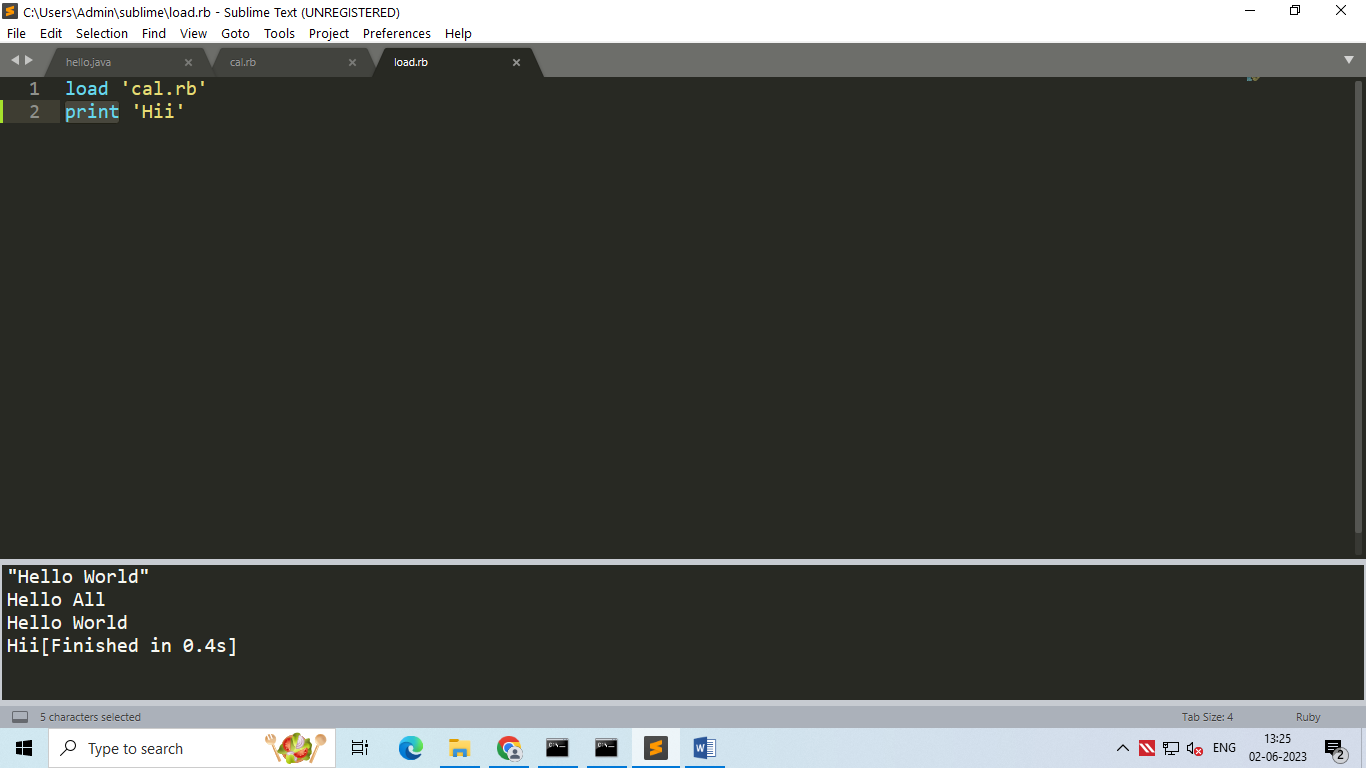


1. Load function:

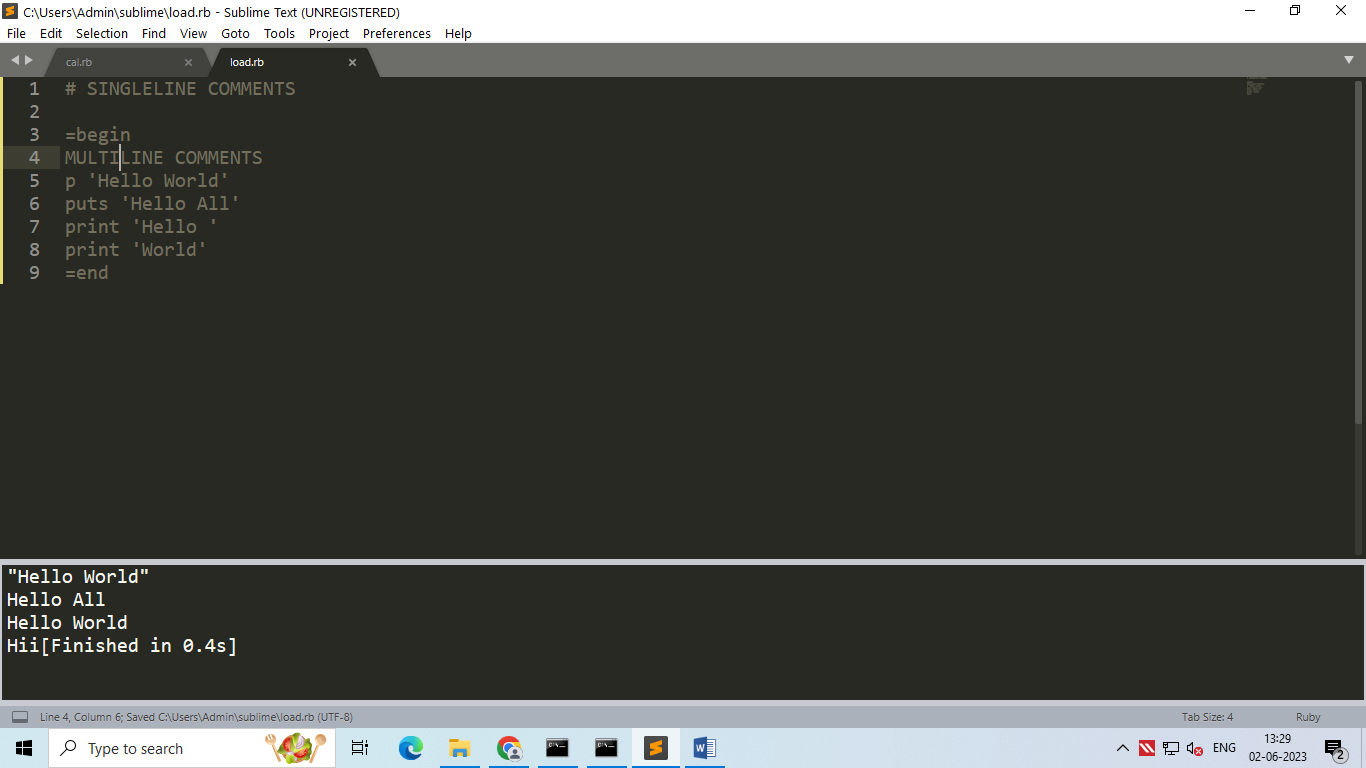
Load Function is used to call one file in another file. Both file should be saved in same location.



O/P:



1. Comments :



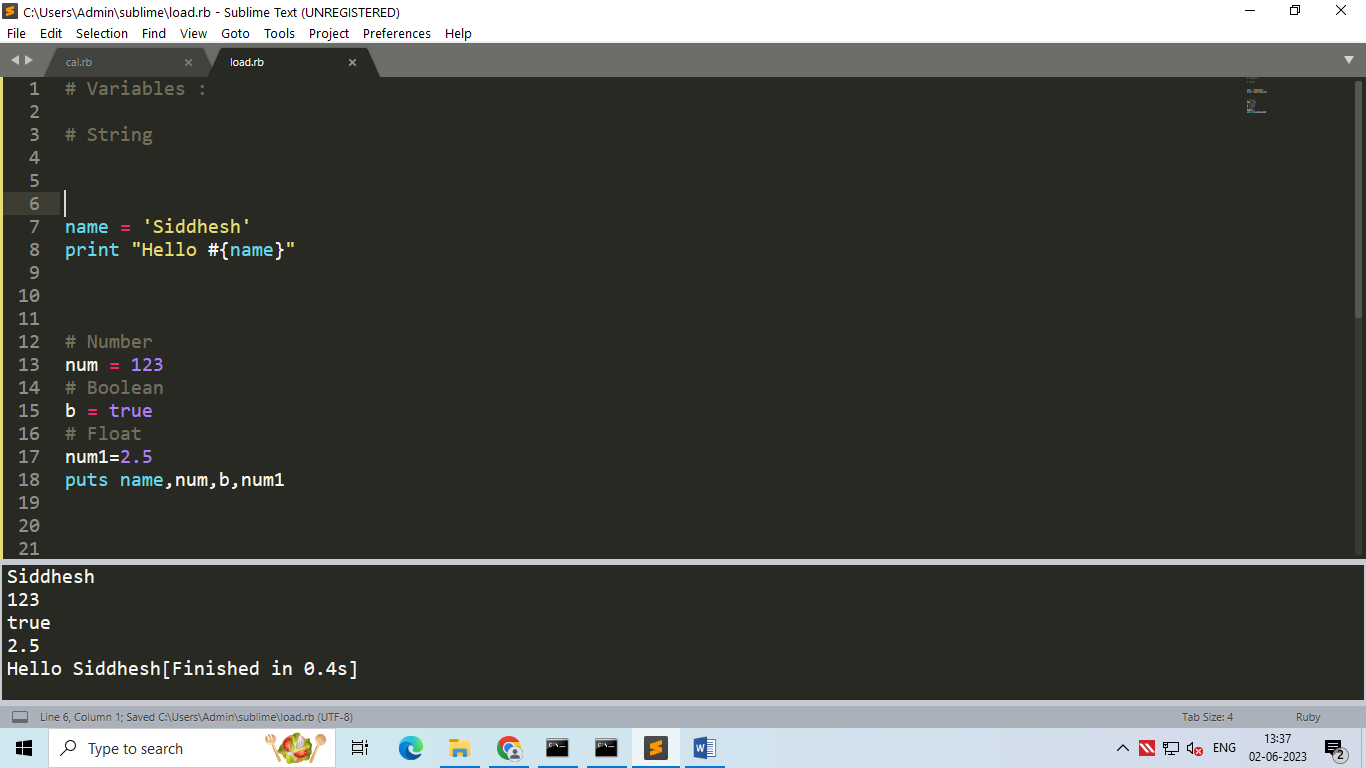
1. Variables:



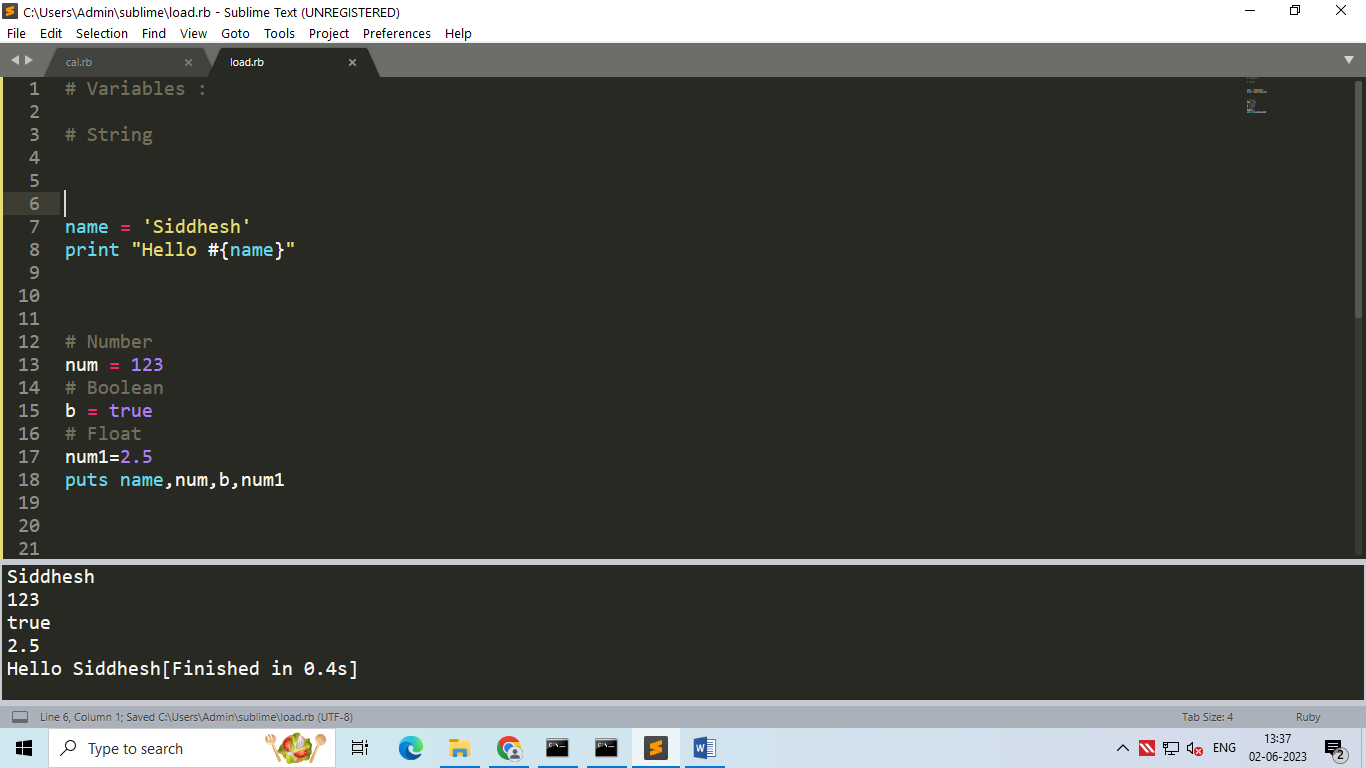
O/P:



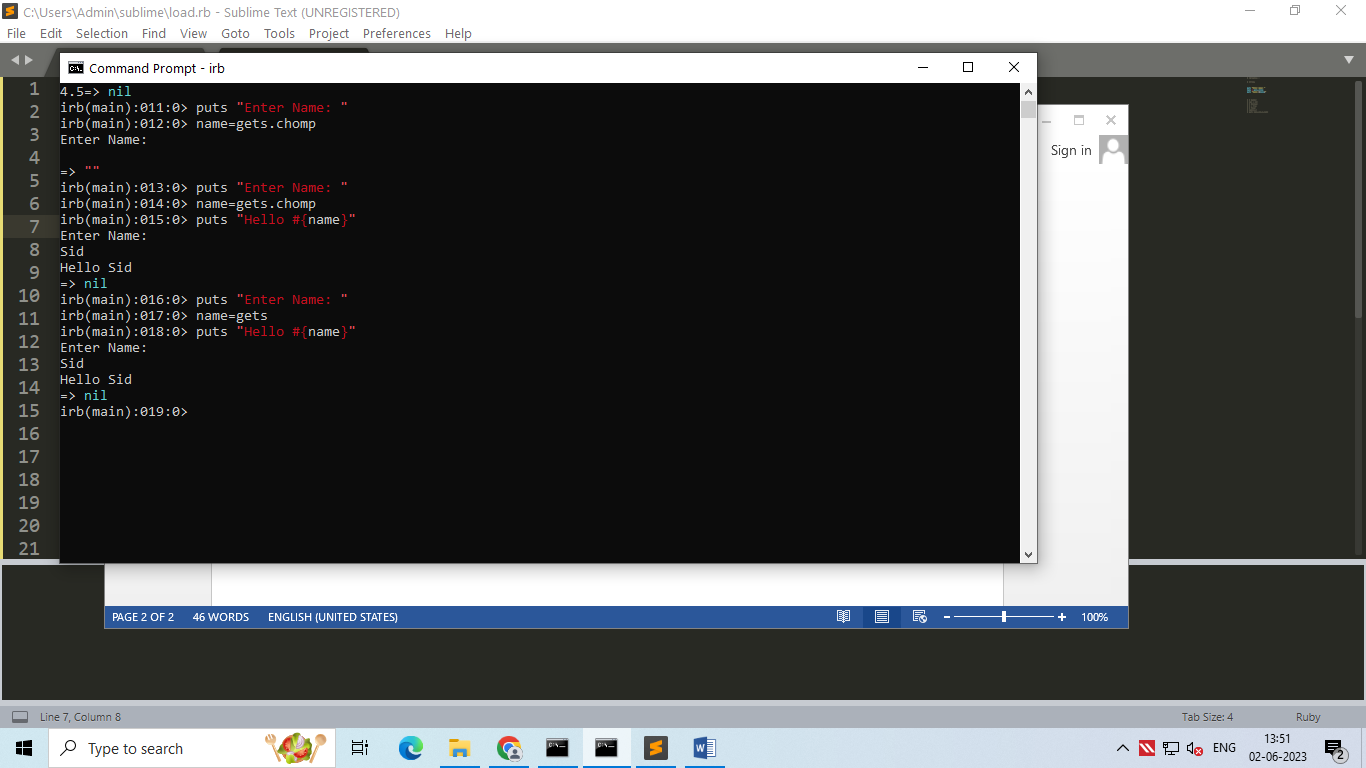
1. #{} :



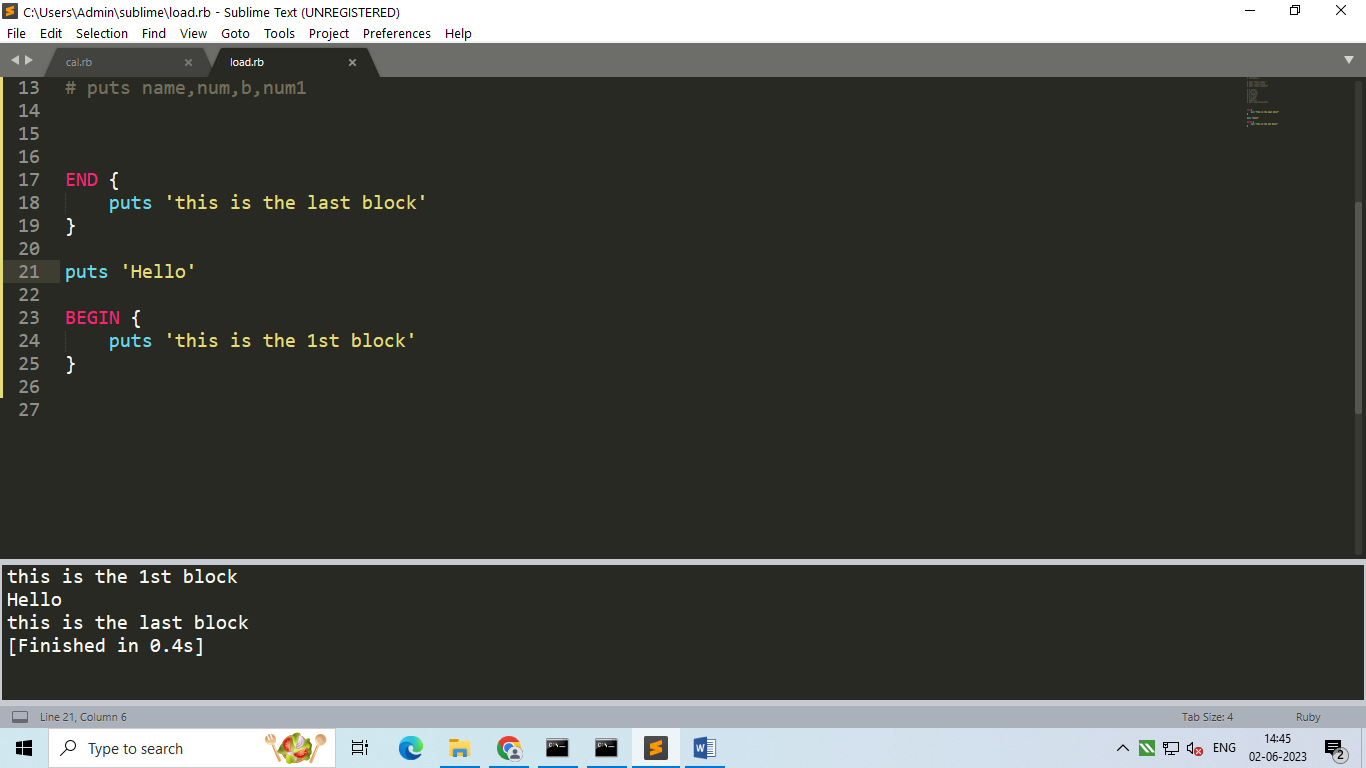
O/P:



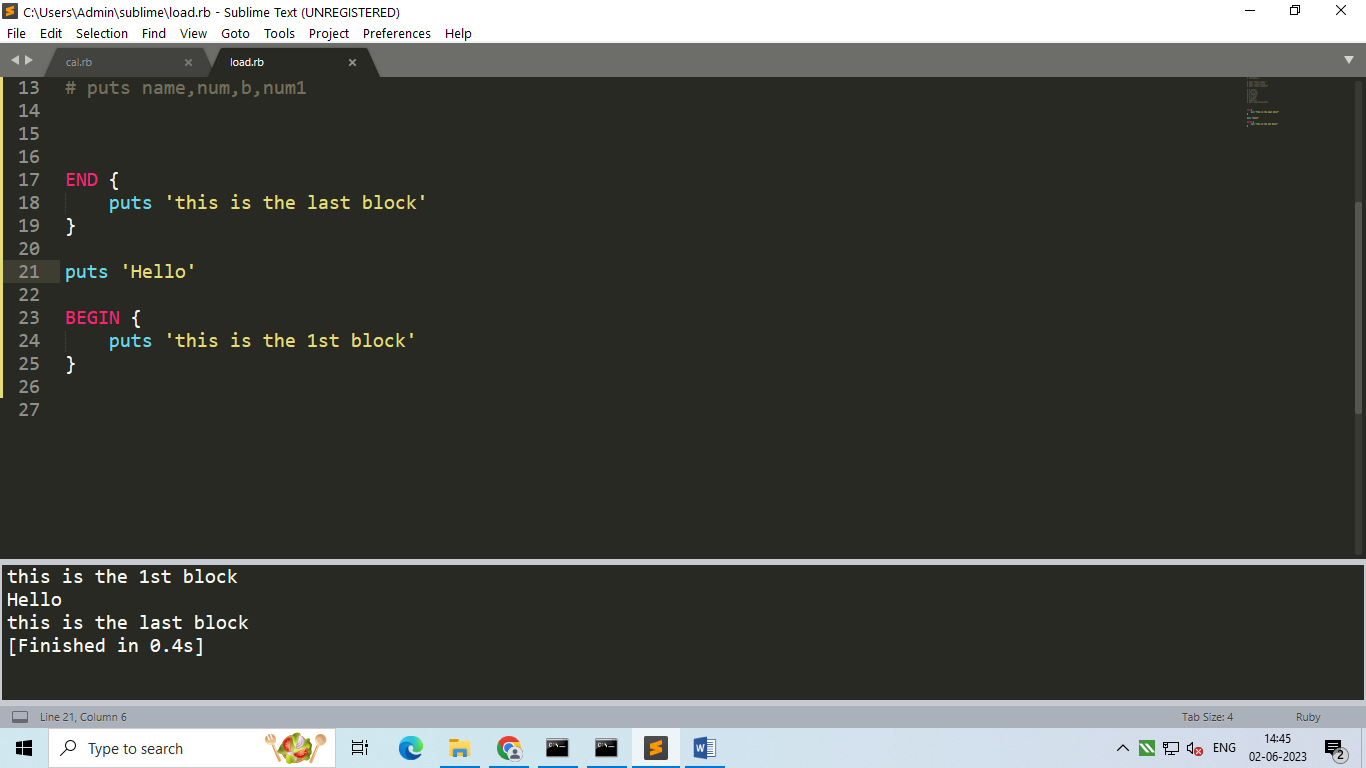
1. User Input :



1. Begin & End Block :



O/P :

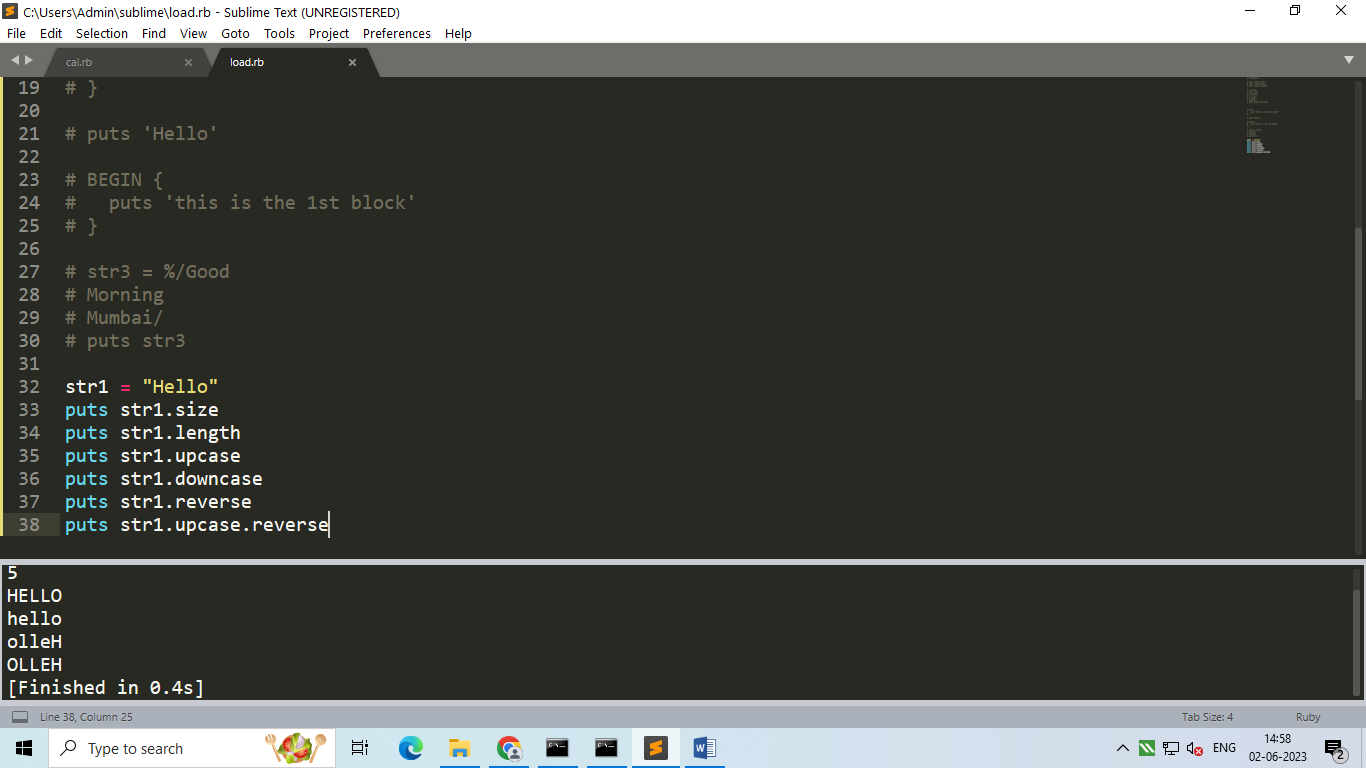


1. String Operations :

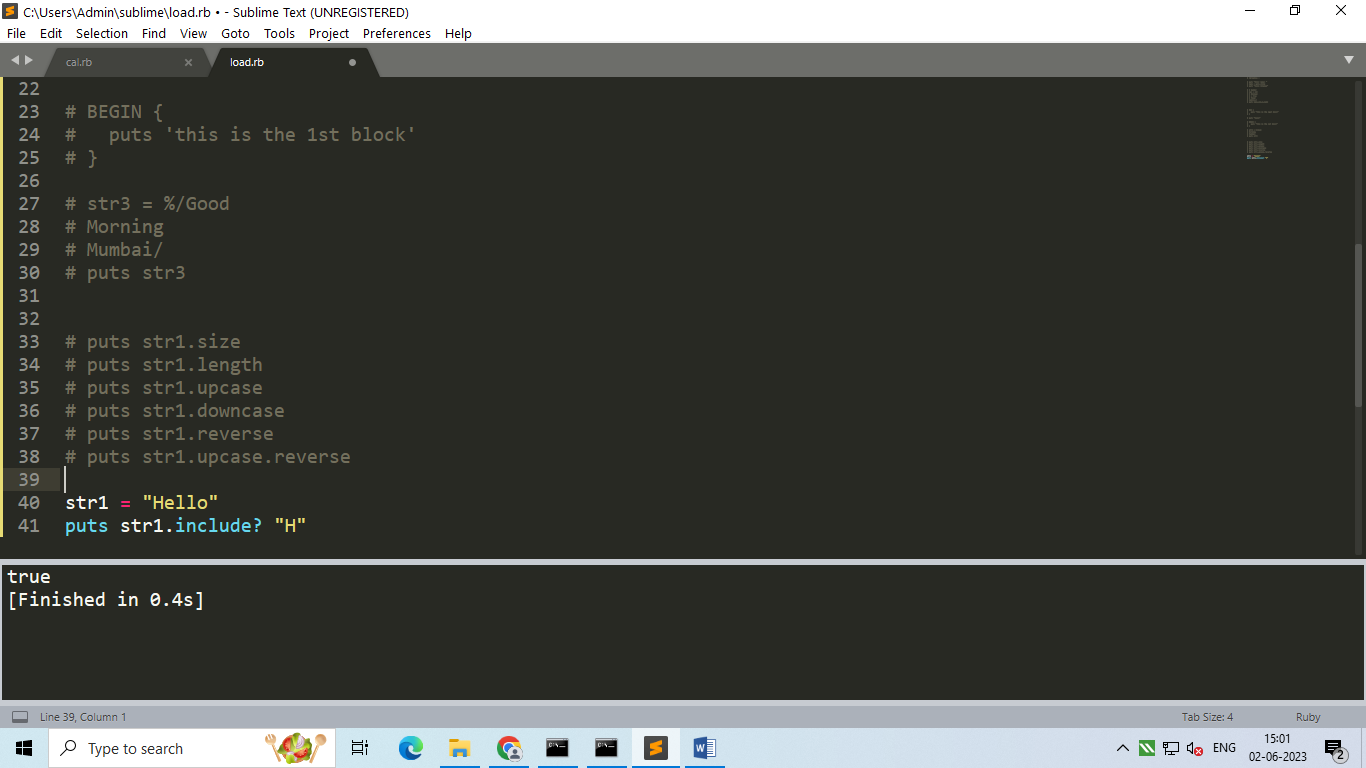
* Multiline :

|  |  |
| --- | --- |
|  |  |

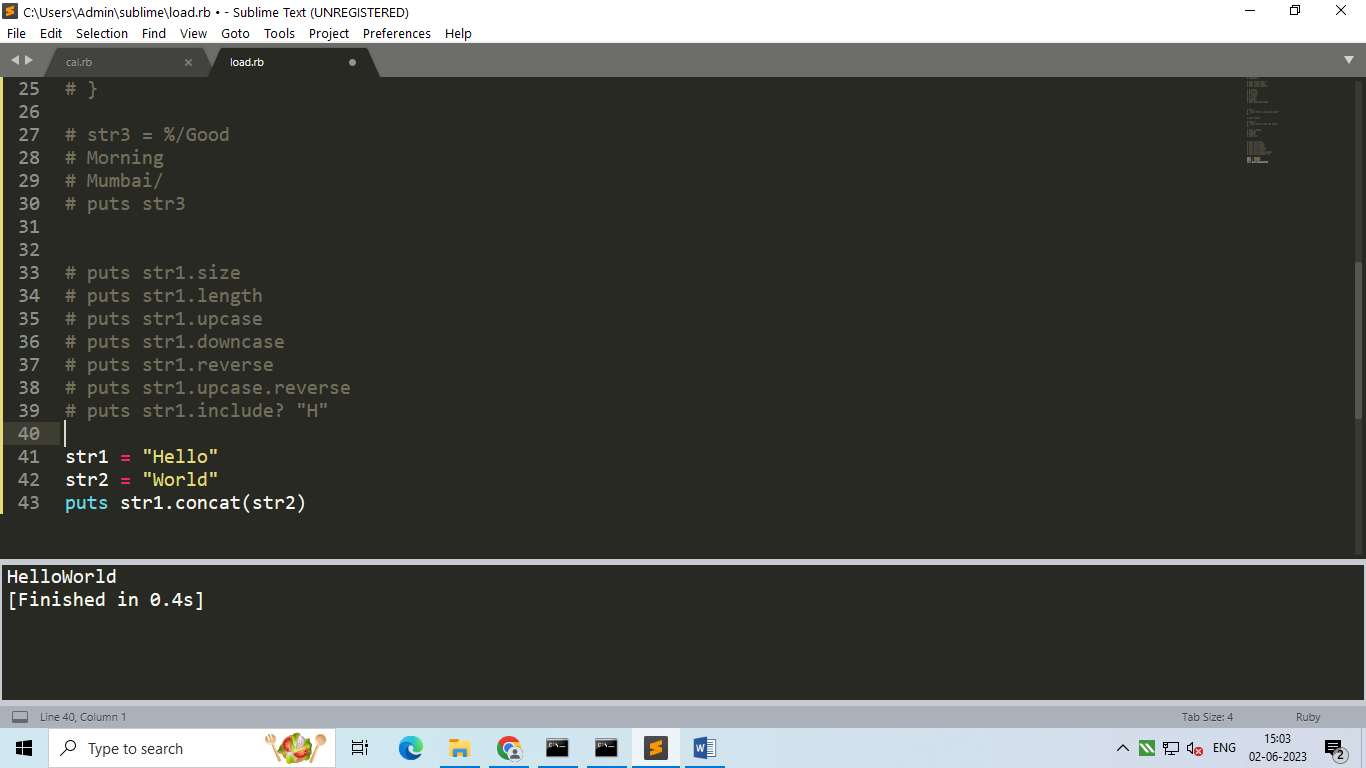
* Size , case, reverse :



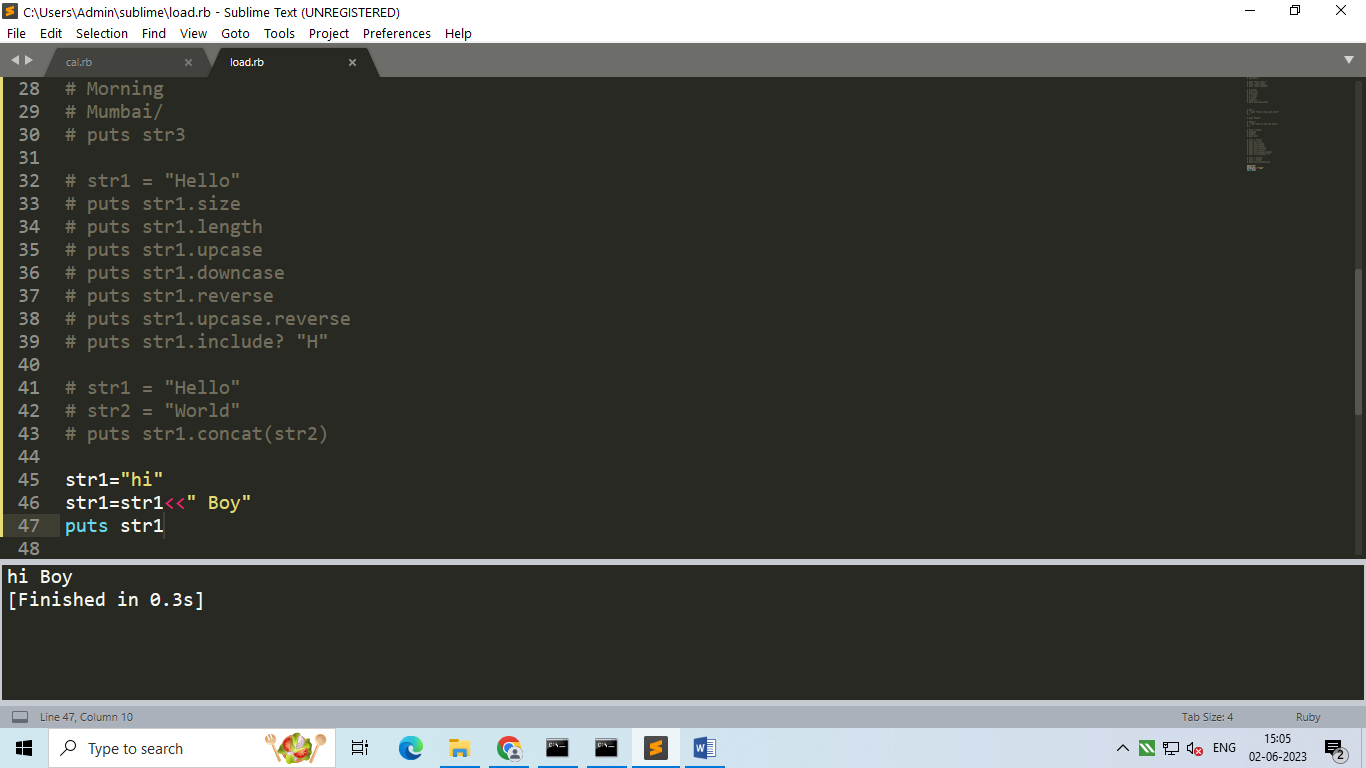
* Include :



* Concat :

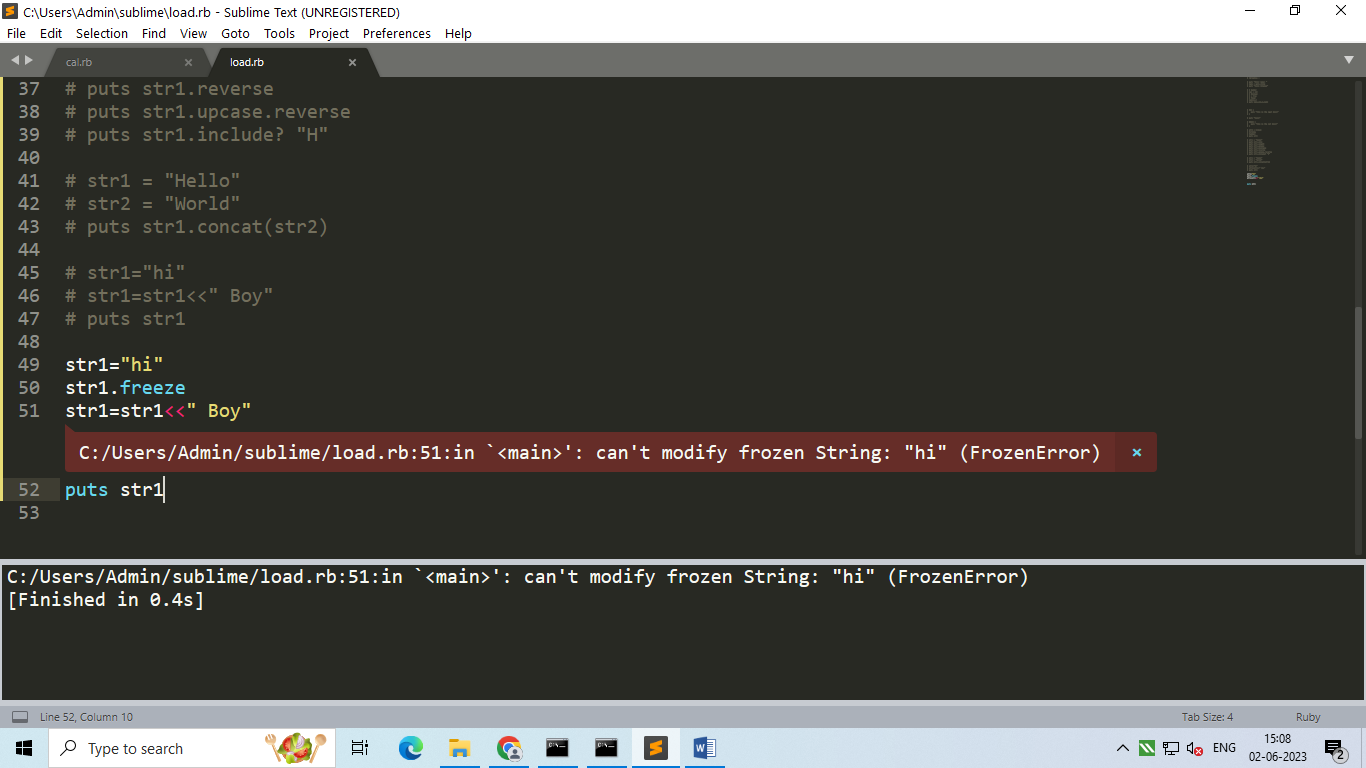


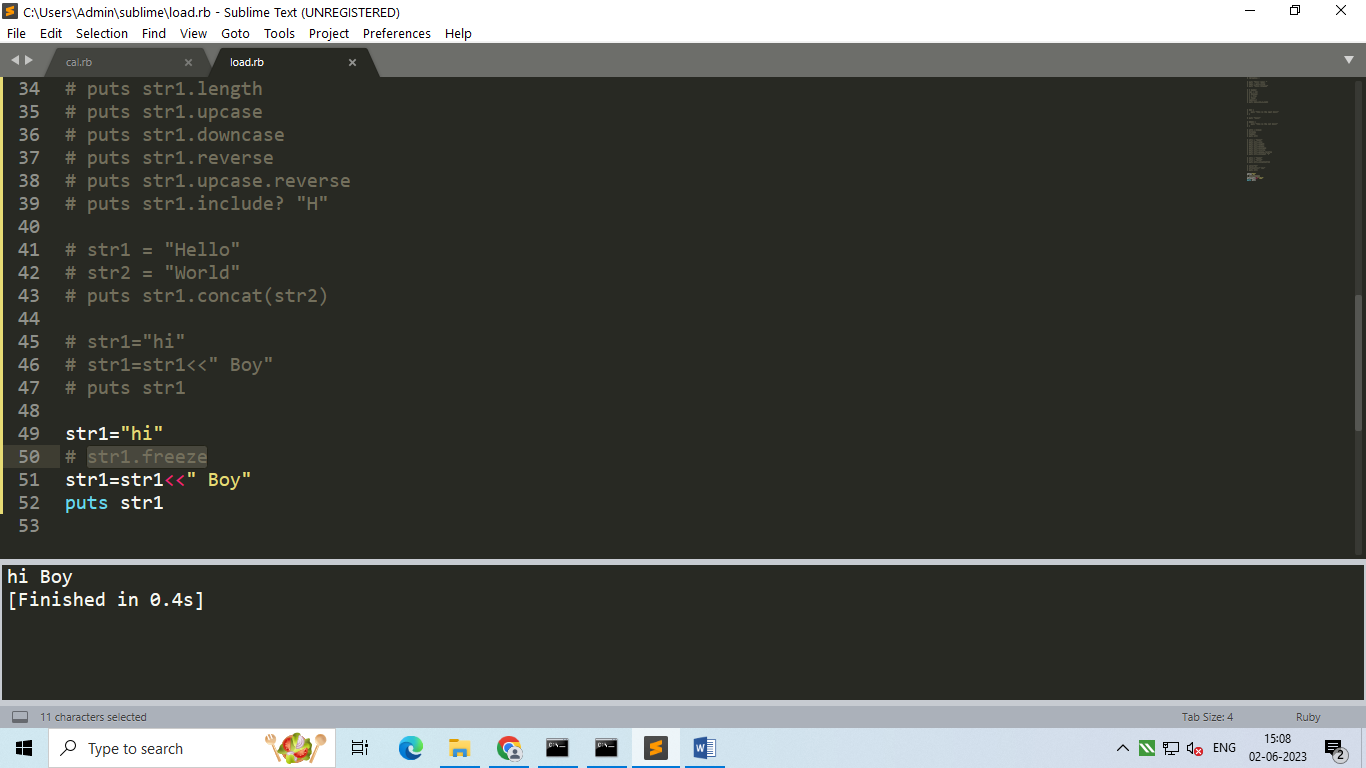
* Append :



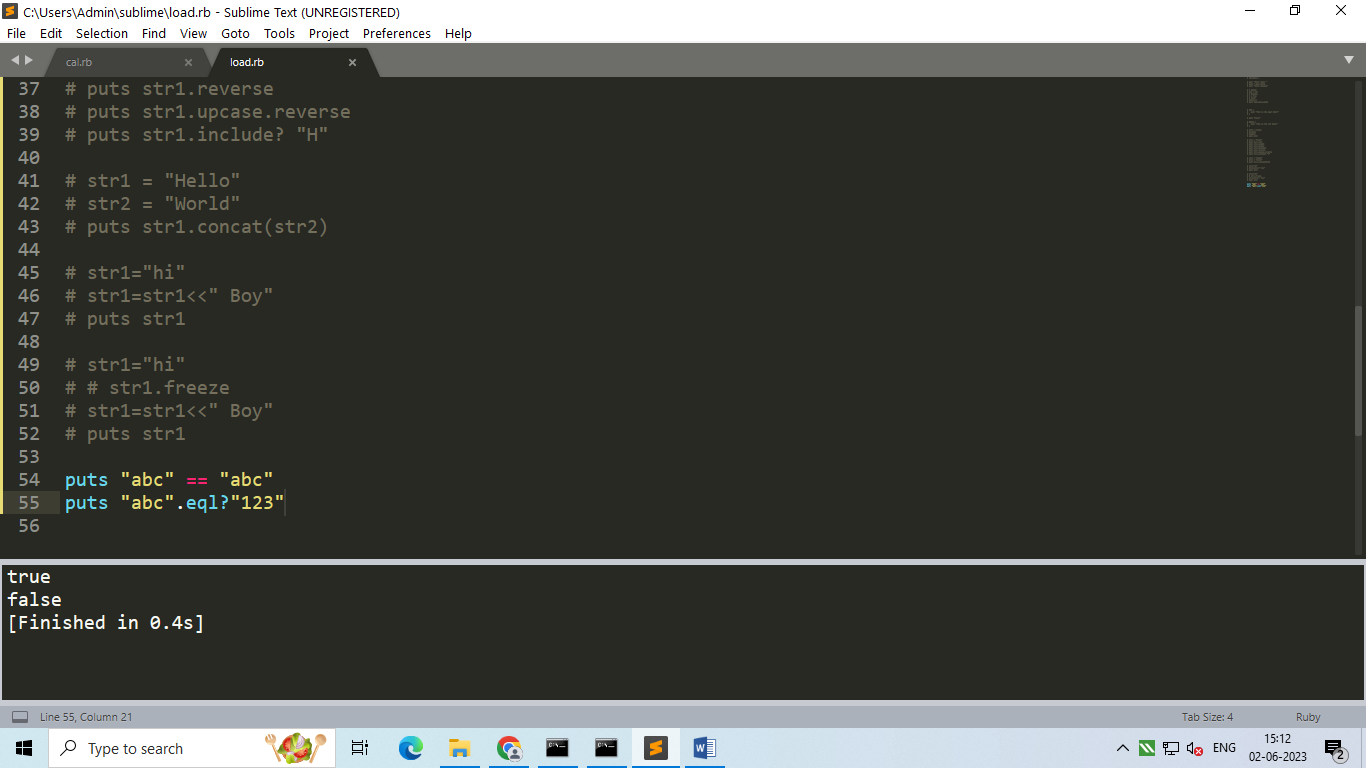
* Freeze :

After freezing a String you cant append it.

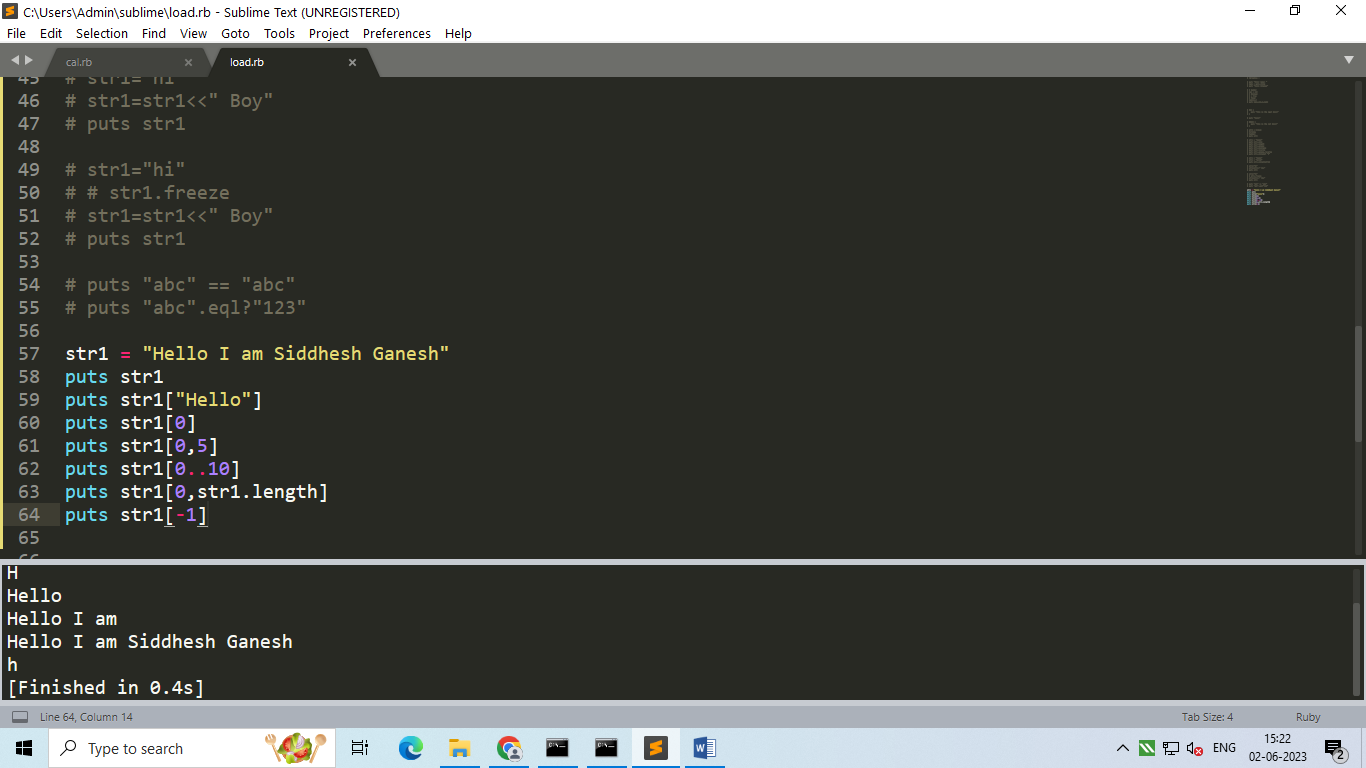




* Comparison :

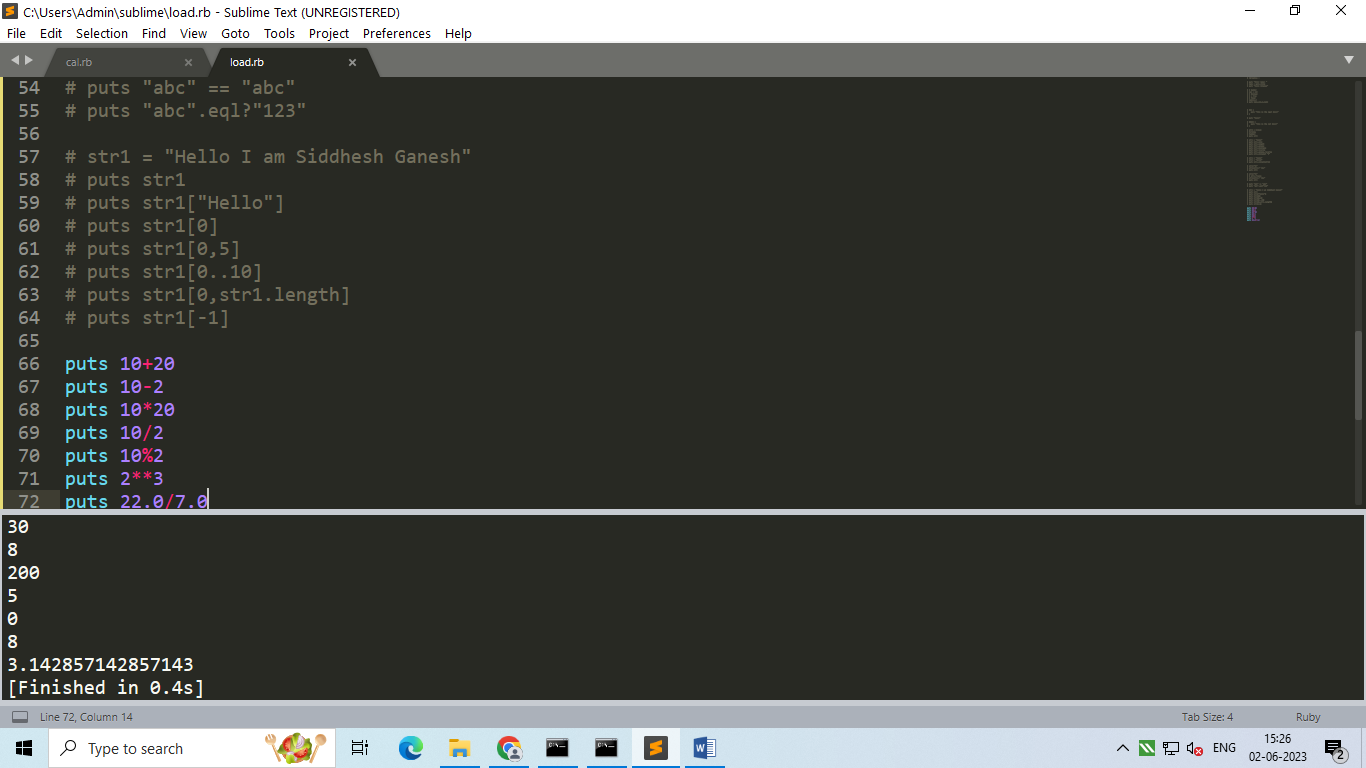


* Slicing :

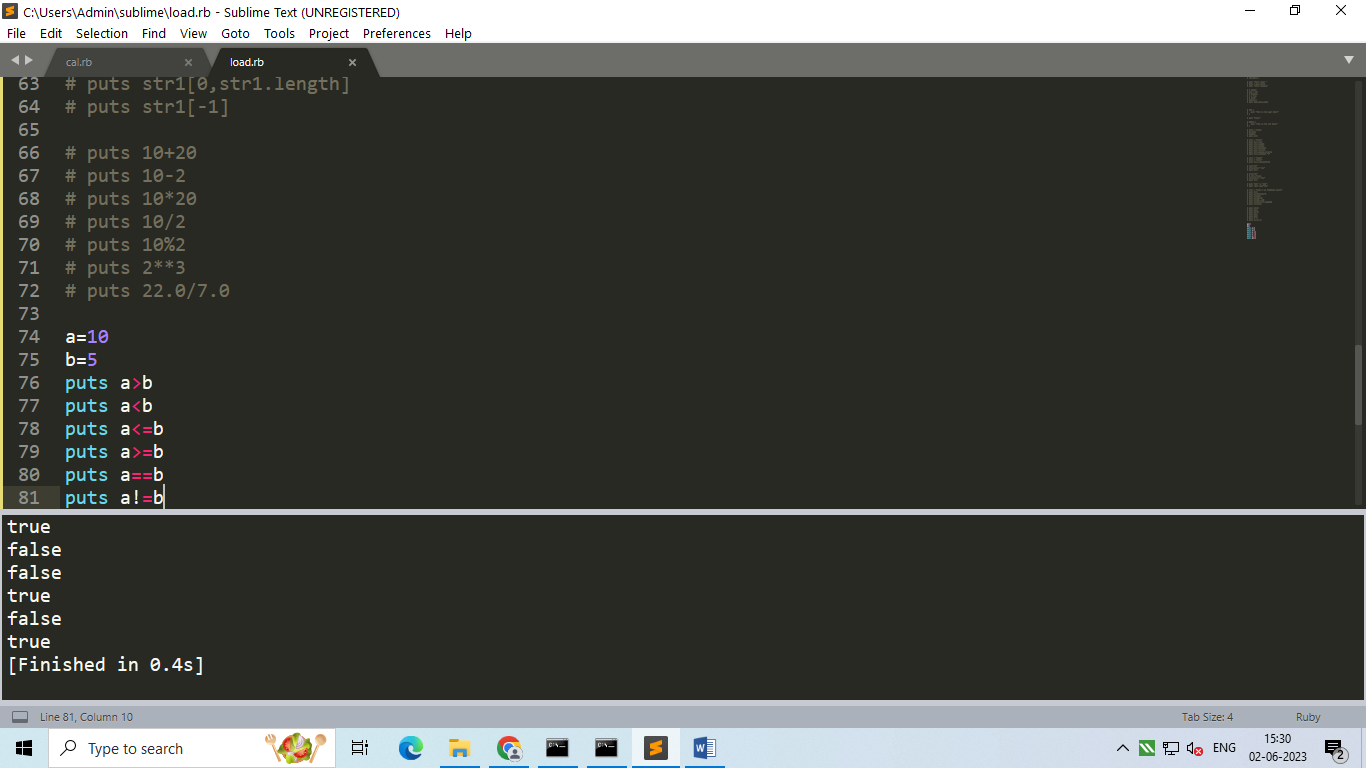


1. Operators :

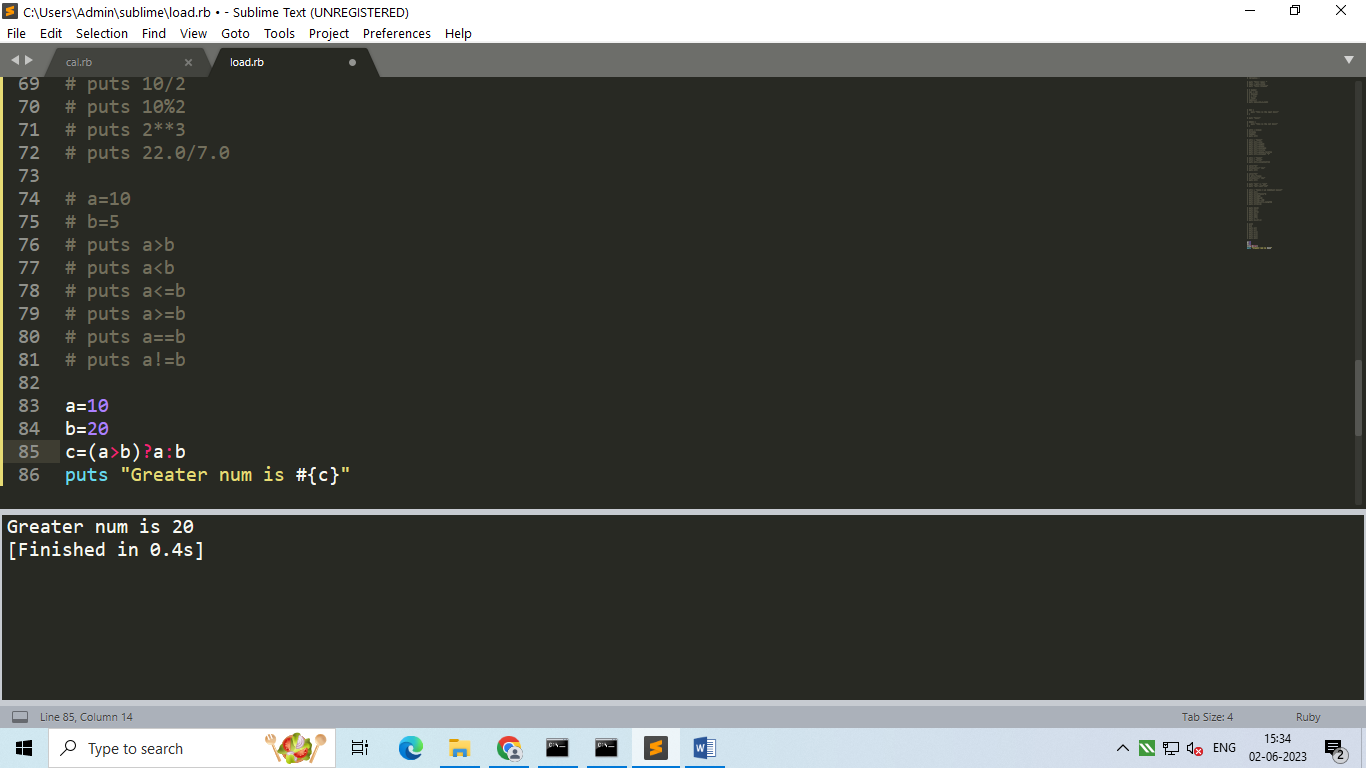
* Arithmetic :



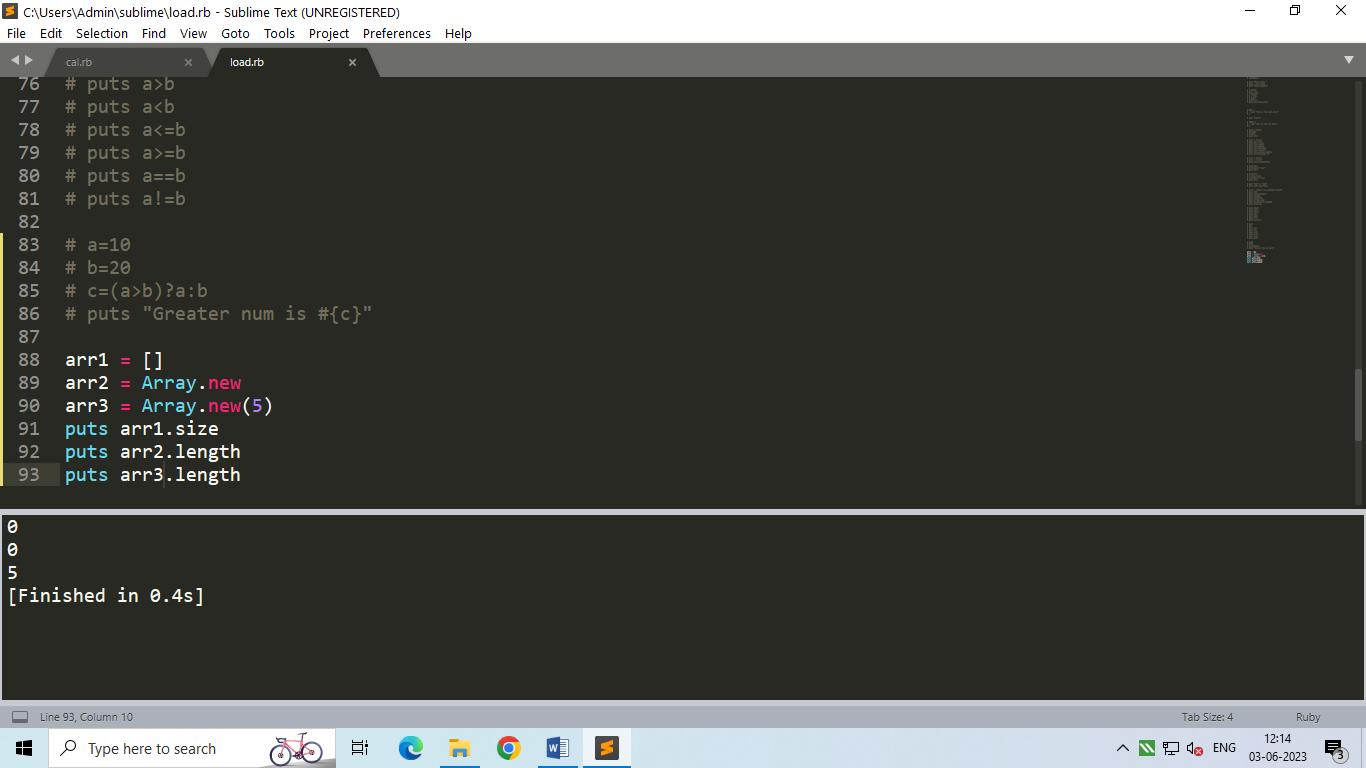
* Relational :



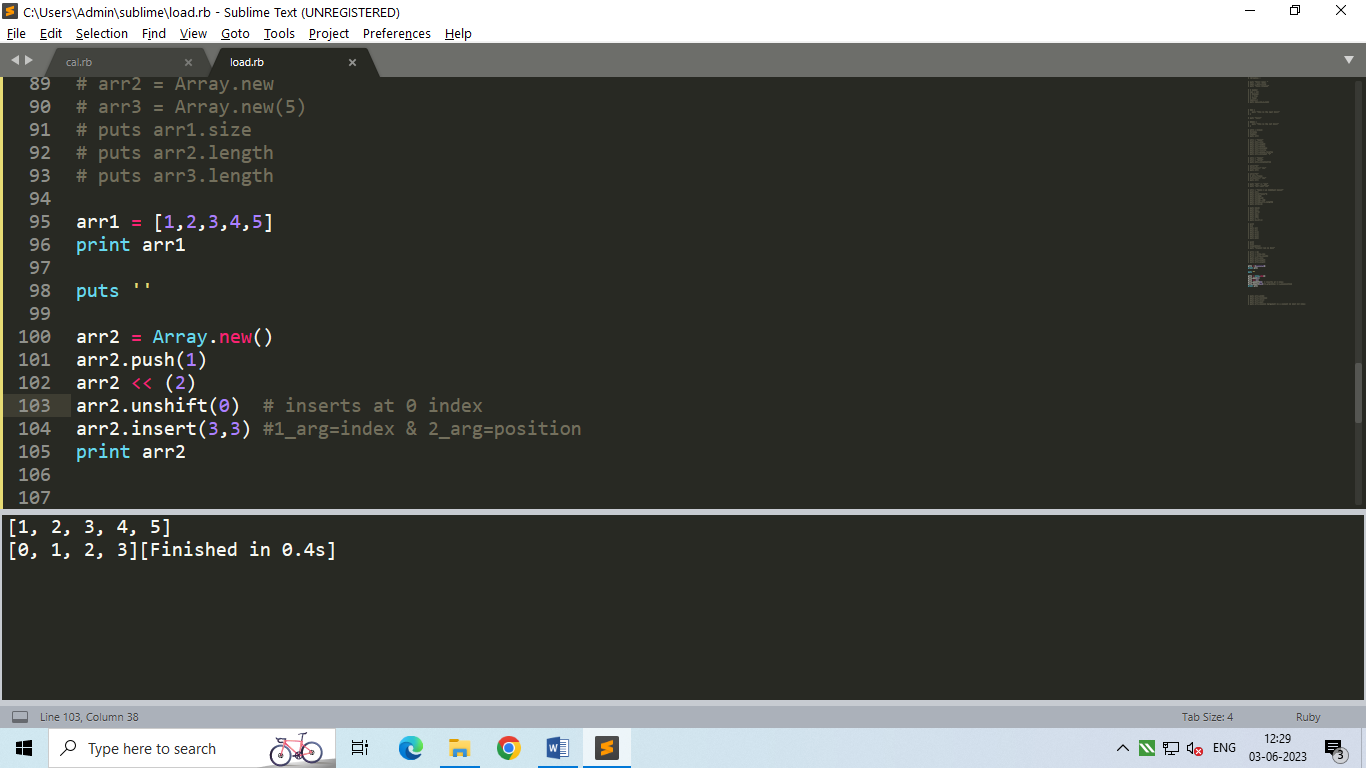
1. Ternary Operator :



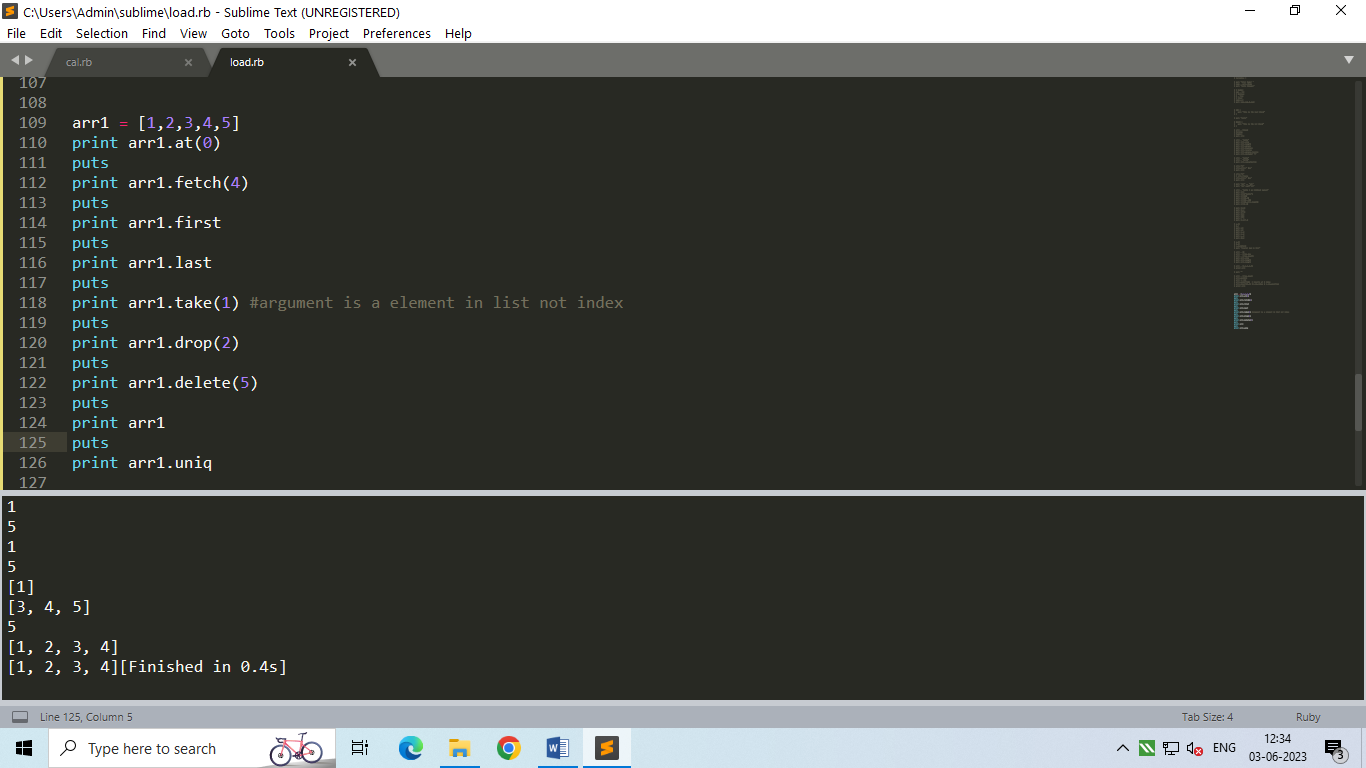
1. Array Declaration :



1. Inserting elements in Array :



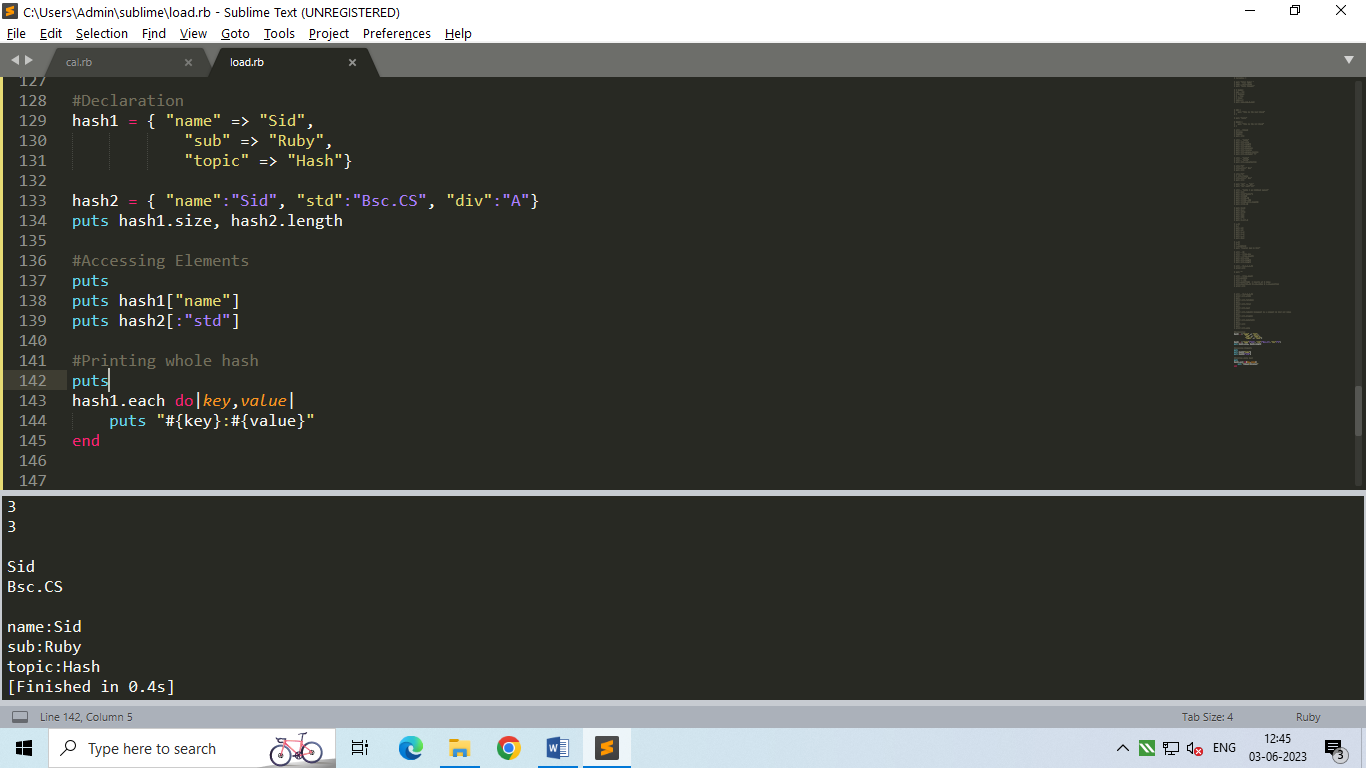
1. Accessing Elements from array :



1. Hashes :

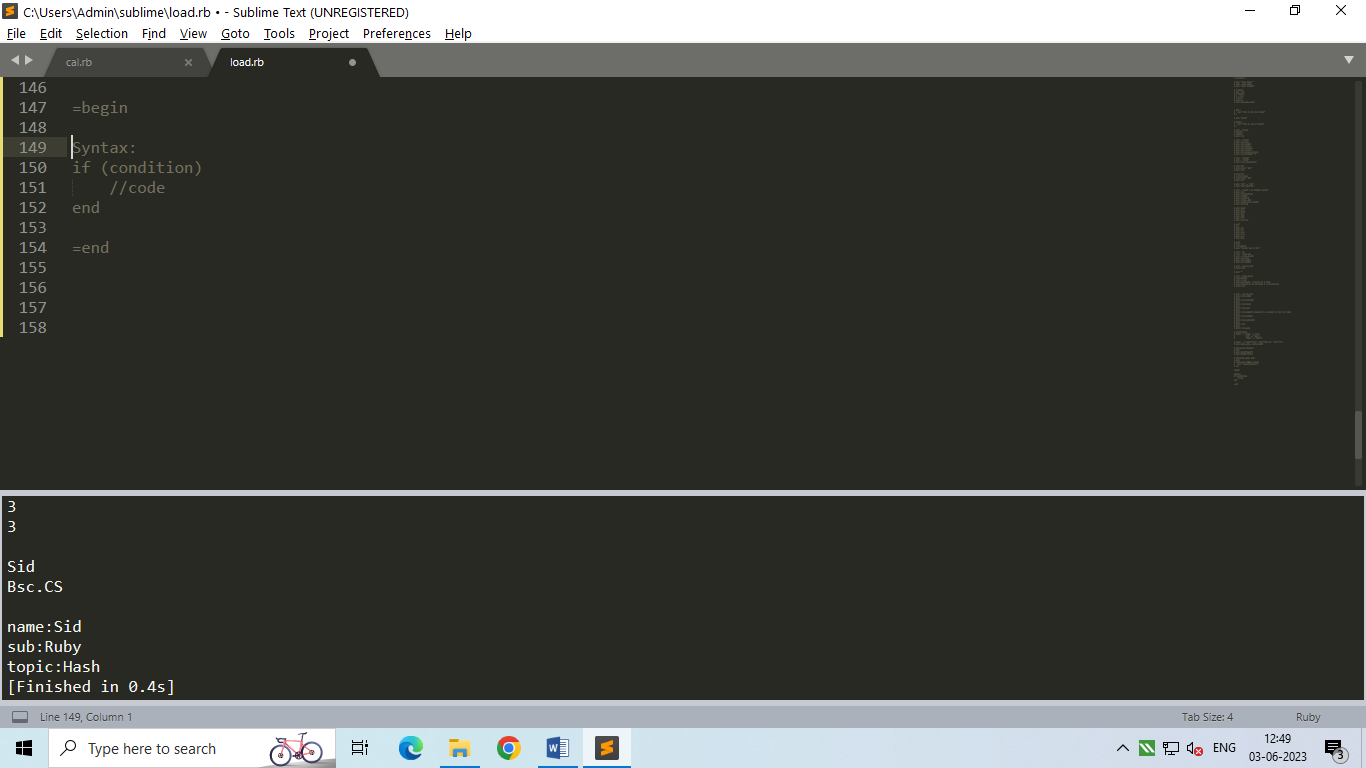
Array stores value with a numeric index ,

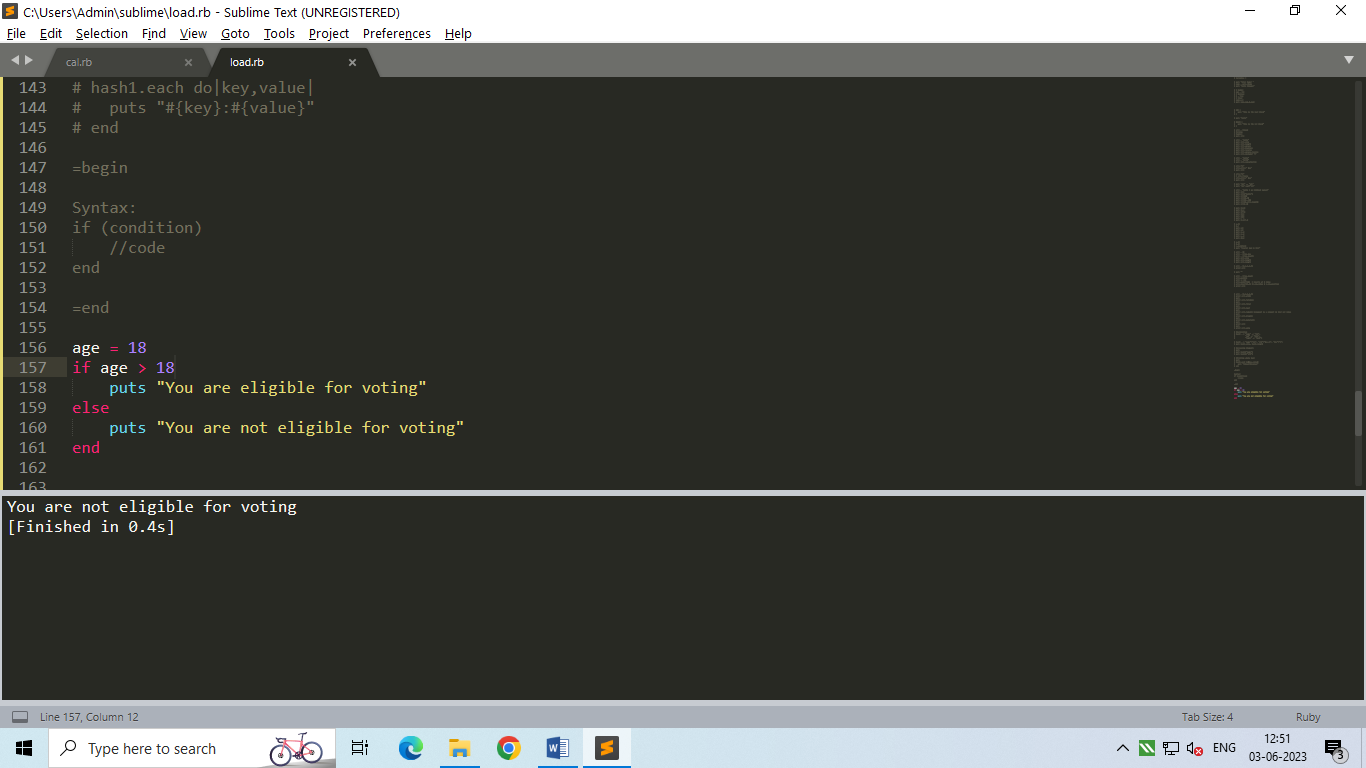
Hashes store info using key value pairs.

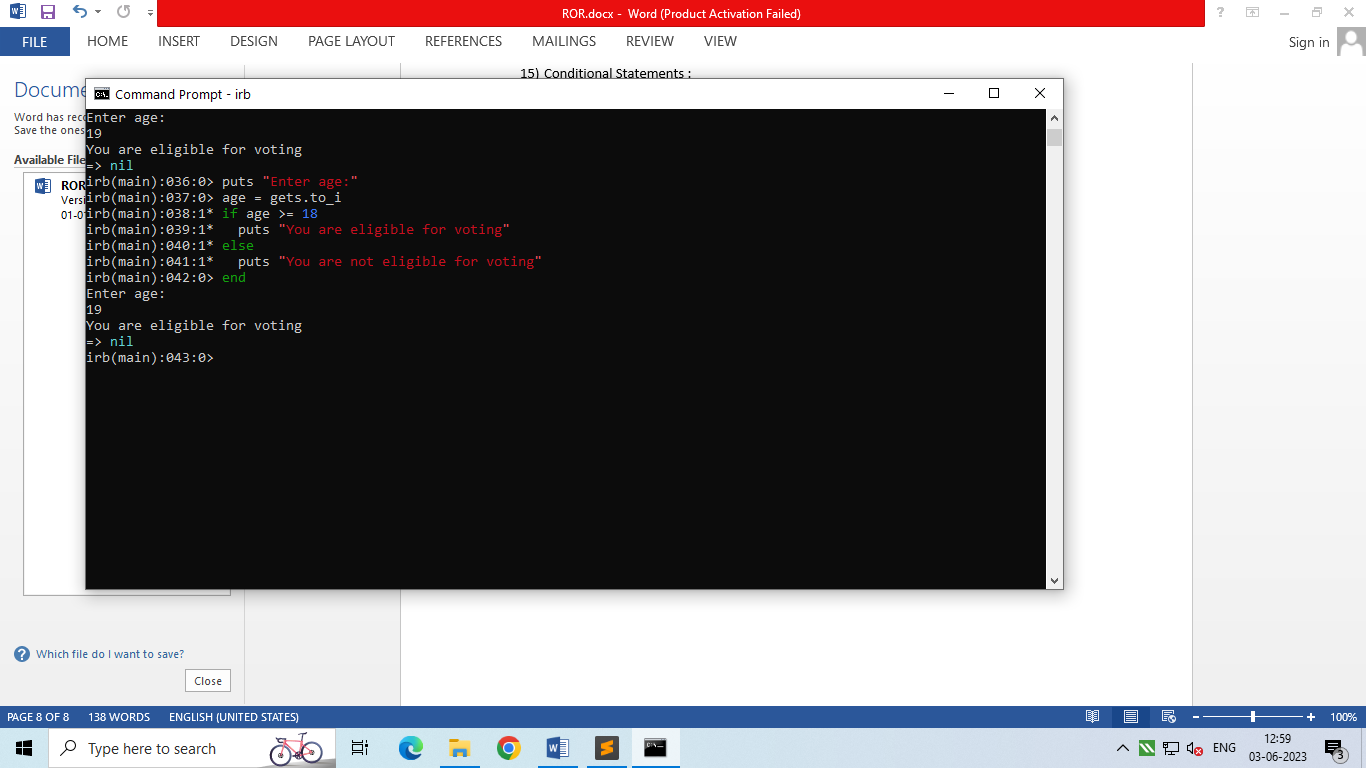


1. Conditional Statements :

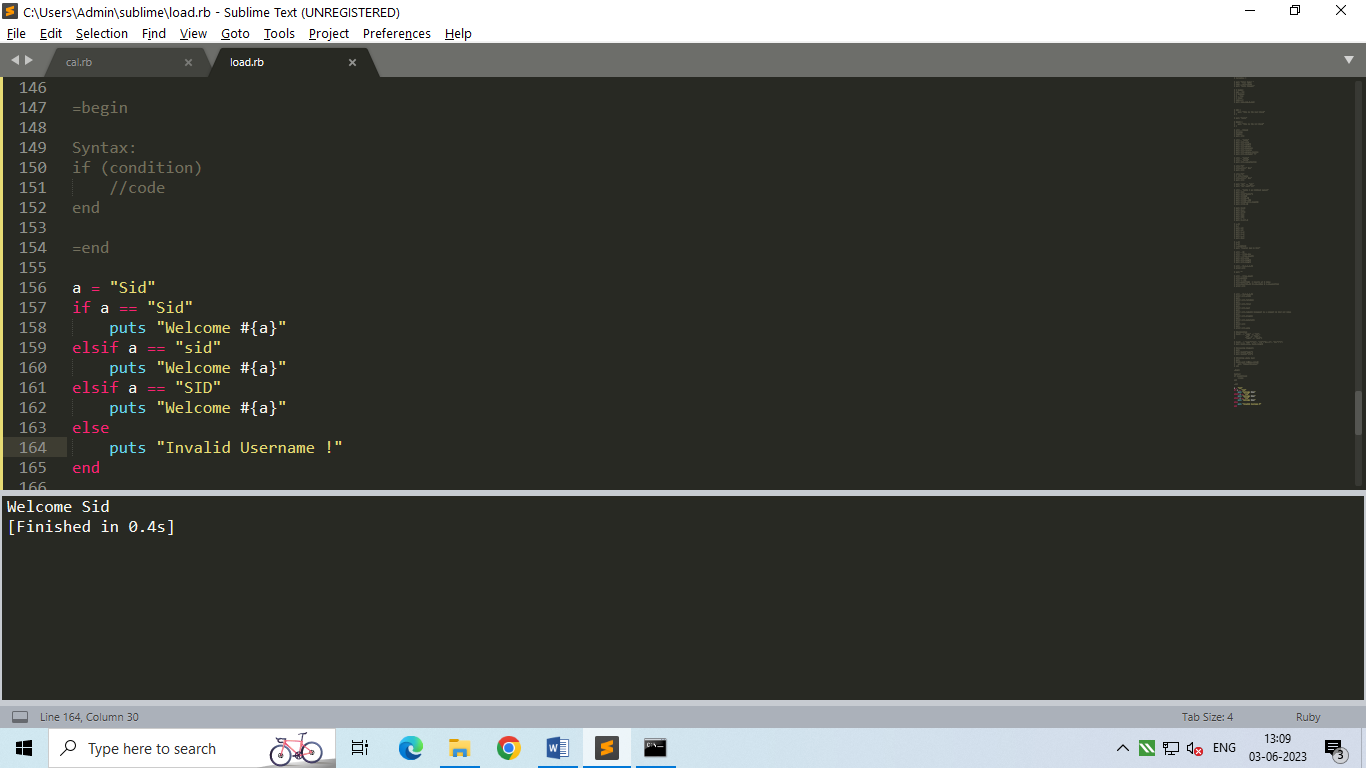
* If …. Else :



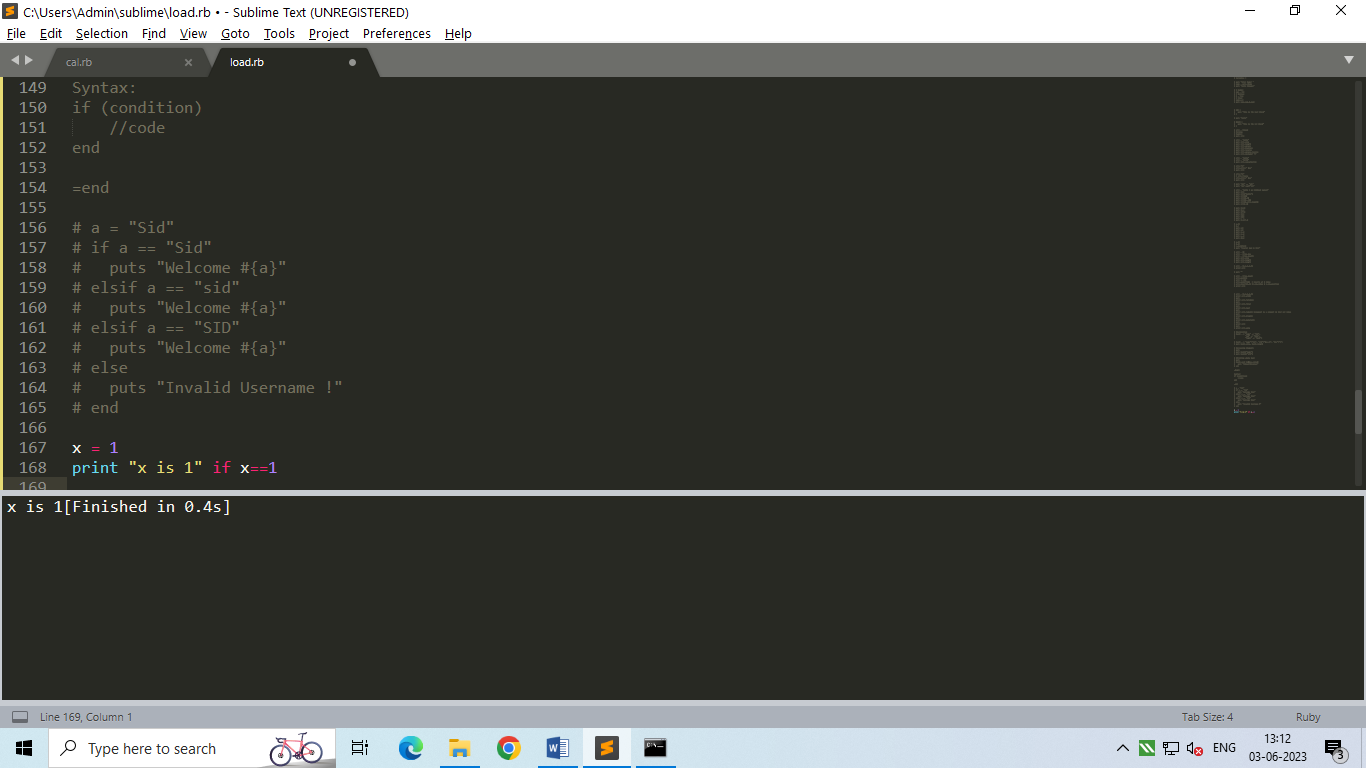


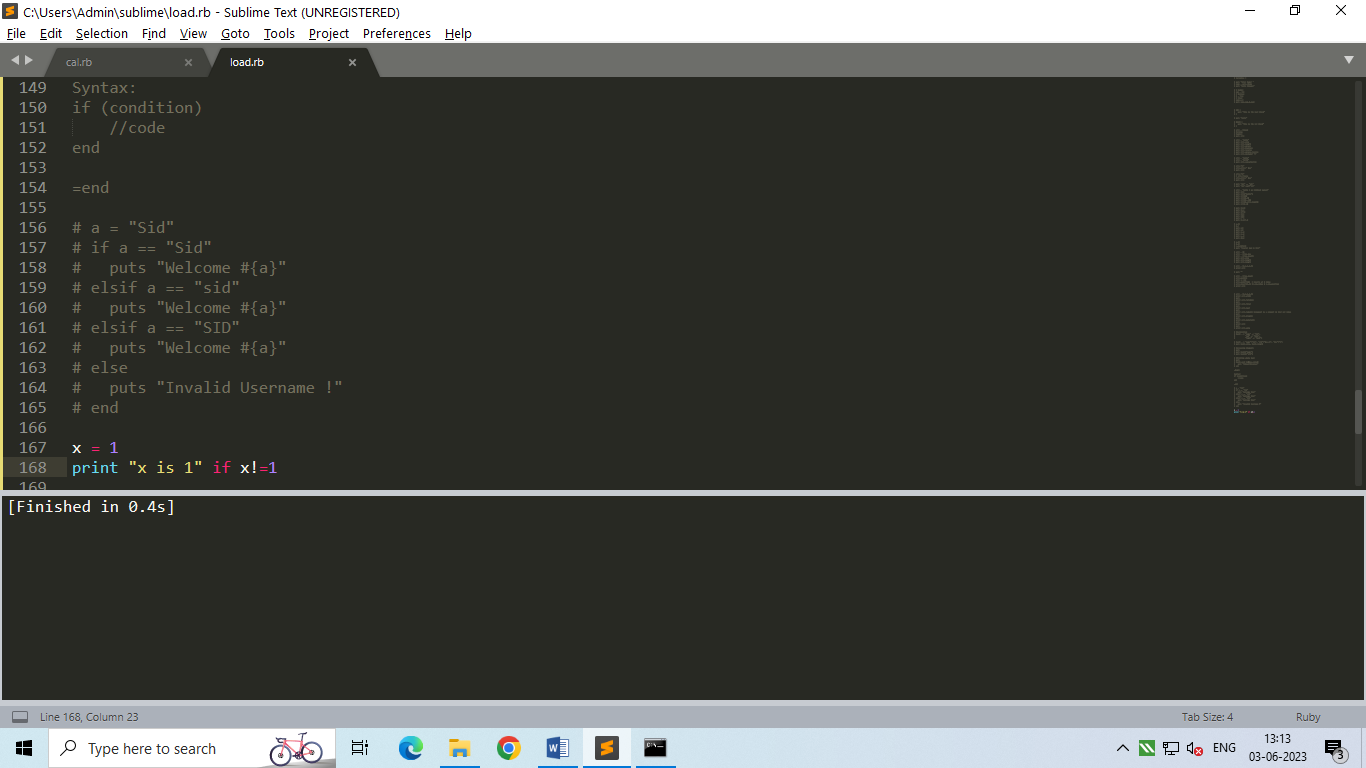


* If….elsif….else :



* If modifier :

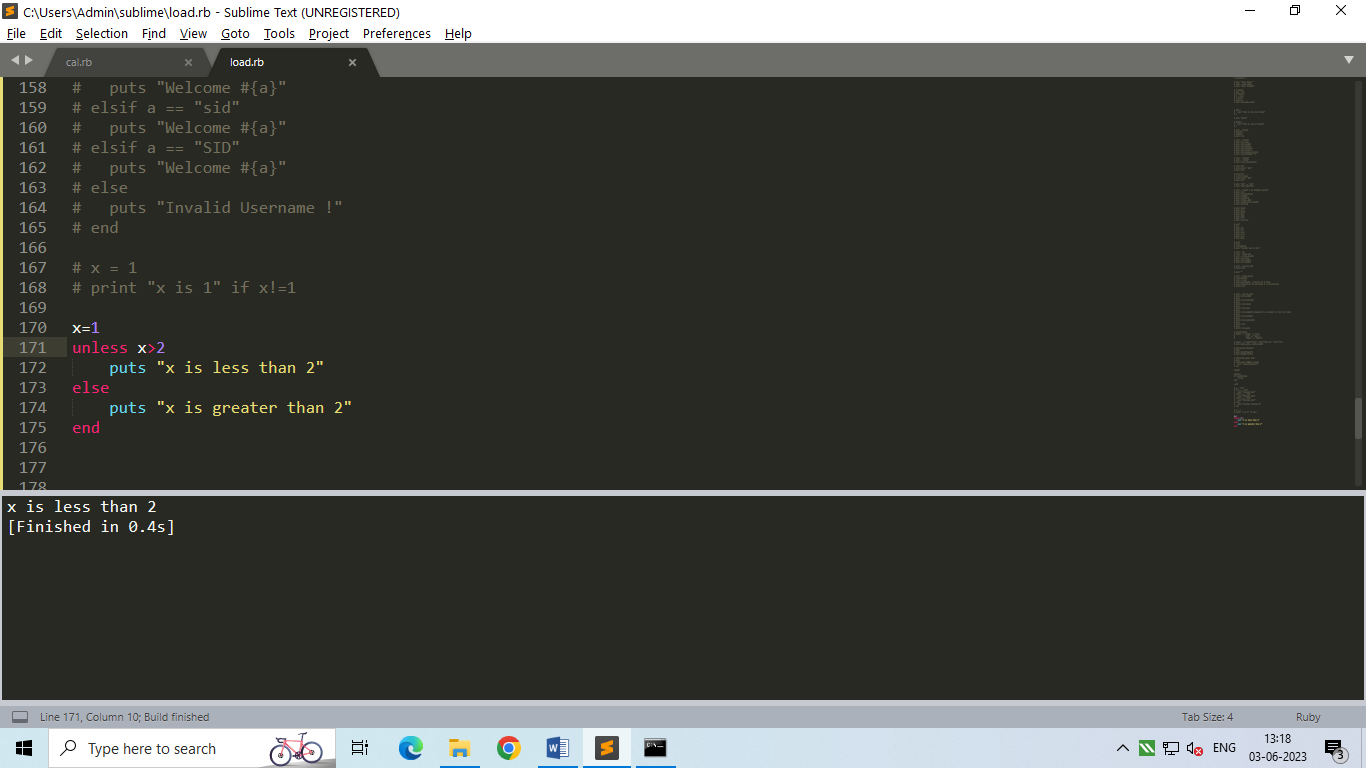


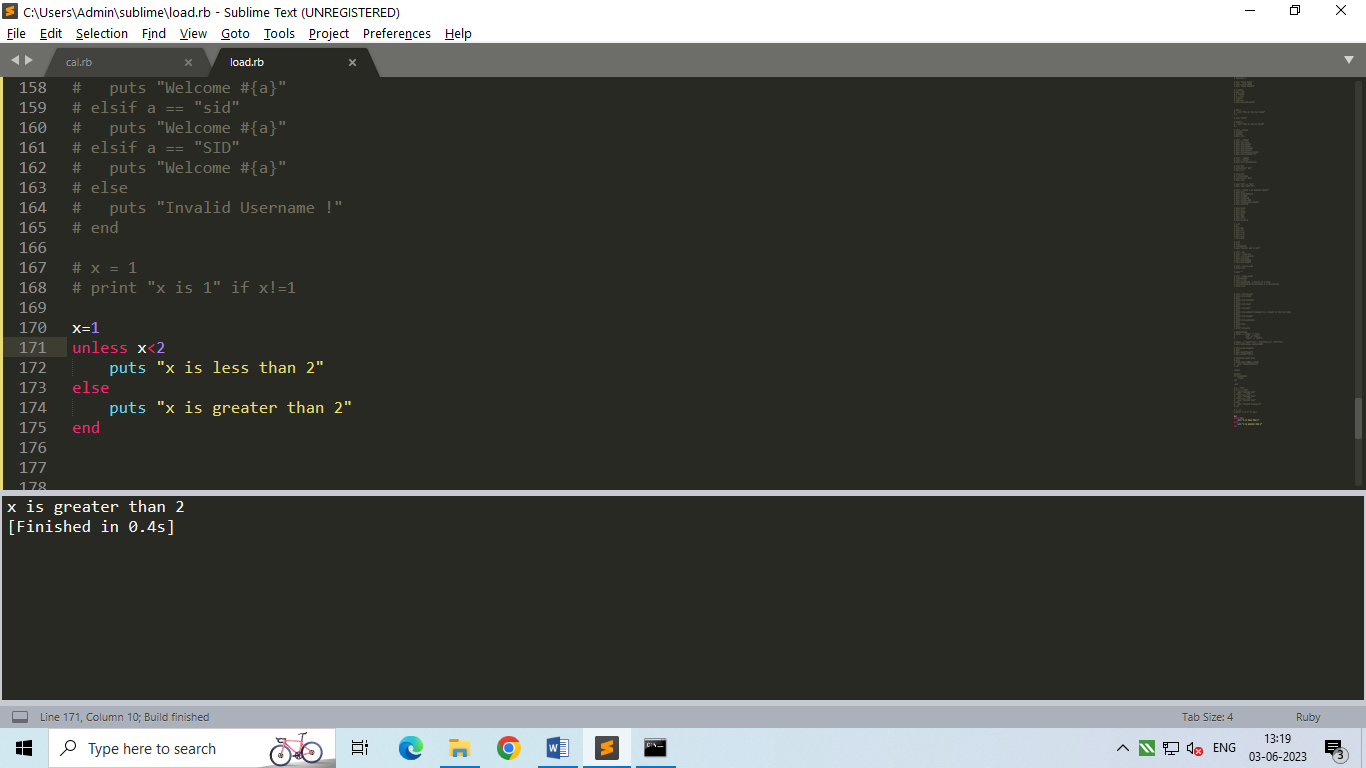


* Unless …. Else:

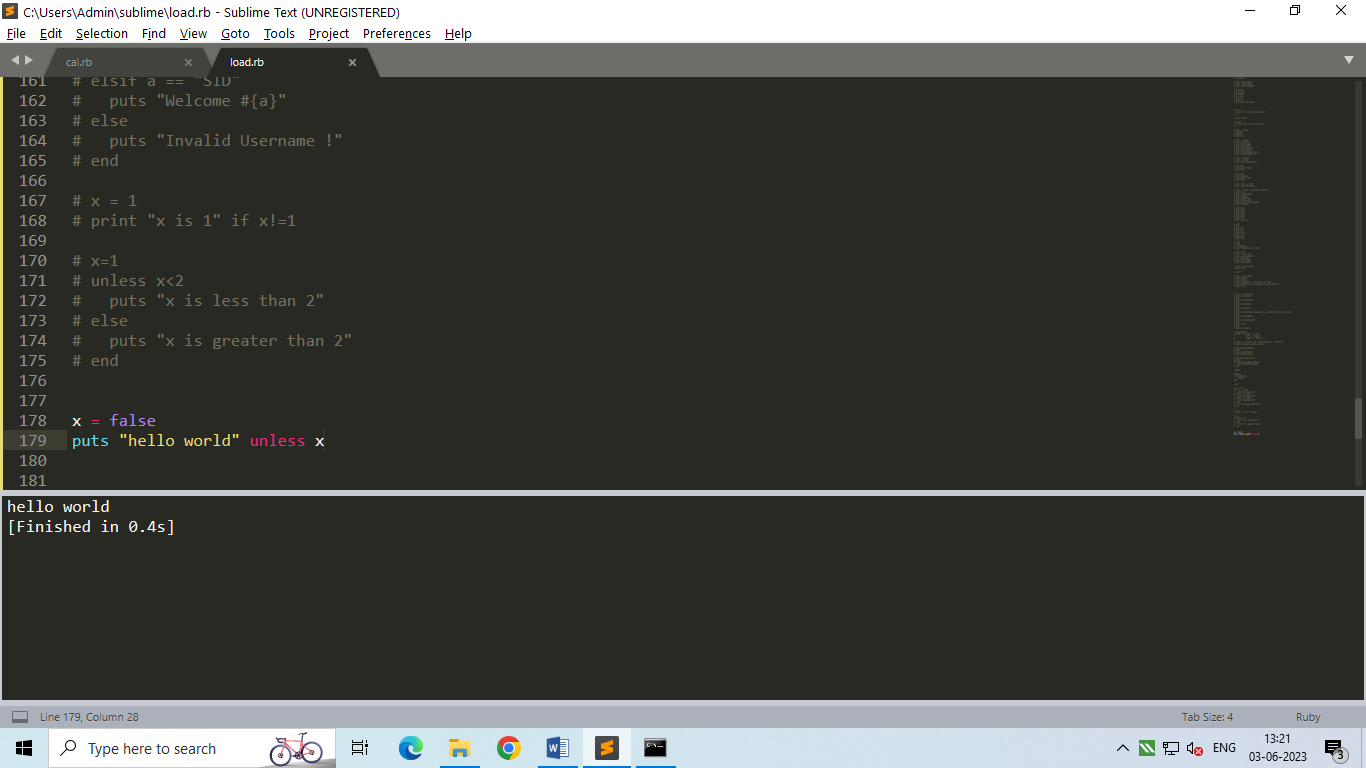
It excutes code if ‘unless condition’ is false.

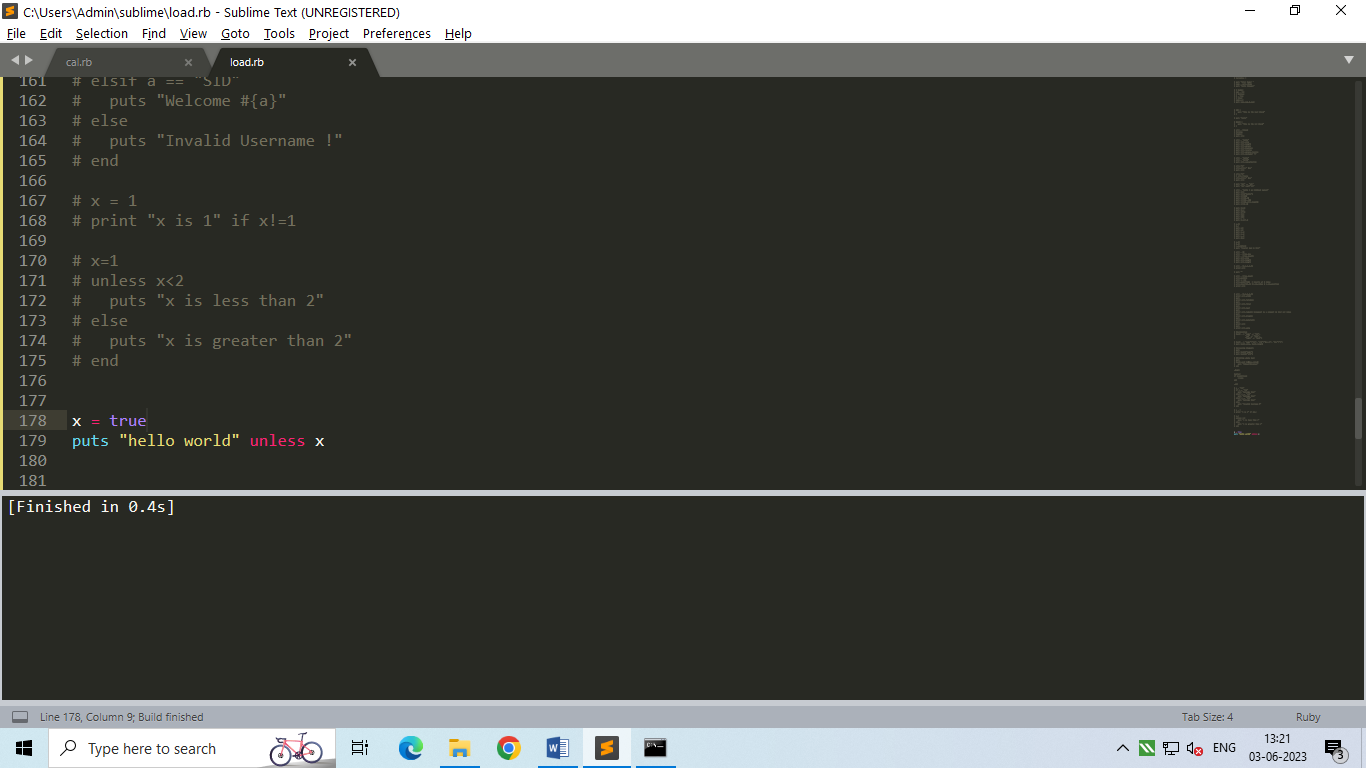
If ‘unless condition’ is true it executes else.





* Unless modifier:





1. Case :

* Syntax:

case expr

when expr1

//code

when expr2

//code

**.**

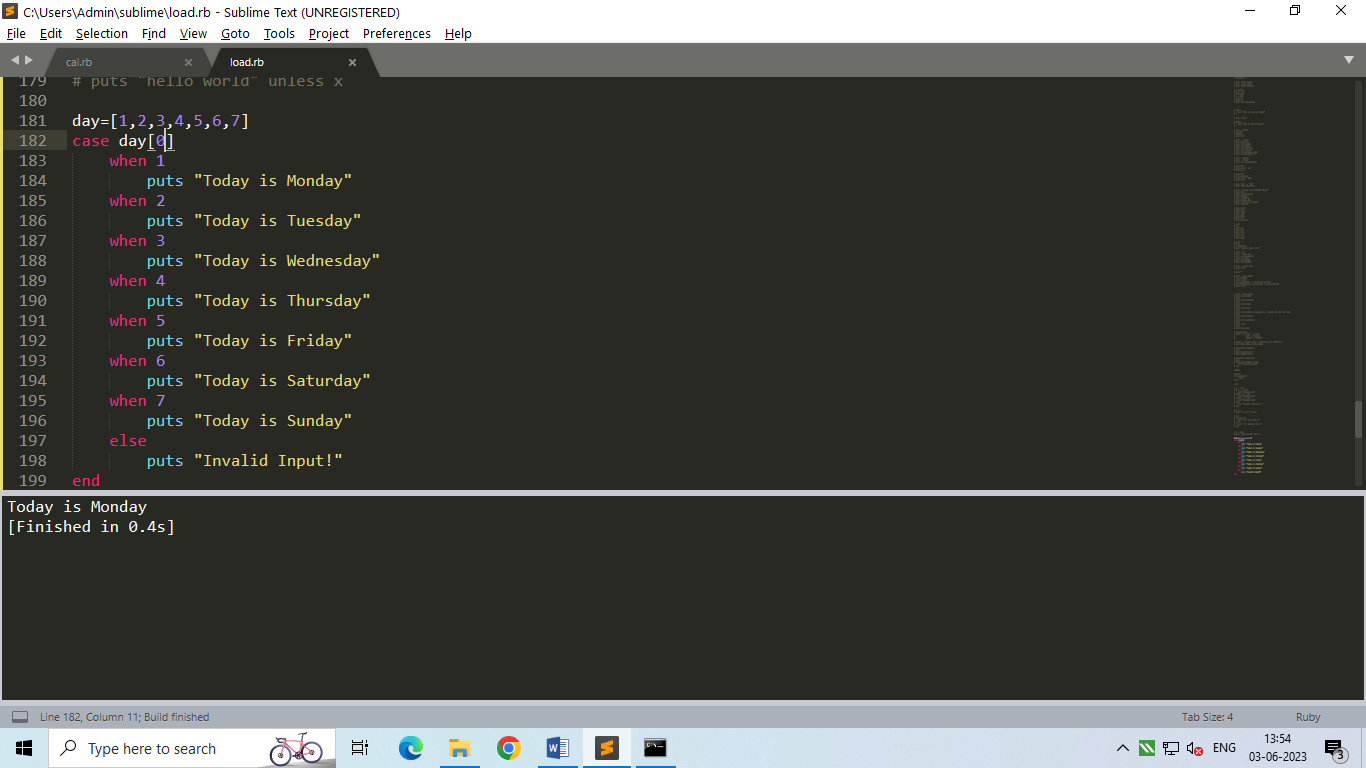
**.**

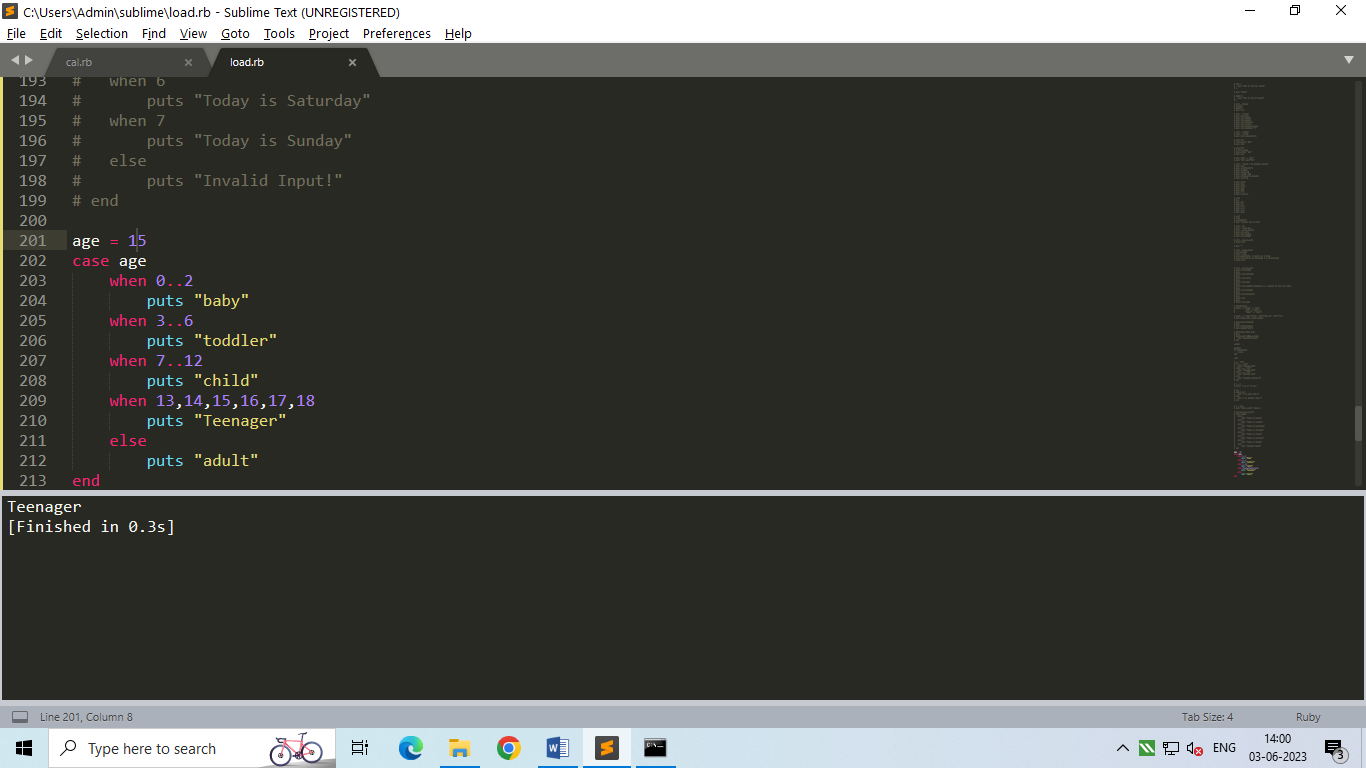
else

//code

end

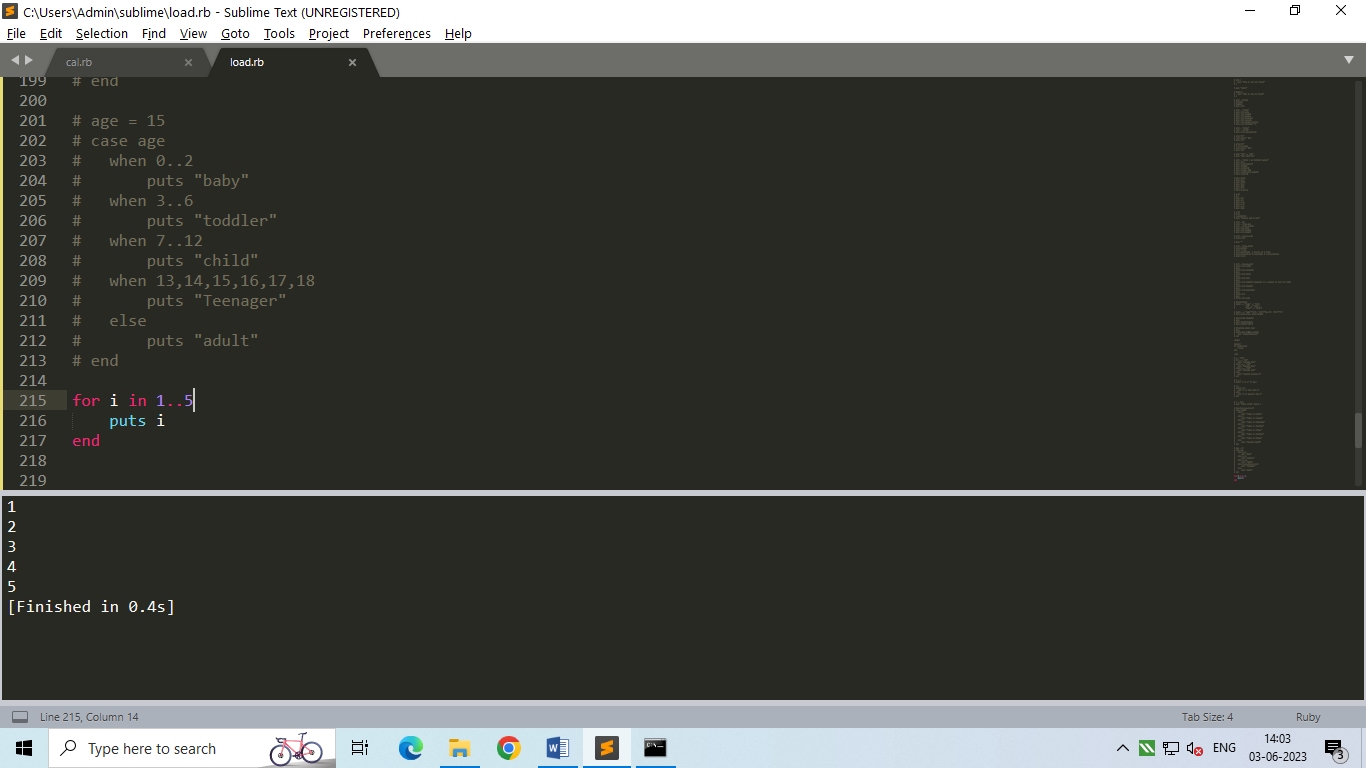
* eg:

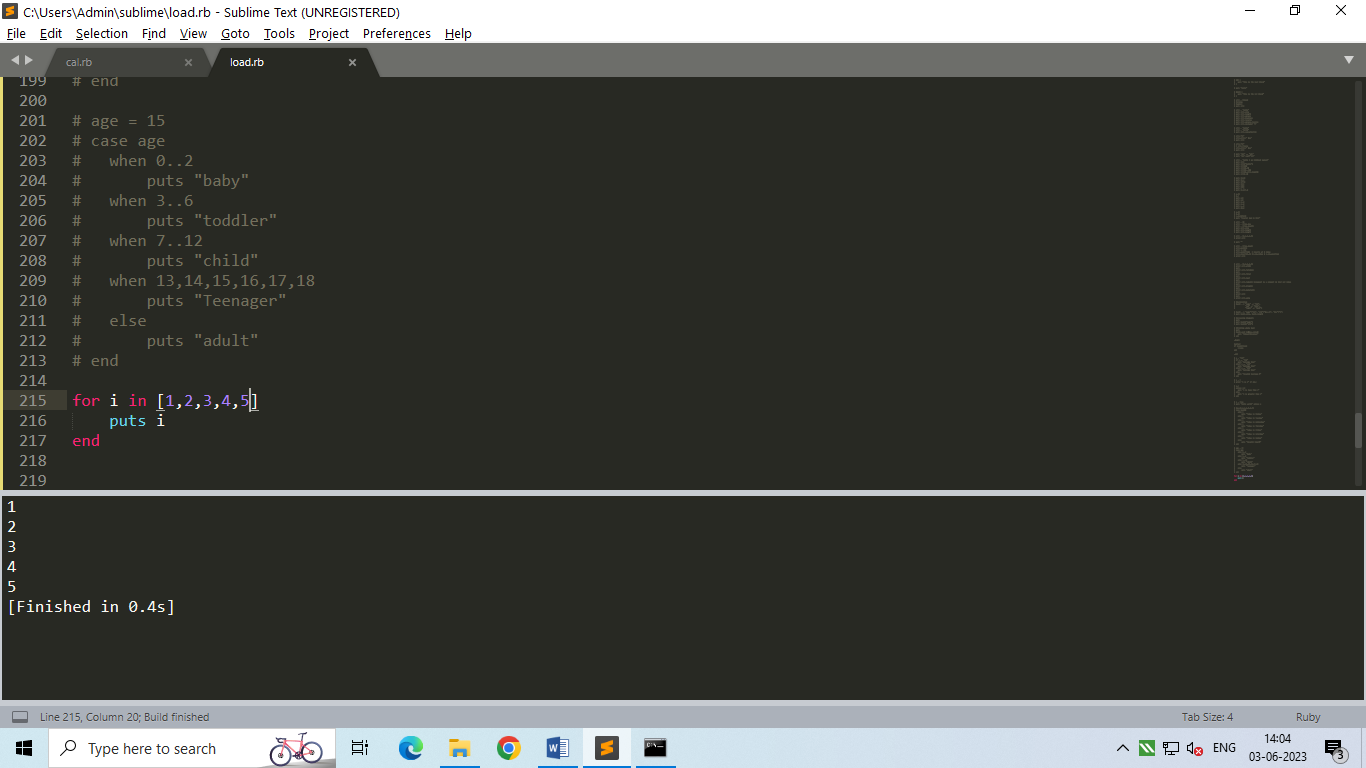




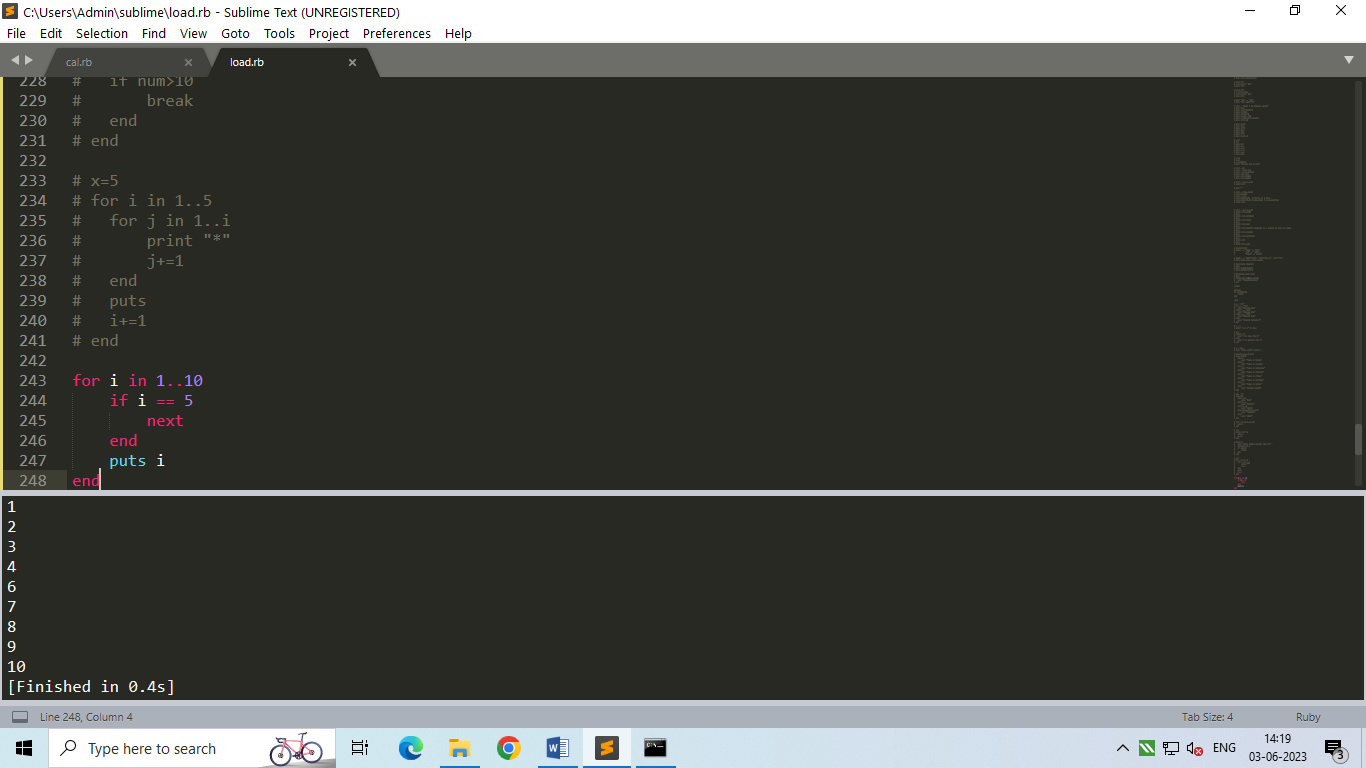
1. Loops :

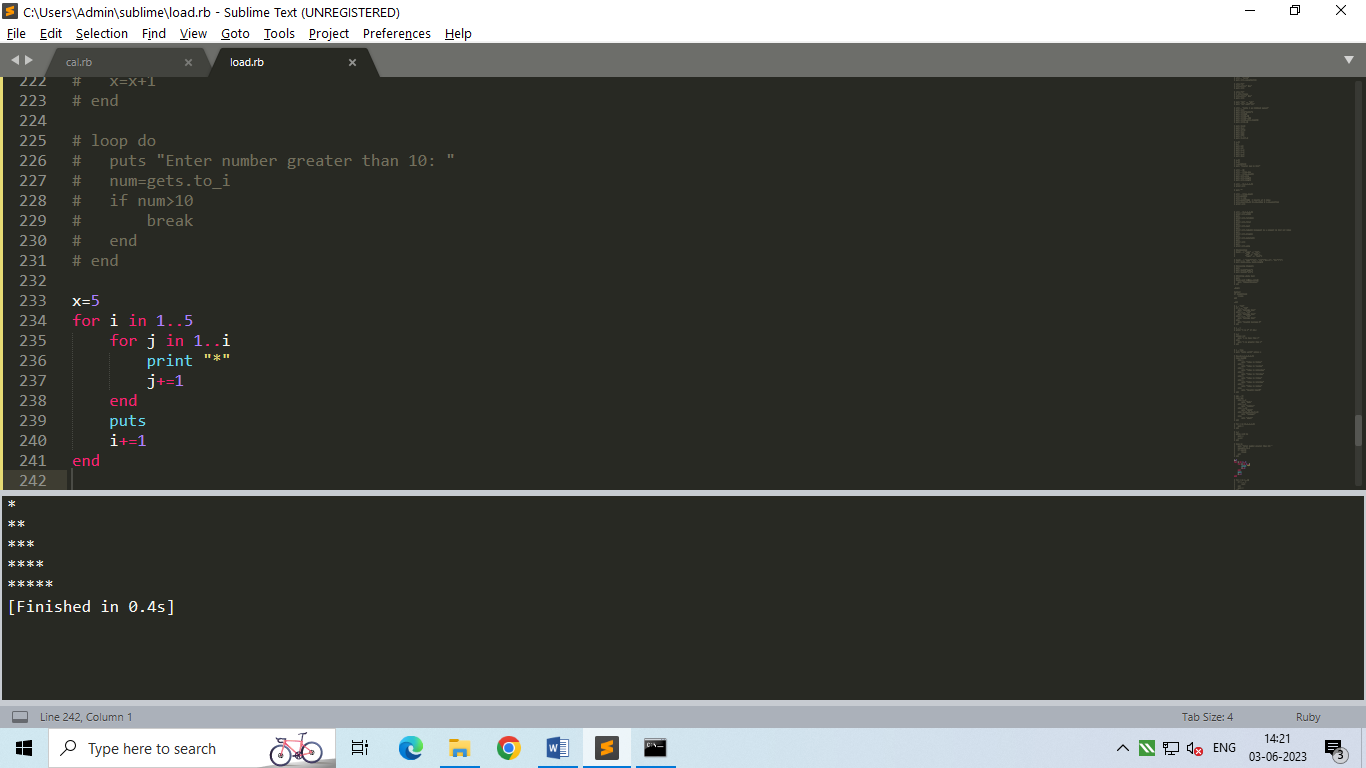
* For loop :



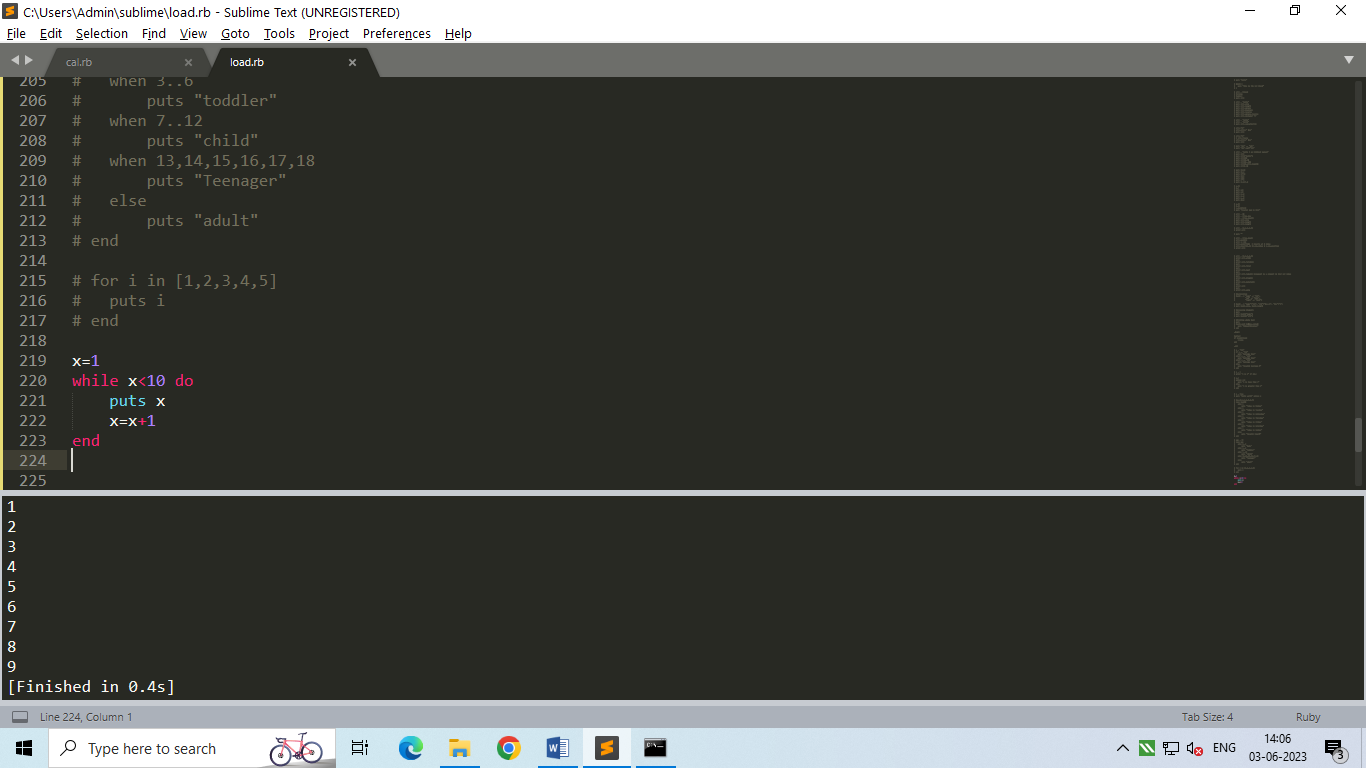


‘next’ is used to skip a condition

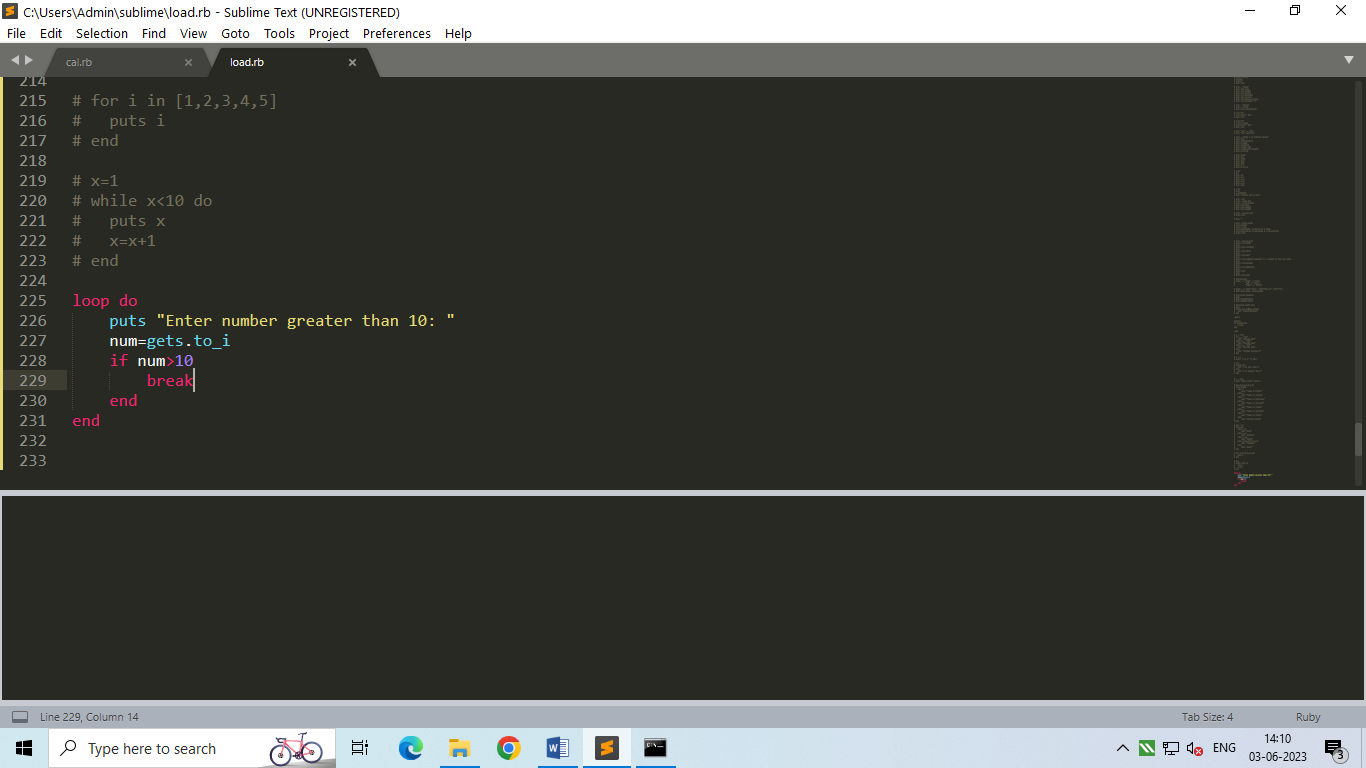


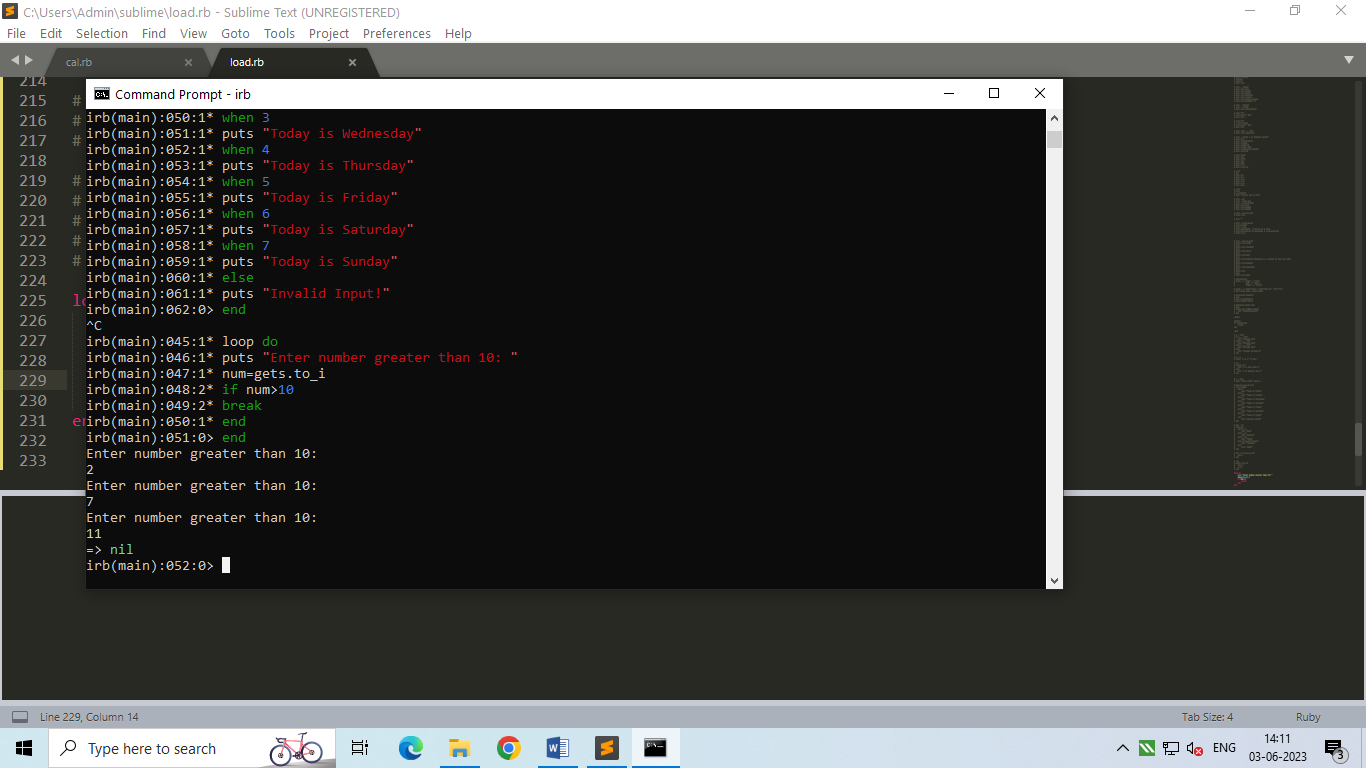


* Do..While Loop :

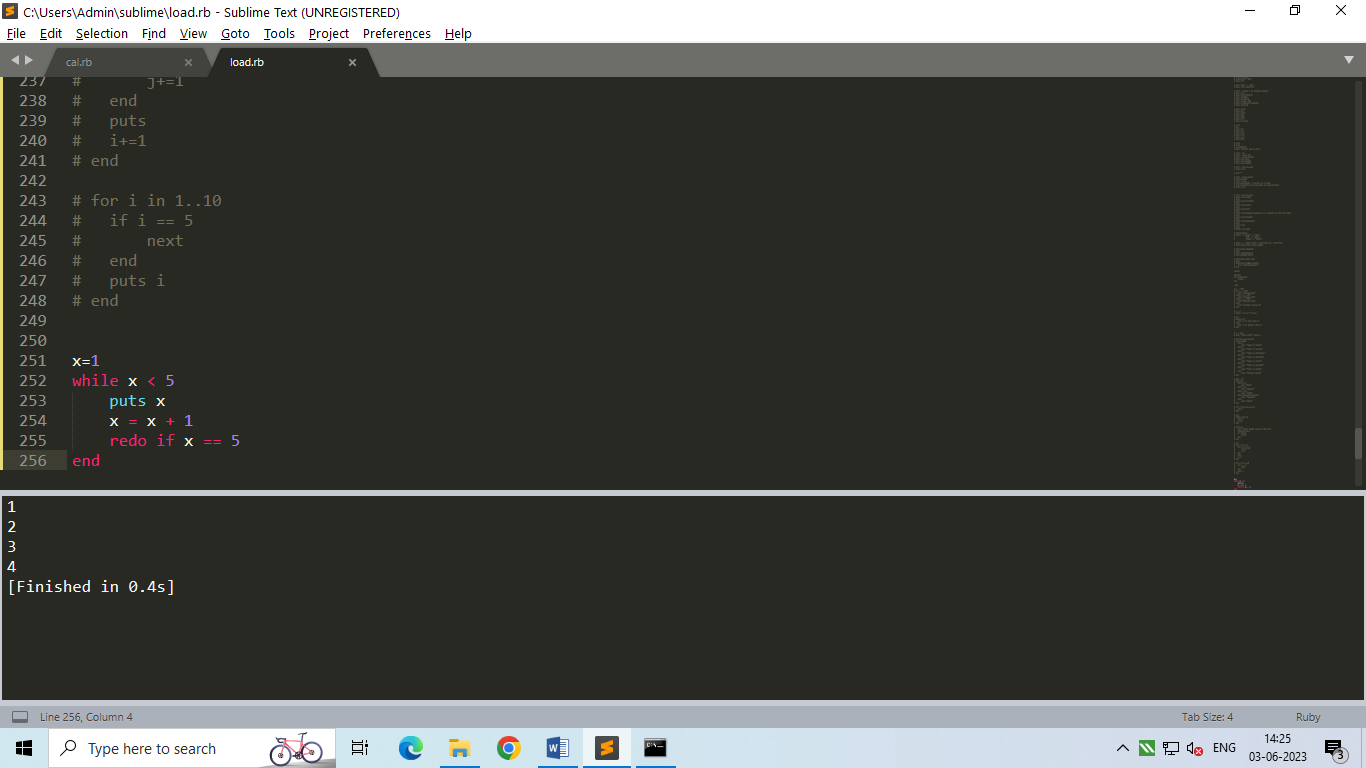


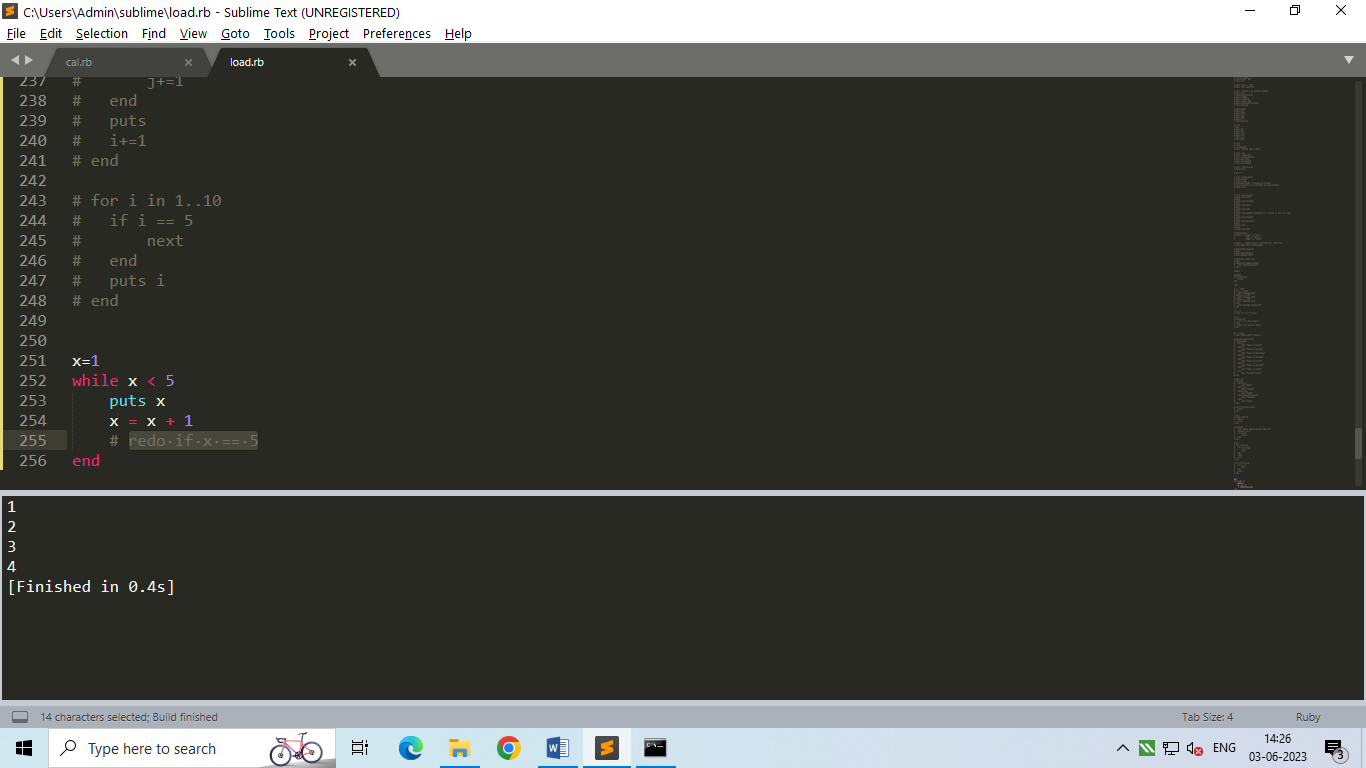
* Loop…do :

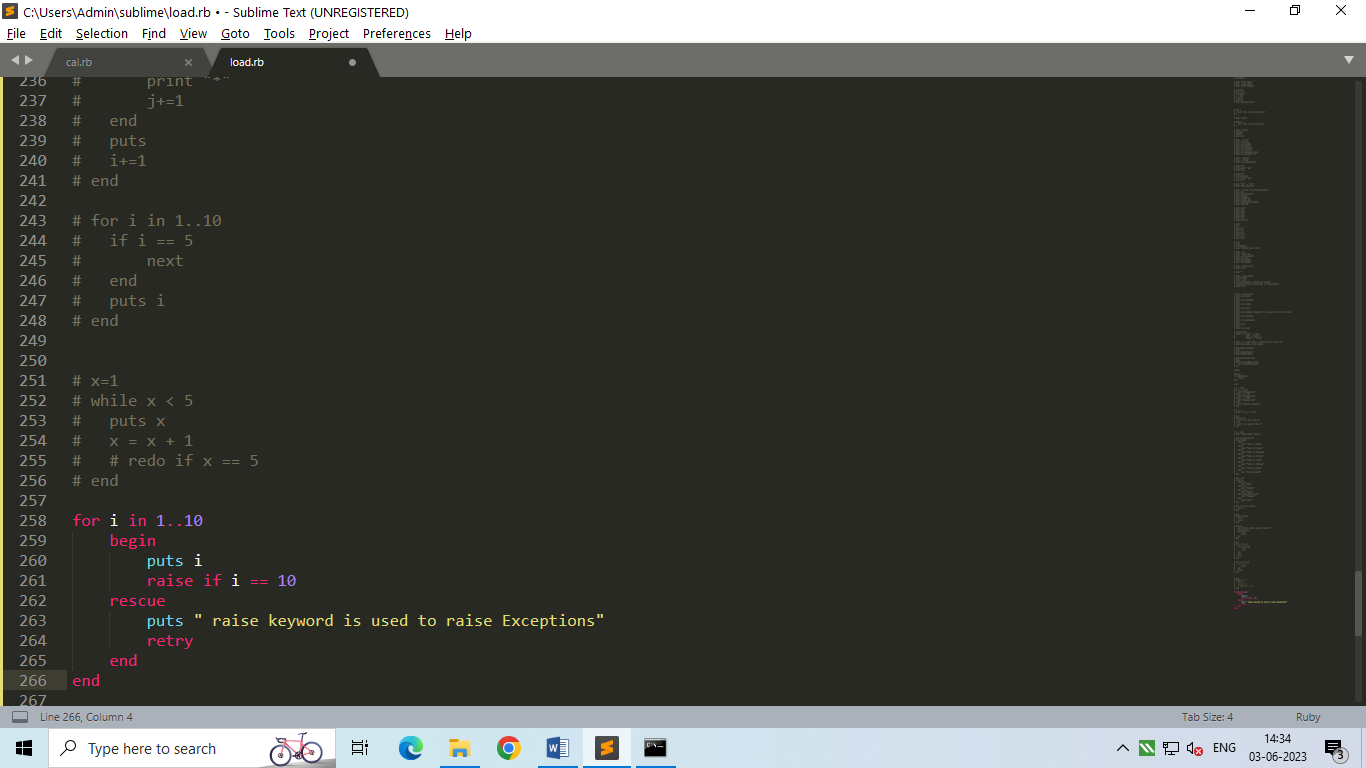




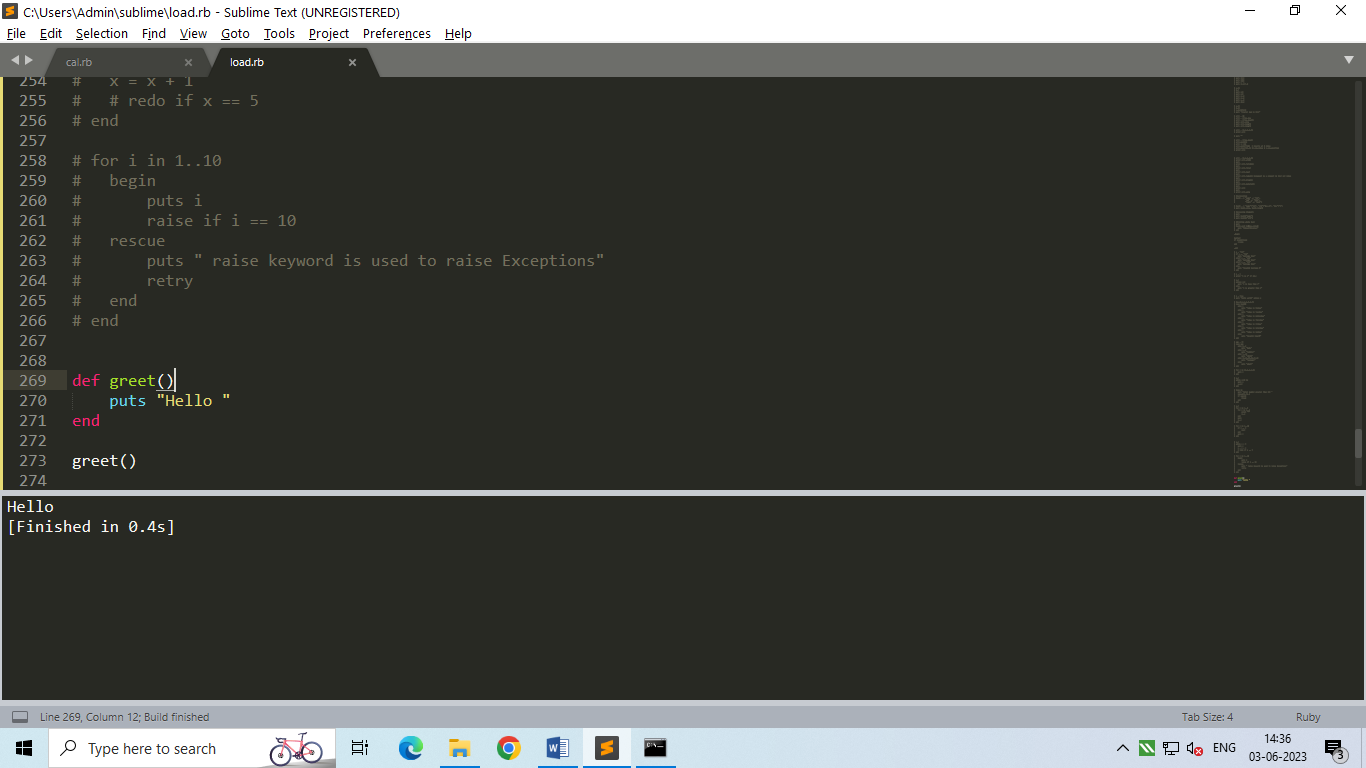
* Redo : repeats current iteration.
* Retry: repeats whole loop from start. It works only in BEGIN and rescue block

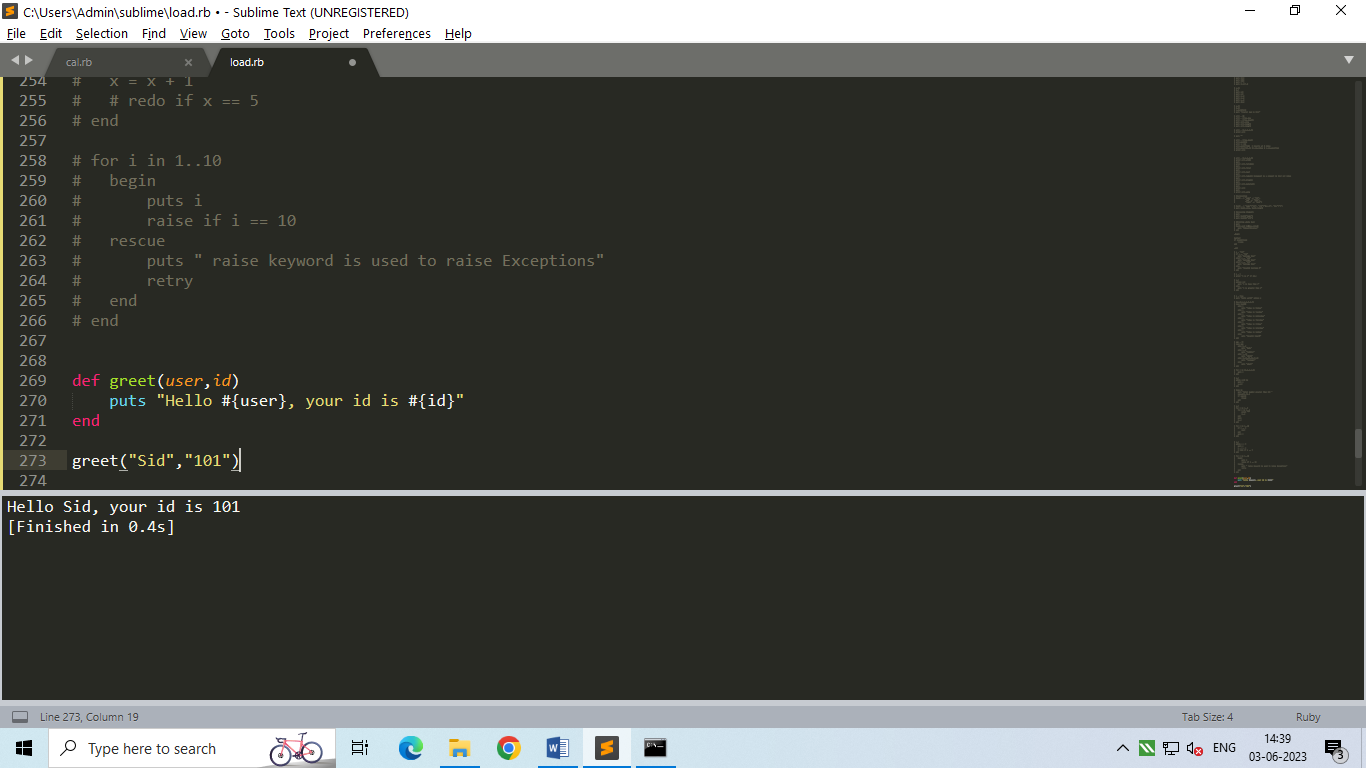


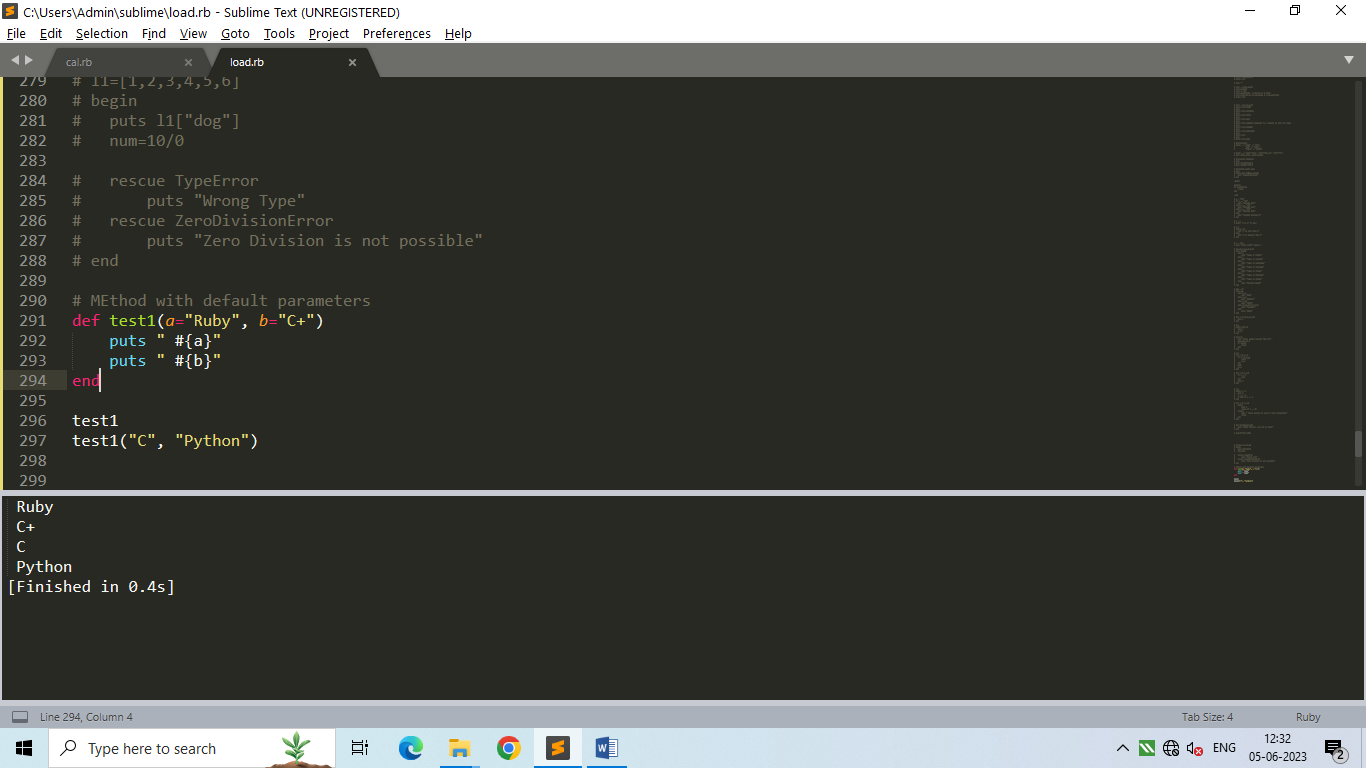


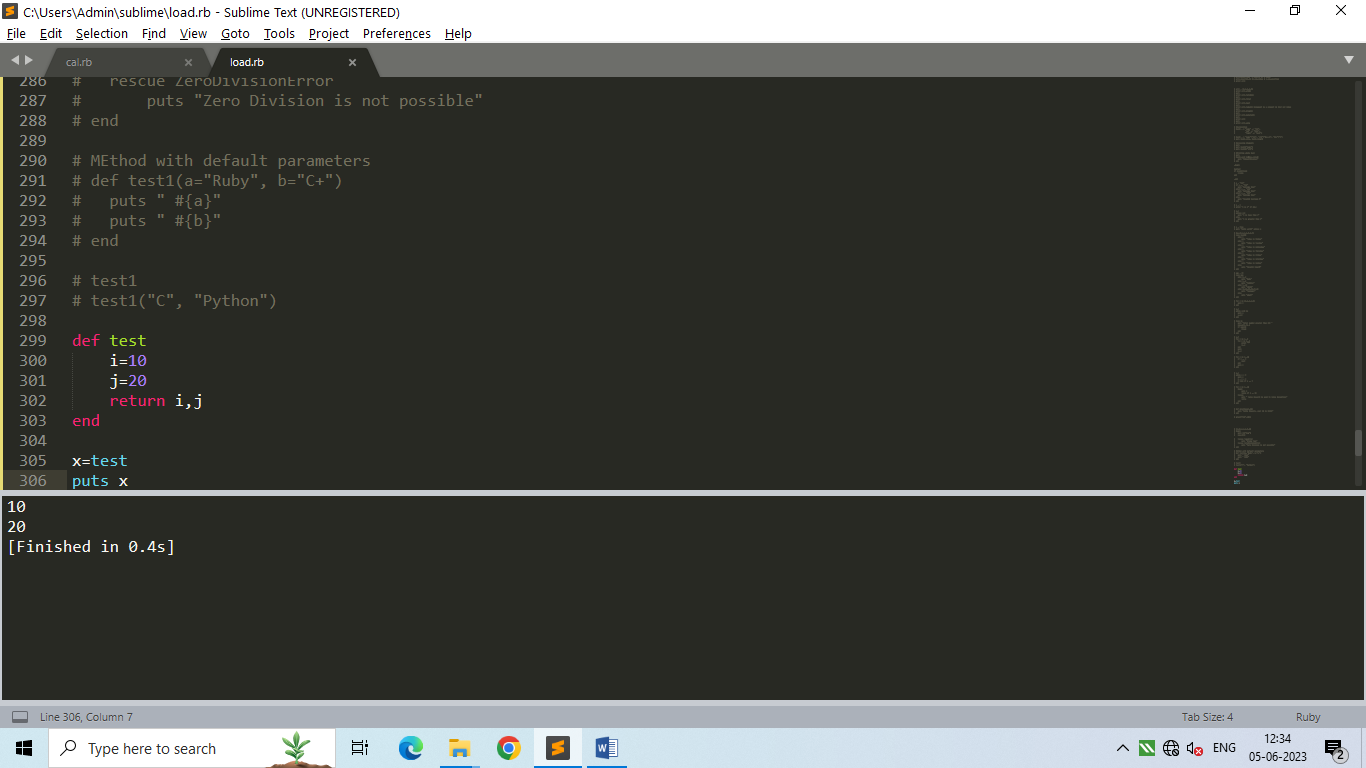


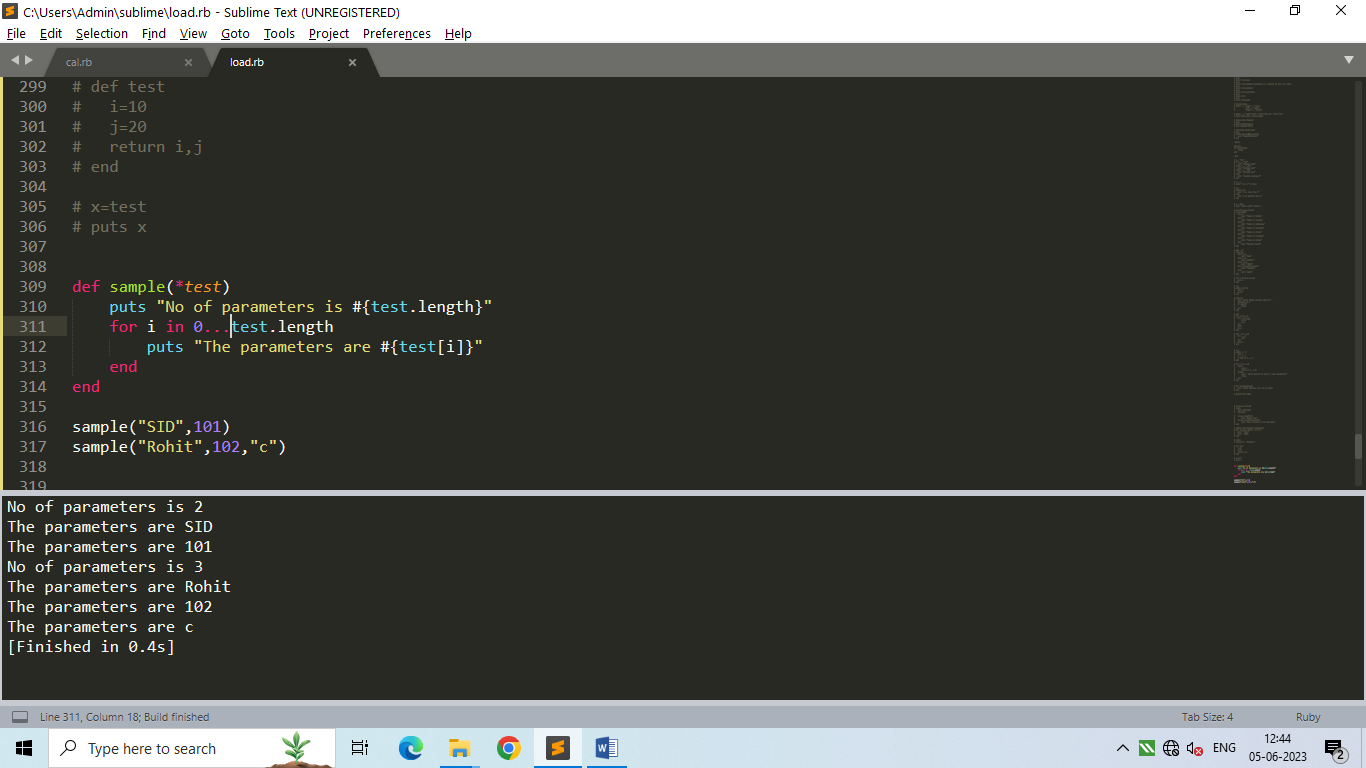
1. Methods :











* … = it doesn’t includes the last index
* .. = it includes the last index

1. Exception Handling :

Syntax:

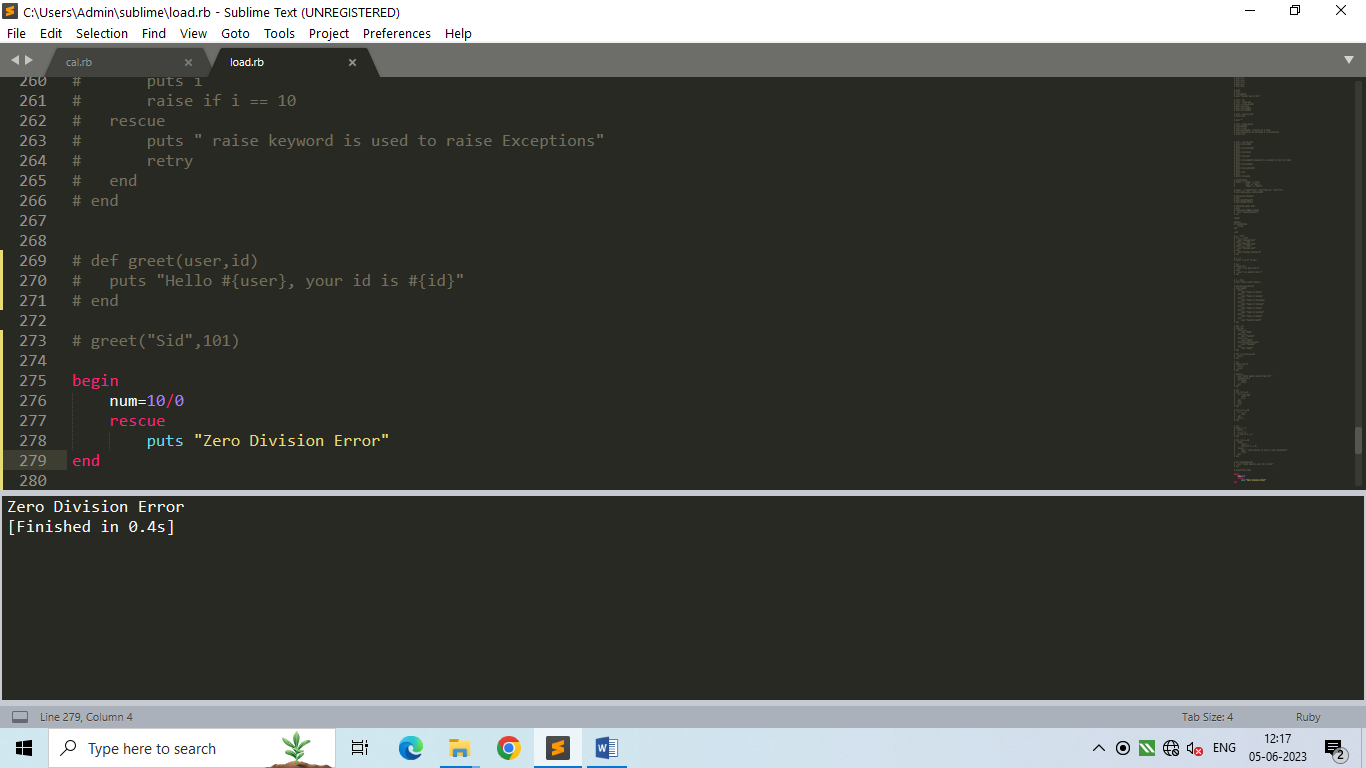
begin

//code

rescue

//error msg

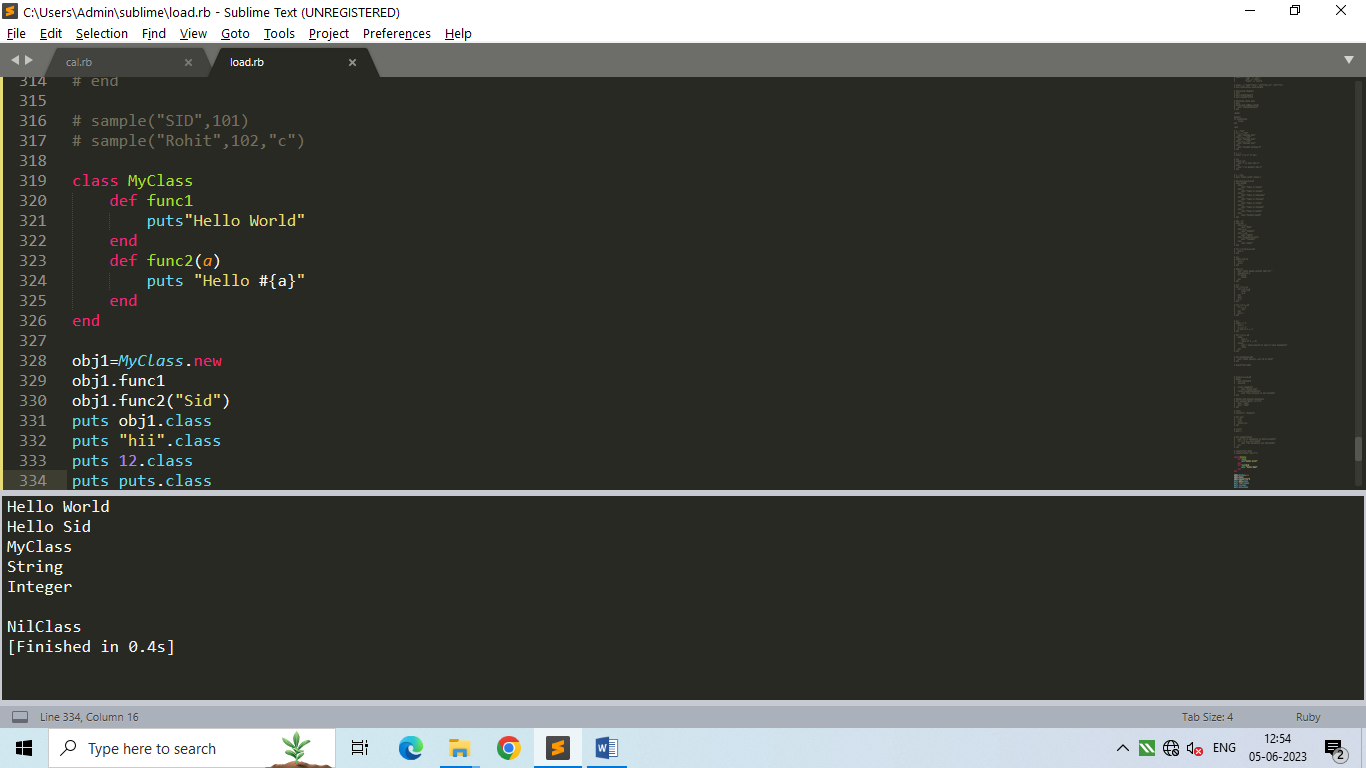
End



1. OOPs :

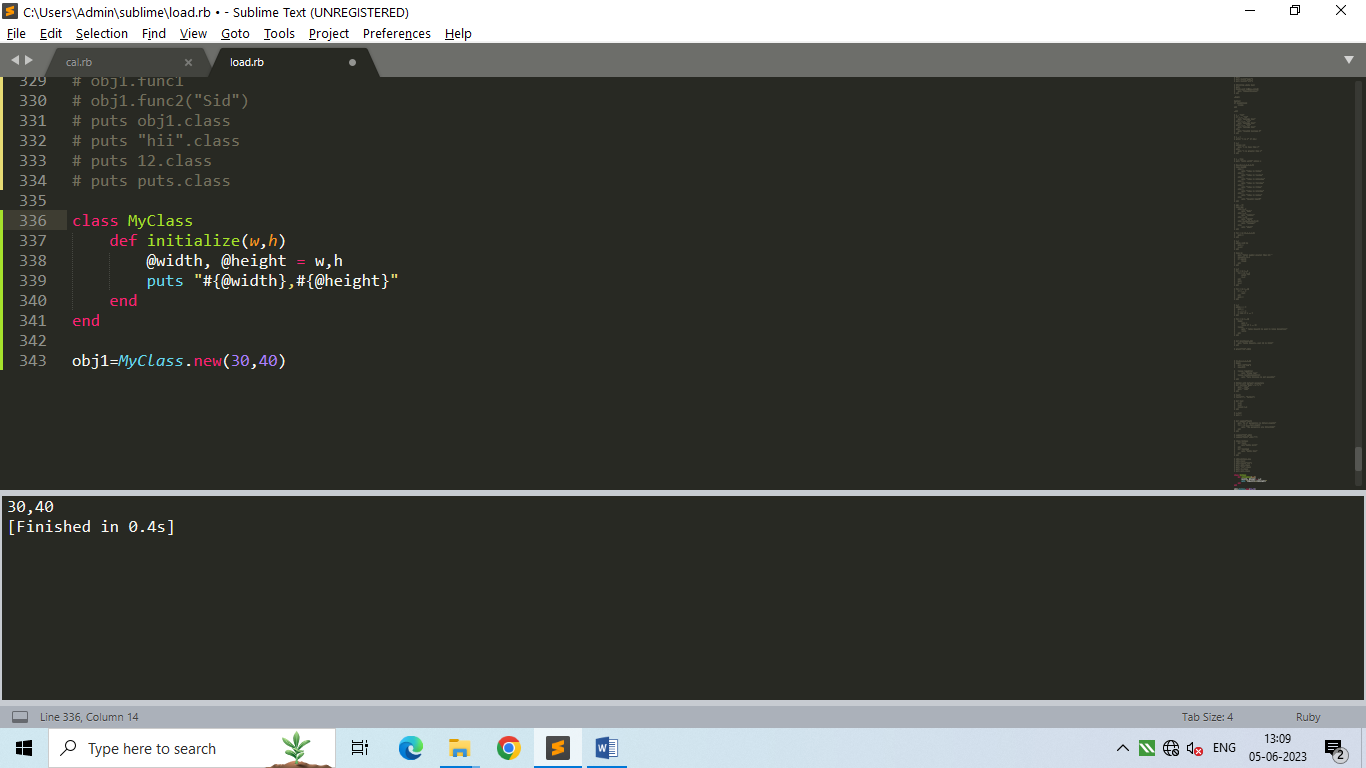
Name of a class in Ruby should always be capital.

* Basic :

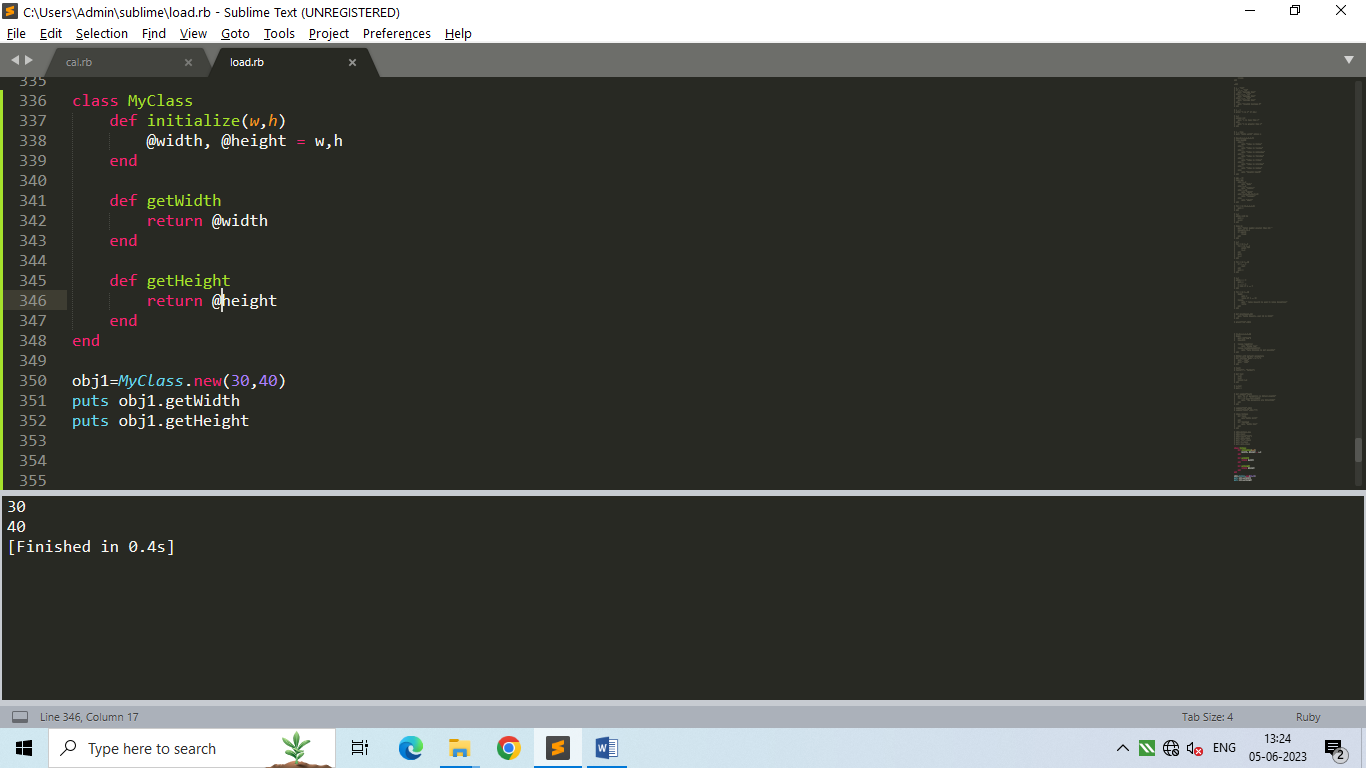


* Getter, Setter, Initialize :

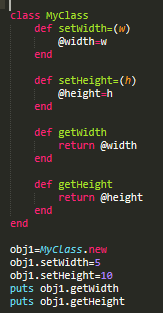
1. Initialize methods : initializes class variables( scope remains inside class ) at the time of object creation. It acts a constructor for a class in Ruby. Its automatically called when object is created.

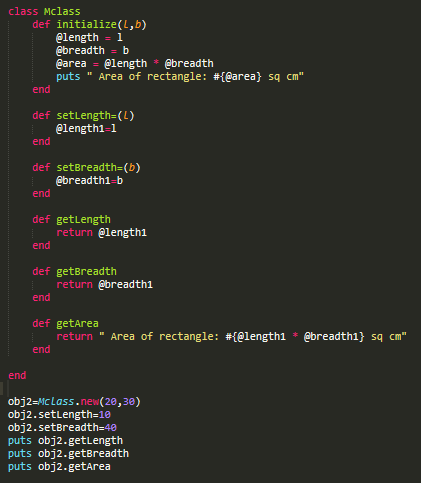


1. Getter Method :

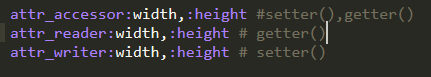


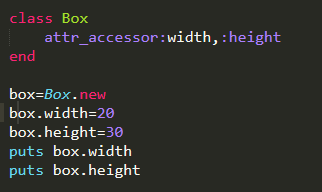
1. Setter Method:

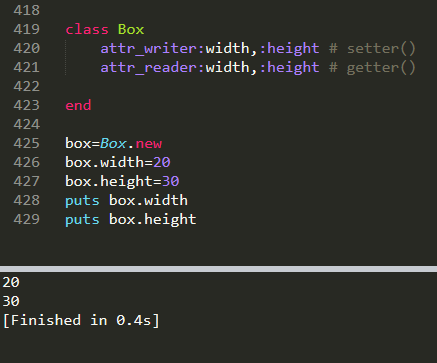




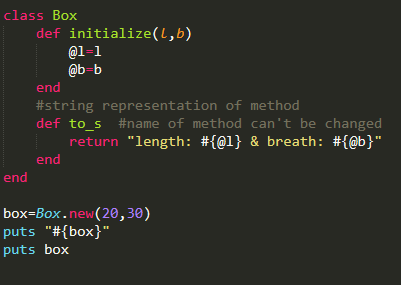
1. Shortcuts for setter, getter, setter-getter

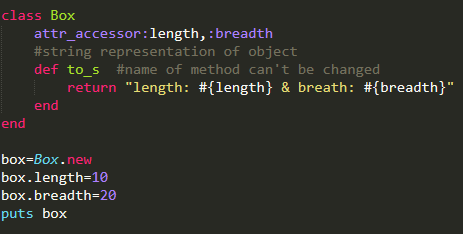






* to\_s :



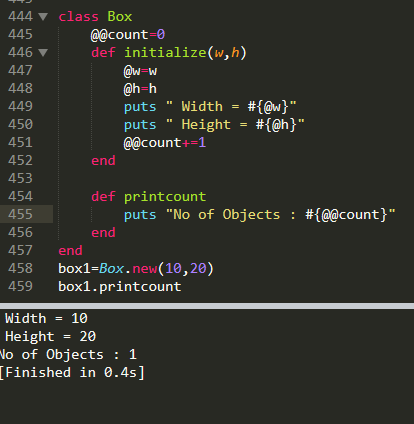


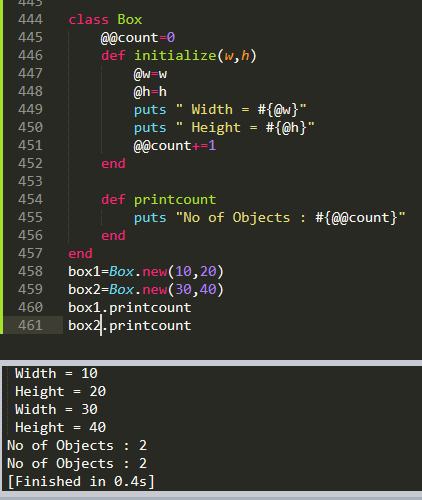
1. Class Variable:

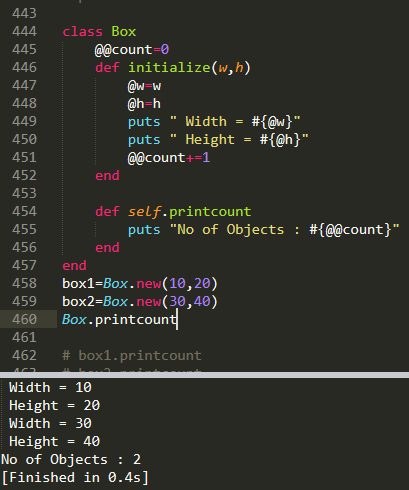
* **Syntax : @@variable\_name**

**It remains constant throughout the class.**

**self.method\_name can be used to create methods that can be called without objects of class.**







1. Bocks & yield, with and without parameters in Ruby:

Features of Blocks: A Block consists of chunk of code, A block has a name, Codes in a Block is enclosed with braces{}, A block is invoked from a function with the same name as that of the block, We invoke a block using the yield statement .

* Syntax:

def method\_name()

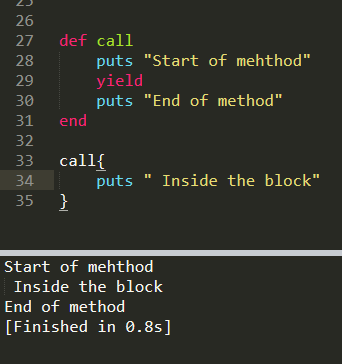
//code

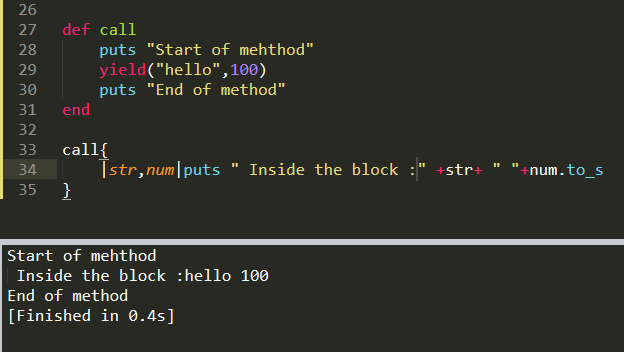
yield

block\_name {

//code

}

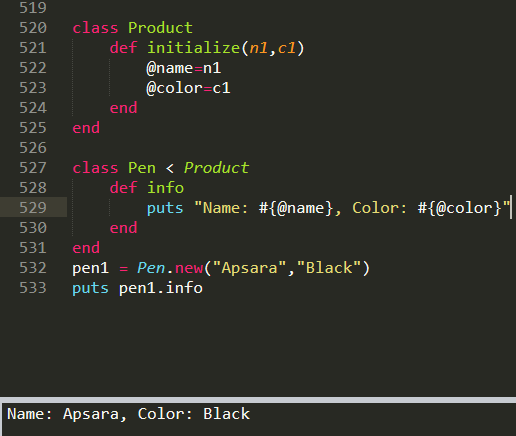


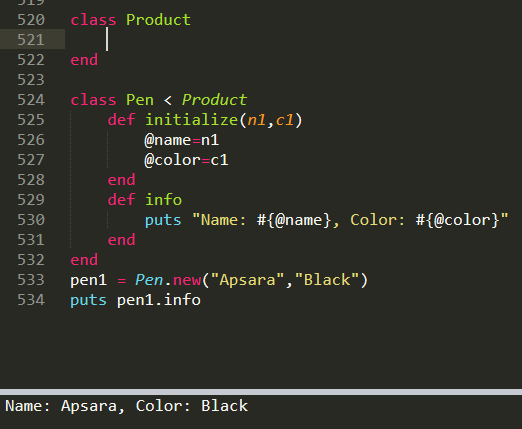


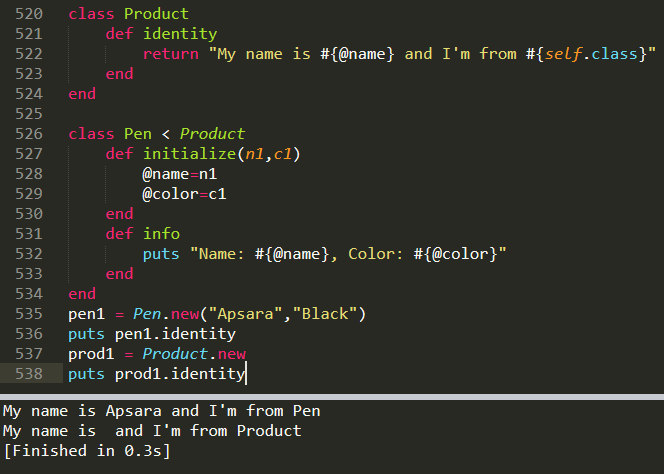
1. Inheritance :





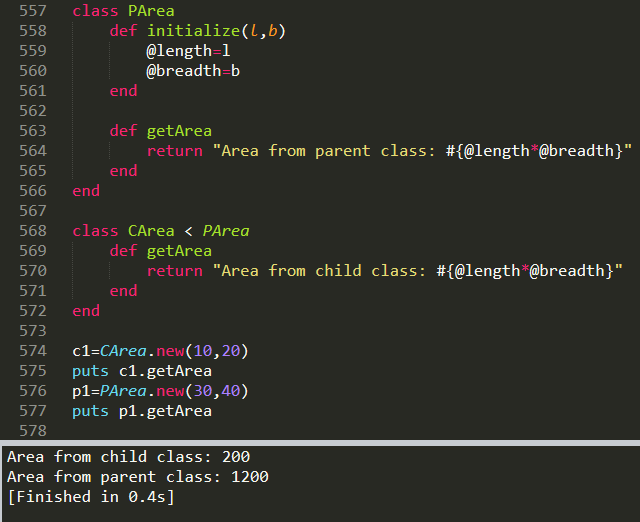




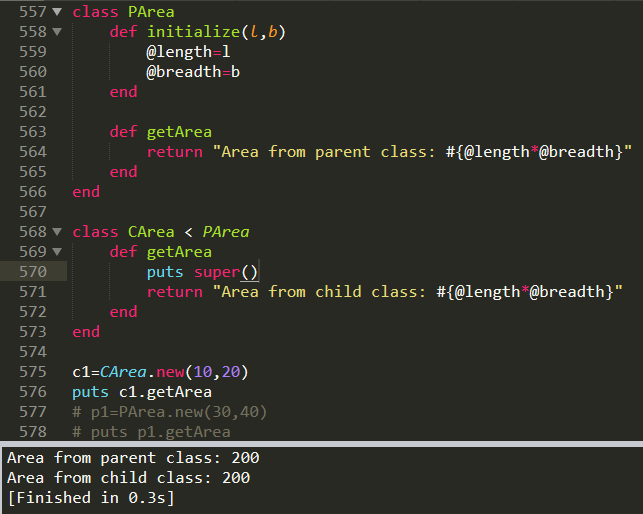


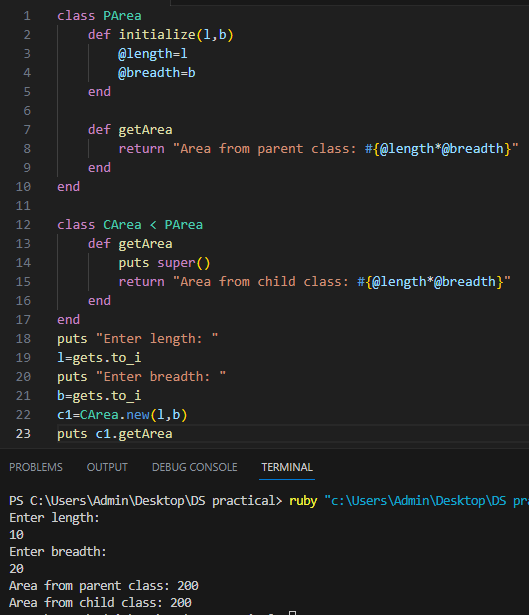
1. Method Overriding :

If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.

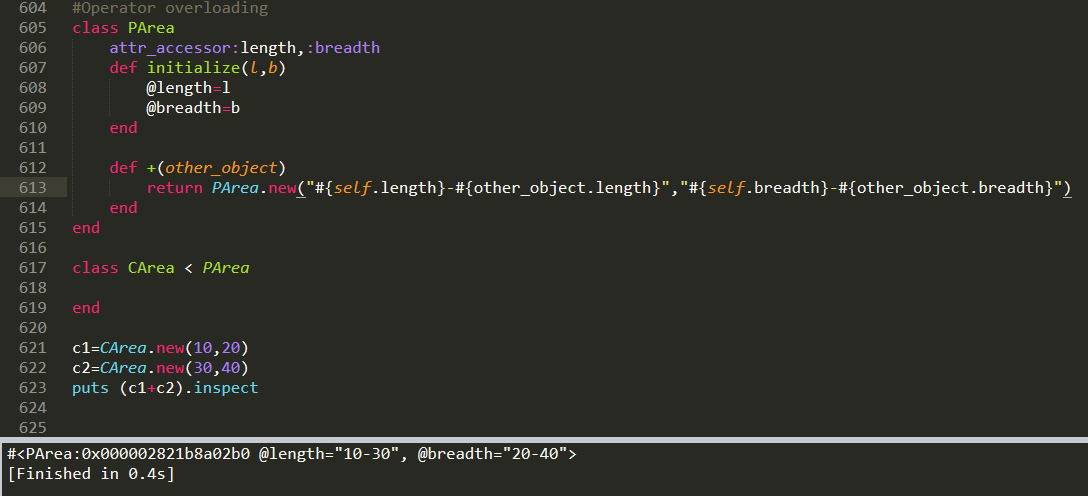


* super() keyword:





1. Operator Overloading :



1. Modules :

Modules are a way of grouping together methods, classes and constants.

Modules provides a namespace and prevent name clashes.

Modules implement the mixin facility.

* Syntax:

module Module\_name

constants\_name=value

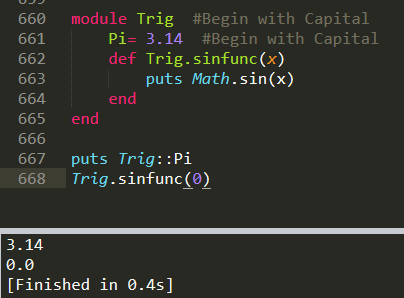
def Module\_name.Method\_name(x)

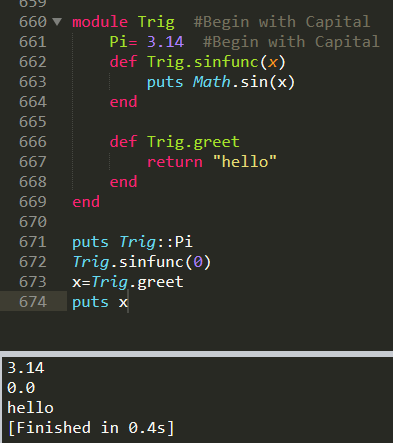
end

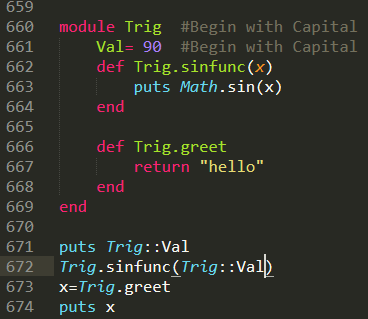
end

puts Module\_name::constants\_name

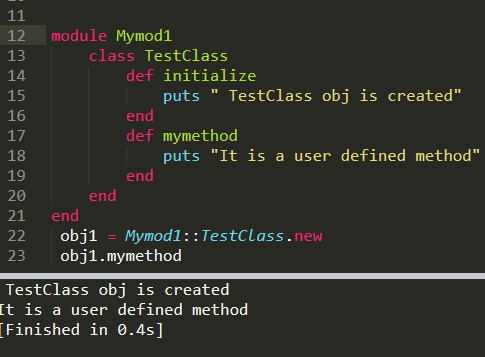
Module\_name.Method\_name(Module\_name::constants\_name)

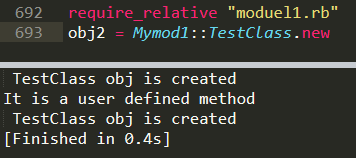






* Class inside a module :





1. Require statement:

It is similar to include statement of C and C++ (eg : #include<stdio.h>) and import statement of Java (import java.sql.\*) .

It is used to include any defined ruby program in 3rd program body.

* Syntax:

// require statement :

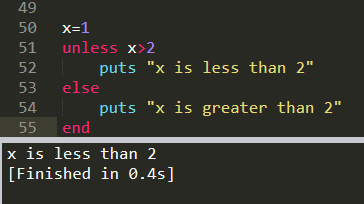
$LOAD\_PATH<<'.'

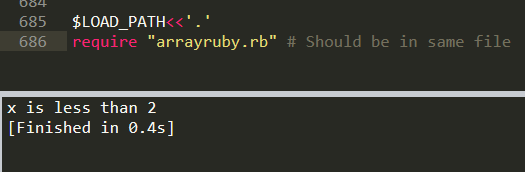
require "filename"

//require\_relative :

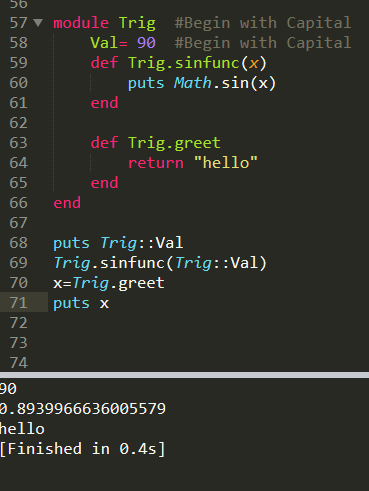
require\_relative "filename"

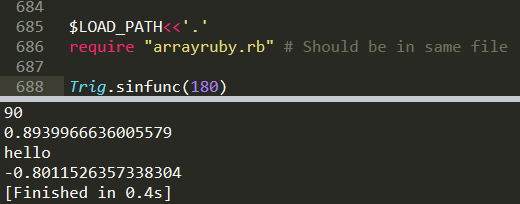
1-eg:

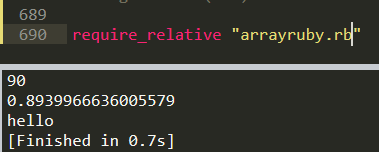




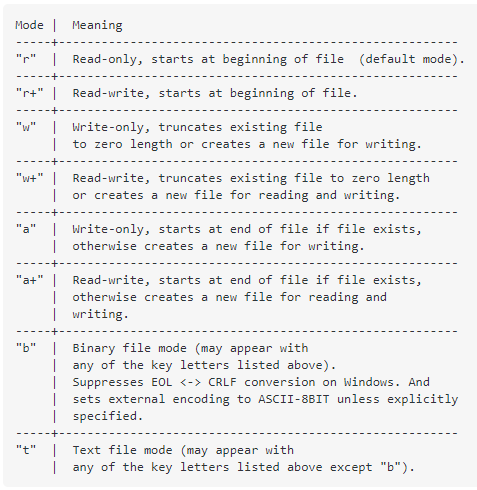
2-eg :



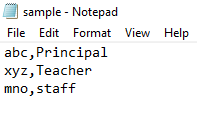


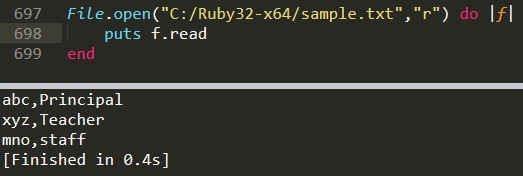


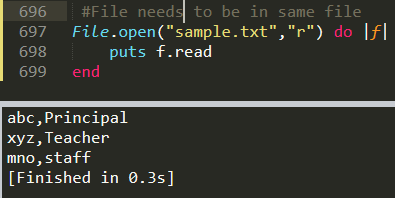
1. File Handling :



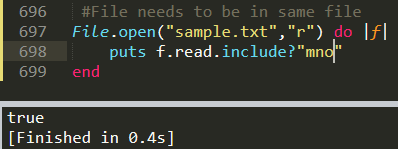
1. Read: (“r”)

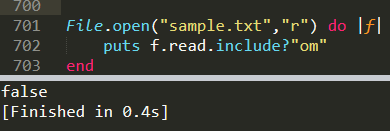






* + - include : (to search a string)



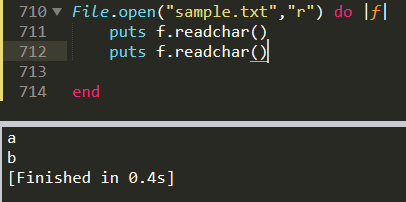


* + - readline:

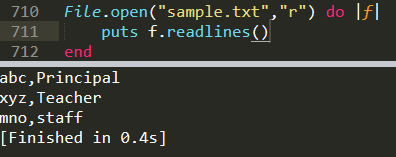
The diff between readline method and read method is that read method returns all records from a file while readline method returns line by line records.

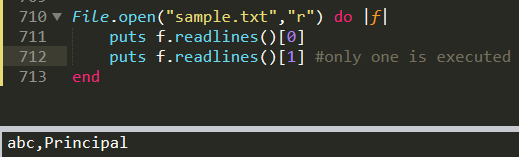
|  |  |
| --- | --- |
|  |  |

* + - readchar():



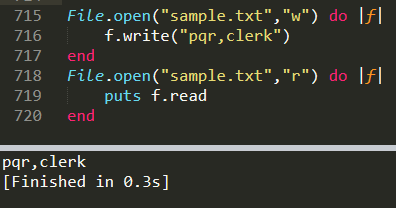
* + - readlines():





1. Write:

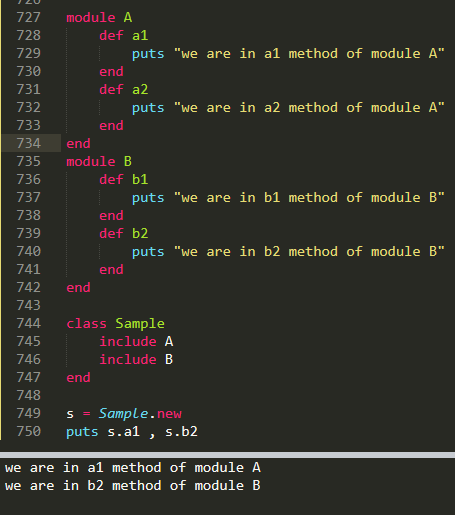
It overwrites the file. It can create a new file.



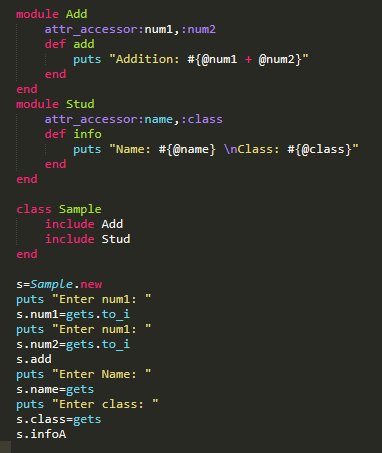
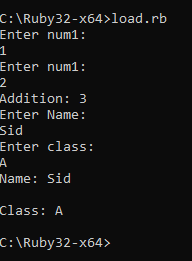
|  |  |
| --- | --- |
|  |  |

1. Mixin :

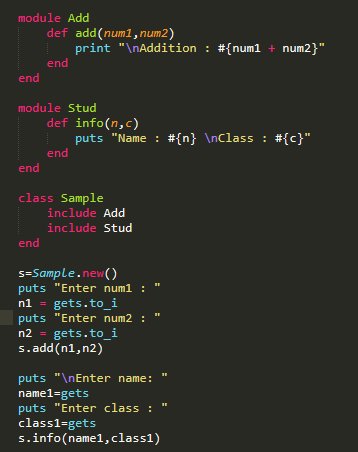
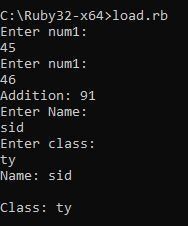
Multiple Inheritance is not supported in Ruby directly. For it we have to perform Mixin.



1. Create two module in a single file use first module to add the number and second module is use to display the name and class of a student take a data from the user on console call the module by creating an object of a class. **#By accessor Method**

**# BY Normal Way**

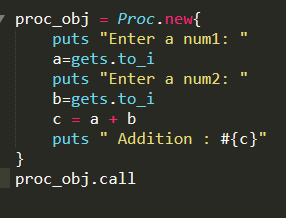
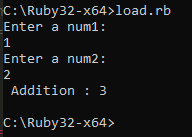
 

1. Procs :

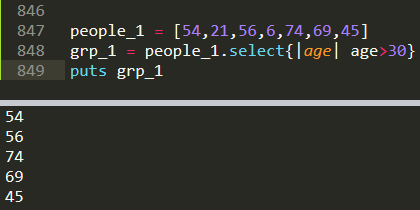
“Everything in Ruby are objects.”

But Ruby Blocks are not an object.

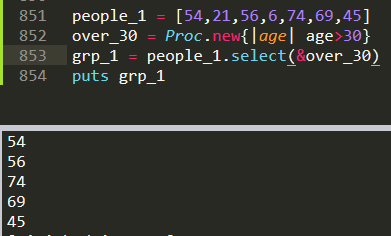
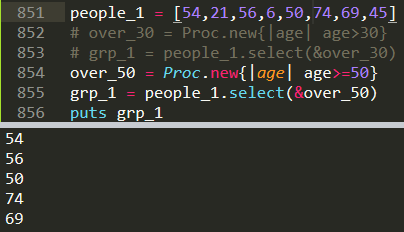
Procs class can turn Block into an object by wrapping block in its instance. This turns our block into a first class function and then we can perform all sorts of things with blocks as we can do with a normal object.

* Block without object:



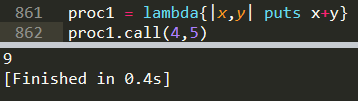
* Block with object :

* Multiple arguments :

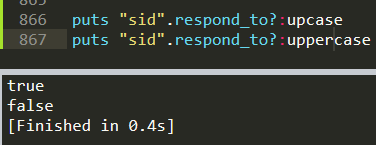


* Lambda :



1. “respond\_to?” Method :

To check whether the given method exists for a given object or not, we use the method called .respond\_to?



1. Send() method :

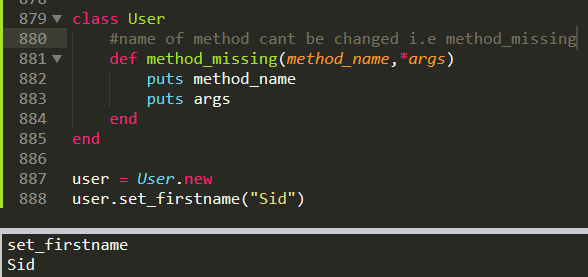
It is used to pass message to the object. send() is an instance method of the object class.

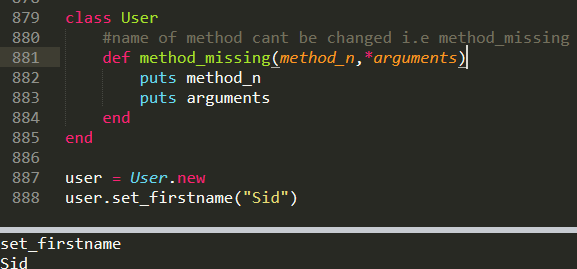
The first argument in send method is the message that you are sending to the object (i.e the name of the method).It could be string or symbols, but symbols are preferred. Then arguments those need to pass in method, those will be the remaining arguments in send() .

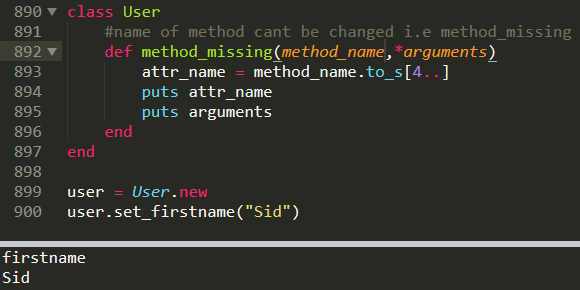


1. method\_missing :

“method\_missing()” method returns the name of method missed by the programmer at the time of declaration and also returns the missed arguments of the method.







* Rails Project:

1. Create a new folder and set the path of the same folder in cmd and type the following command to create the project directory: rails new proj\_name.
2. Now in cmd, set the path till this project directory.
3. Now type the following commands : a)bundle install, b) rails s/server. This commands will show you the localhost and that you can run through browser.
4. To create new page in rails :

a) Open the created rails directory in code any editor, in the code we can see multiple folders.

b) In that open, app folder. First of all we need to create the controller, for that right-click on controller folder and select new file. Save this file as ‘home\_controller.rb’ and type the following code:

class HomeController < ApplicationController

def index

end

end

Here, index is the action.

c) Now go to config folder and open ‘routes.rb’ folder and use get method to call the action(index) through controller: get ‘home/index’

d) Now we need to create home folder in view folder. In this folder, we need to create html file(index.html.erb).

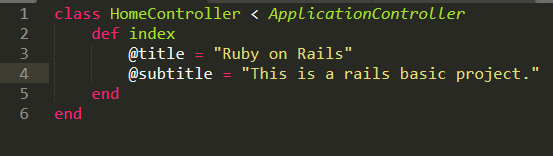
e) Now run the server and execute the project with the following path : <http://127.0.0.1:3000/home/index>

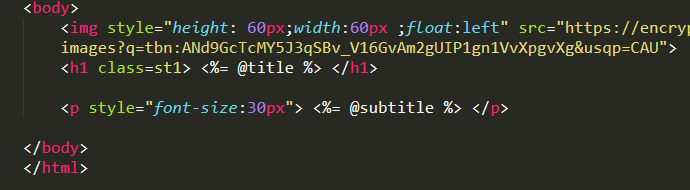
f) If you want to use controller to show the content, use the following syntax in routes.rb file: “get 'home' => 'home#index' “. To execute this run the following path: <http://127.0.0.1:3000/home>

g) To set the root page, type the following syntax : “ root 'home#index' ”. To execute this run the following path: “ <http://127.0.0.1:3000> ”

* instance variables:

Ruby Tag: “ <%= code %>



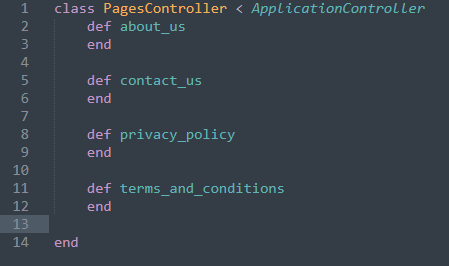


OR

1. Run this Command:



1. Create Controller



1. Create pages: about\_us,contact\_us,privacy\_policy

