* When proficiency : try out the vim IDE.
* Make Towers of Hanoi

Basics

You call a rb script in the console with

Ruby test.rb

LOOPING

print "Quel est votre prénom ?"

prenom = gets.chomp

prenom.capitalize!

puts "Vous vous appelez #{prenom} #{nom} et vous venez de #{ville}, #{pays}!"

variable : #{mavariable}

gets recupere le dernier input de l’utilisateur, mais avec un retour a la ligne automatique ajouté par ruby. .chomp permet de supprimer ce retour a la ligne.

On modifie les strings avec une approche objet.

Prenom.capitalize

Ce que fait ruby dans ce cas, c’est en fait copier le contenu de prenom et le capitaliser. Si on veut modifier directement le contenu de prenom sans le copier, on rajoute un point d’exclamation.

Prenom.capitalize !

* Get input while making sure it’s an integer : put gets.chomp entre parentheses et integer:

print "Saisir un entier : "

saisie = Integer(gets.chomp)

* If

then

Elsif

End

* Unless

Else

* If saisie contient un s, doubler le s

if saisie.include? "s"

saisie.gsub!(/s/, "ss")

else

puts "Rien à changer !"

end

.include ? 🡺 return True

* Print numbers from 1 to 10

for nombres in 1...10

puts nombres

end

* Puts is short for put string
* The primary difference between them is that puts adds a newline after executing, and print does not.
* Boucles FOR: in ruby we also use the command “loop”

i= 0

loop do

i += 1

print "Ruby!"

break if i == 30

end

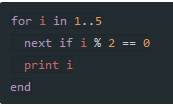
Otherwise we use “for” for the equivalent of “foreach… as”:

for i in 1..20

puts i

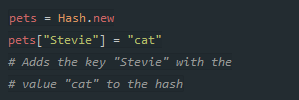
end

* **NEXT** is used to **SKIP** certain steps in a loop.



* Hashes

Hashes are collections of key-value pairs. Like arrays, they have values associated with indices, but in the case of hashes, the indices are called "keys."



**ARRAYS**

Exemple

empty\_arr = Array.new

=> []

matzes = Array.new(3, "Matz!")

=> ["Matz!", "Matz!", "Matz!"]

Syntaxe

# Array.new copy constructor

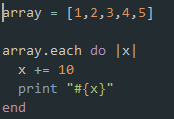
variable = Array.new(some\_array)

Exemple

more\_matzes = Array.new(matzes)

=> ["Matz!", "Matz!", "Matz!"]

* **FOREACH …. AS**

****

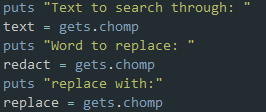
The |x| is actually important, it defines the needle in the haystack

*The « Are you Serious? » section of Ruby.*



The .times operator is used to repeat x times a command.

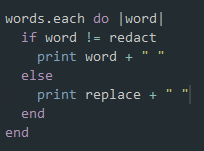
* **FIND AND REPLACE APPLICATION:**



UI where it asks the user input

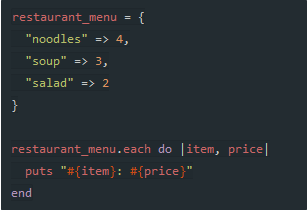


Creates an array with all the words of the string wherever it sees the delimiter “&nbsp;”



Glances at each word au cas par cas and replaces it if “word” = “redact” with “replace”

* Iterating over hashes:



* **HISTOGRAM**

Input a text, it will tell you how many of each word is there in it.



Input your text



Create array out of text.



Create a hash named frequencies



For each word in words, raise the counter by one frequencies[word] (la case est créee automatiquement dans frequencies



Sort by highest frequency

"a".to\_sym becomes :a

you can always convert between different types using Ruby's "to" methods.

* **to\_s** converts values to **s**trings.
* **to\_i** converts values to **i**ntegers (numbers.)
* **to\_a** converts values to **a**rrays.

[12,47,35]­.max

=> 47

you turned the poem into a list using lines.to\_a. The lines component decided the way the string should be split up, and then one of our "to" methods, to\_a, converted those splits into an Array. (**to\_a**rray.)

books.values.each { |rate| ratings[rate] += 1 }

Each entry in the book array turns into a key within the ratings hash.

:symbol = this is a symbol. It is a type of variable.

A Ruby symbol is a thing that has both a number (integer) representation and a string representation.

**Blocks.** Chunks of code which can be tacked on to many of Ruby's methods. Here's the code you used to build a scorecard: books.values.each { |rate| ratings[rate] += 1 }.

Dir["/\*.txt"]

Searches for only text files in teh directory

print File.­read("/com­ics.txt")

Prints the content of the file.

FileUtils.­cp('/comic­s.txt', '/Hom­e/comics.t­xt')

Copies the file called comics.txt inside the Home directory

**File.open("/Home/comics.txt", "a") do |f|**

**f << "Cat and Girl: http://catandgirl.com/"**

end

Opens a file in write mode and adds the content at the end of the file.

**File.open("/Home/comics.txt", "a") do |f|**

**f << "Cat and Girl: http://catandgirl.com/"**

end

file method that gives you the last time this file was opened

def load\_­comics ( path )

.. comics = {}

.. File.forea­ch(path) do |line­|

.... name, url = line.­split(': ')

.... comics[nam­e] = url.s­trip

.... end

.. comics

.. end

Creates a method where we create an array with the text inside a file (separator here is “:”

“0” is always treated as true in Ruby.

unless name.length

print “username required”

end

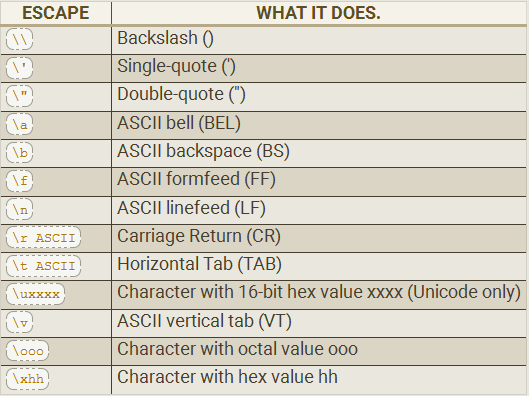
will always give true and the command inside the code will never print out, even if the username length is 0

You can bulk assign values like this:

formatter = "%{first} %{second} %{third} %{fourth}"

puts formatter % {first: 1, second: 2, third: 3, fourth: 4}

You will almost always use #{} to format your strings, but there are times when you want to apply the same format to multiple values. That's when %{} comes in handy.



The ARGV is the "argument variable," a very standard name in programming, that you will find used in many other languages. This variable holds the arguments you pass to your Ruby script when you run it.

$stdin

(standard input is a global variable. It is the input given to ruby via the console)

puts "Do you like me #{user\_name}? ", prompt

likes = $stdin.gets.chomp

*“You are the stdin to my life”* – Anonymous

Refactoring

Just a fancy way of saying rewriting the code to make it look better.

We can put the if or unless after the expression for more clarity if needed:

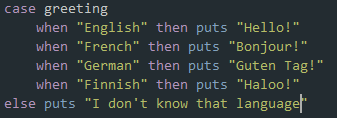
http://i.gyazo.com/d0c451b6539054949f9549ac18555595.png

We can go further and write an if… else in one single line, like so:

http://i.gyazo.com/875473ee8f19385c835b94977145b75d.png

http://i.gyazo.com/8fdaa0bcc834136a184a099d756cdc78.png

The switch function works like this:



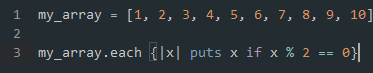
end

We first specify what variable we are gonna be checking (greeting). Then we present all the possible cases and say if this “then put this”.

**Conditional assignement:**

By writing myVar ||= “This value”, it will only become “This value” if it didn’t have any other value before (equal to nil). If it had a value before, nothing happens.

Now here is a simple algorithm to display only the pair values of an array:



This one prints all the letters from L up to a P:

http://i.gyazo.com/415d52637c7f8c9ee70bb483900c42f7.png

We can check whether an object will respond to a method (here, the .next method, which returns the next integer after this one.)

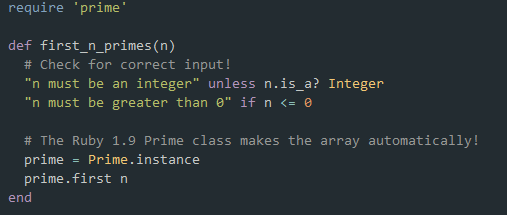
http://i.gyazo.com/f7d1743c5a4b6bd21001a6f6dac825e8.png

Using << is the same as using the .push method. It works for strings as well as arrays

http://i.gyazo.com/bf22a7711200e27233bd143833e8afec.png

http://i.gyazo.com/432392a582d456407bf5997e033b6f00.png

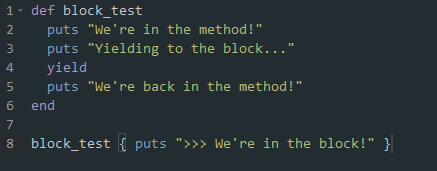
To sum up refactoration, here is a program that creates an array of the n first prime numbers. Look how simple it is compared to Javascript or PHP:

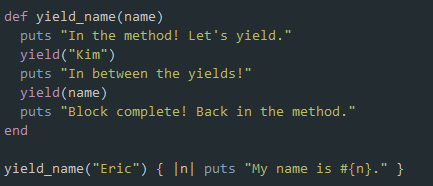


(require is used to call the prime method)

Methods

* **.select** followed by a block selects only the elements of the array that satisfy the condition inside the block
* **.collect** takes an array an applies the expression to each element in the array.
* **yield** is used when we want to process the code inside the brackets (called a block) inside a method at a given moment:





Difference between select, collect, map and each

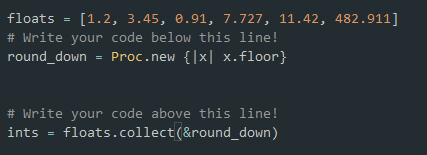
collect and map will apply the block to the array. If we use a conditional (n<2) in the block, it will return the array with true and false accordingly for each value ([true, true, false]).

Select will return only the values that satisfy the conditional inside the block.

each will return the values of the array untouched but will evaluate them anyway.

**Blocks**

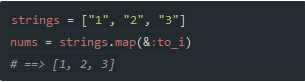
We can’t save blocks to a variable. We are gonna use procs instead. This is how to create on:



Not the proc is between parenthesis, and not between brackets.

If we want to call a proc outside of a method, we use the **.call** method.

We can convert every element of an array into something else by calling a proc:

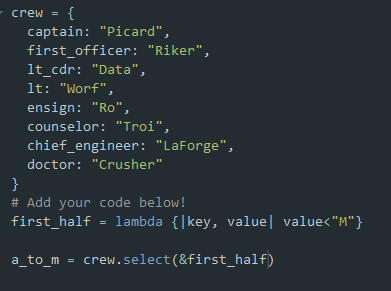


Another way is to create a lambda:

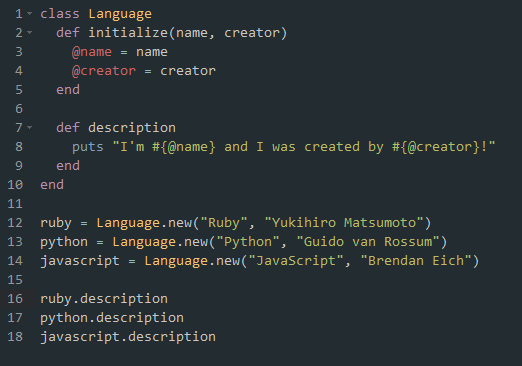
http://i.gyazo.com/9edb048d70a1346cc0ef20f4f7707ff3.png

It’s the same as creating a proc. The only difference is that a proc returns immediately, so the code afterwards is likely to be ignored, whereas the lambda gives back the control to the method as soon as it is finished.

The following lambda returns only the values of the hash that start with a letter that comes before M. Since it is a hash, the lambda must have two parameters |key, value|



Object oriented programming



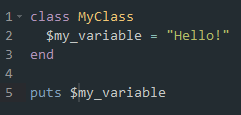
Attributes are declared differently depending on the scope we want to give them.

**@name (instance variable)** is accessible only by the instance of the class (normal, since each instance has its own name)

**@@files (class variable)** is class-wide.

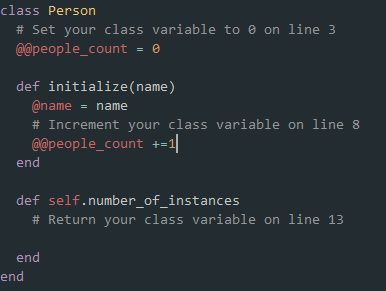
**$manufacturer (global variable)** is accessible globally, even outside of the class

Only the last one will be accessible if we don’t define an instance of the class. For instance:



They are not a very good idea, because they can be changed and tampered with from anywhere in your program

Class wide variables are cool, and can be used to do some neat tricks, like a counter that counts the number of instances we have created for a given class. Look at the @@people\_count in the following example:

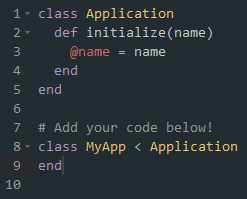


self.number\_of\_instances is a class-wide method, it can ba applied like this:

Person.number\_of\_instances

Inheritance:

Fox < Mammal



You can directly access the methods of a superclass (in case you have overwritten one in the subclass) with the super() keyword.

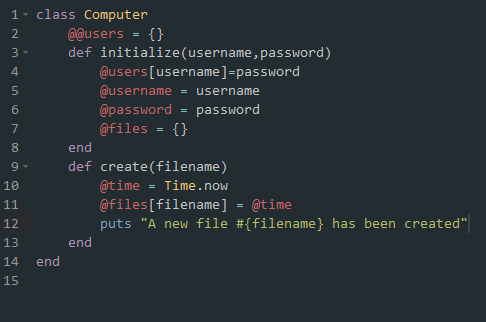
Any given Ruby class can only have one superclass.

What happens when you want to create a class out of the methods and attributes of several superclasses? You then use **mixins**

Here is a class message with a built in counter that counts sent messages:

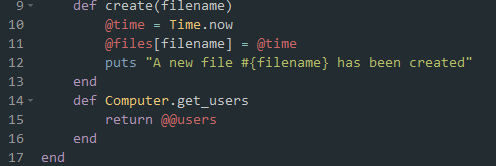


**Exercise: Computer class:**

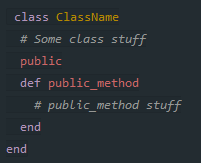


When the create method is called, it catches the time it was created with Time.now and stores it into the @time variable, then prints a message telling the file has been created.

We are then gonna create a class method to grab @@users and modify it. Instance variable-> instance method; class variable, -> class method! simple, right?



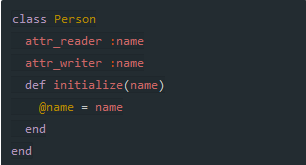
We can set the auth level of our methods inside of the classes to public or private (can only be called from inside the class. ClassName.method won’t work)



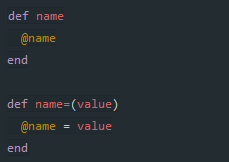
**This separates the private implementation from the public interface**

Private methods are useful when you wanna set up a method that attributes a unique id to each instance of the class. You want it to be done automatically and not be tampered by external manipulation.

attr\_reader and attr\_writer are ruby equivalents of PHP set and get methods:



generates:

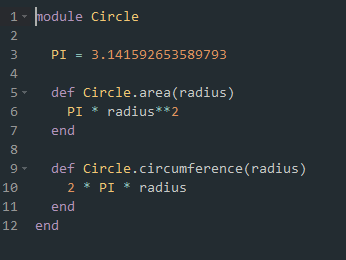


(remember, otherwise, in order to access instance variables from outside, you would have needed to create a method for each one of them just to create them (second picture))

attr\_accessor serves as both attr\_reader and attr\_writer

**Modules:**

A module is like a toolbox that contains a set of definitions:



That’s why you won’t include variables in a module: variables, by definition, change, and a module should be made up of constant definitions.

**scope resolution operator:** It’s a double colon (::). It basically tells Ruby where it should look for the definition of the constant we propose him (so that it distinguishes between John::FavoriteBook and Helena::FavoriteBook)

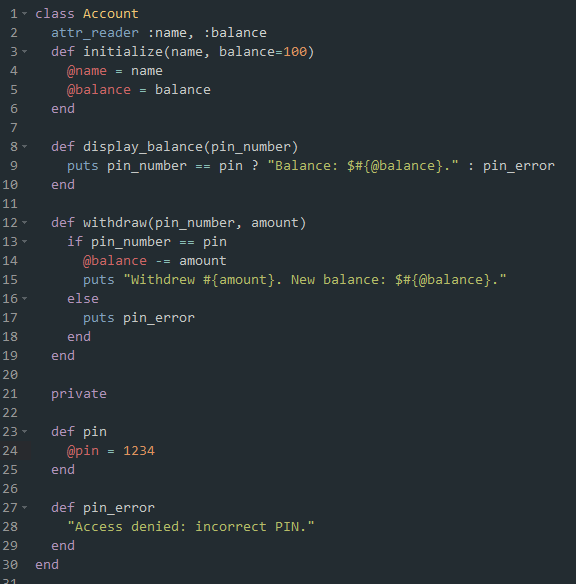
We make a module available by calling “require moduleName” or including it in a class. We then don’t have to use a scope resolution operator everytime we want to call a related method and we can just call the method anytime.

In the following case, our module is like a chunk of code outside of class that we are gonna include in both our classes. Practical! then we can call up the jump method like if we had defined it inside each one of our classes directly.



Whereas **includes** mixes the module at an instance level, meaning that instances can access it, there is another command, **extend**, which mixes the module at a class level, meaning that the class itself can use the method, as opposed to the instances.

**Bank app**



Note how we access balance: the @ means its an instance variable, the $ is just the dollar notation here. If we withdraw, the balance of the instance get decremented. You can only check tyour balance if you know your pin number. Unrelated news, balance in initialize is an optional attribute. If you don’t pass anything, it will take the default value 100

QIPKQOSKSMLTQTNLYQT#I#LKPQIJGK#j58u#7F77E5ED#P#vGJ4#GELLPHGKLKIIIPKKUJN#9EEE#E7DBGD0GB9ADHCDBMDE#GMJPIHMGHPP#YNWWPVWVQ#K#79EE#A7EGDD0HGDGFDDEEMCD#LELIKKHUHH#PRNPQT#PINTKONQ#HEJNMLNGPHHLHLIHULM#QIKLOQMNNYLQ#275#180#35#HFMKOLIIJQNLQJQHVKO#9EFF#08AAHDIE0CFGCFGDNEF#HNKQJINHIQQ#ZOZSSQSSXY#LJNRP#88FF#B8AAABFCIDHECHGDNDE#PFPLJIOVII#QTOZVRU#LJNPSLLM#HFMLPIHNOMNQQPQIVMO#PJMLNUPNMOSLMUULZMR#616#378#70#NGRLNPJJOLNNPONJWLN#0CGG#D9HEICCJGBFFHCIBOFG#IOLRKJOIJRR#aPaTTRTTYZ#MKOSQ#9CGG#C9HBBEBEABCIAIHIOEJ#KGKIOONWJJ#RYPXV#MNKRSTVS#JGKOONQPQMRLKPPKWNL#MKNUOTPRSSOSRaNT#891#567#63#KHLKLLNPJRSMSSKNXMM#A9HH#J0EECEFJIBKCBBCEPGG#JPMSLKPJKSS#bQUVZZTVW#QLPTR#9HHH#C0EKIJHKKIFFGHBGPFD#JHLQKMXKJ#T#WLVPPOTWP#JHOMNJQKMJLMONQLXKN#VLONVOWWWWWWRUbOQ#1056#792#28#LIOLSSNQNNOPLMTOYNL#BIII#DADKEKCJJJHJDFEKQHJ#KQNTMLQKLTT#VRTbVbTU#RMSTPS#ADII#EALHCEFLJELIJLEGQGD#PISKLYLQ#V#XMWSOUVUW#NIPSMMTLPMKKMMPMYPR#OMUVSTQWSSOVVQWQcQR#1375#855#210#24#56#96#108#160#176#240#312#112#120#640#Wuxh#Xvyi#Ywzj#Zx k#ay.l#bz,m#c 1n#d.2o#e,3p#f14q#g25r#h36s#i47t#j58u#k69v#l70w#m8Ax#n9By#o0Cz#pAD #qBE.#rCF,#Wuxh#Xvyi#Ywzj#Zx k#ay.l#bz,m#c 1n#d.2o#e,3p#f14q#g25r#h36s#i47t#j58u#k69v#l70w#m8Ax#n9By#o0Cz#pAD #qBE.#rCF,#sDG1#tEH2#uFI3#vGJ4#wHK5#xIL6#yJM7#zKN8# LO9#.MP0#Wuxh#Xvyi#Ywzj#Zx k#ay.l#bz,m#c 1n#d.2o#e,3p#f14q#g25r#h36s#i47t#j58u#k69v#l70w#m8Ax#n9By#o0Cz#pAD #qBE.#rCF,#sDG1#tEH2#uFI3#vGJ4#wHK5#xIL6#yJM7#zKN8# LO9#.MP0#Wuxh#ROQZQQSRSRSUYWTQeWZ#SIMNMMMKQOQRROPQYPS#g#qJUZbzFWFMbf#iflmoo#VSYZbb#GEGHPNIOUPM#P#E#R#Ywzj#Zx k#ay.l#bz,m#Oju.n#Pkv,o#Qlw1p#Rmx2q#Sny3r#Toz4s#Up 5t#Vq.6u#Wr,7v#Xs18w#Yt29x#Zu30y#av4Az#bw5B #cx6C.#dy7D,#ez8E1#f 9F2#Wuxh#Xvyi#Ywzj#Zx k#ay.l#bz,m#c 1n#d.2o#e,3p#f14q#g25r#h36s#Up 5t#Vq.6u#Wr,7v#Xs18w#Yt29x#Zu30y#av4Az#bw5B #cx6C.#dy7D,#ez8E1#f 9F2#g.0G3#h,AH4#i1BI5#j2CJ6#k3DK7#l4EL8#m5FM9#n6GN0#Idovh#Jepwi#Kfqxj#Lgryk#Mhszl#Nit m#Oju.n#Pkv,o#Qlw1p#Rmx2q#Sny3r#Toz4s#Up 5t#Vq.6u#Wr,7v#Xs18w#Yt29x#Zu30y#av4Az#bw5B #cx6C.#dy7D,#ez8E1#f 9F2#g.0G3#h,AH4#i1BI5#j2CJ6#28#R#Kfqxj#Lgryk#Mhszl#Nit m#Oju.n#Pkv,o#Qlw1p#Rmx2q#Sny3r#Toz4s#Up 5t#Vq.6u#Wr,7v#Xs18w#Yt29x#Zu30y#av4Az#bw5B #cx6C.#dy7D,#ez8E1#f 9F2#g.0G3#h,AH4#i1BI5#j2CJ6#k3DK7#l4EL8#m5FM9#n6GN0#Idovh#Jepwi#Kfqxj#Lgryk#Mhszl#Nit m#Oju.n#Pkv,o#Qlw1p#Rmx2q#Sny3r#Toz4s#Up 5t#Vq.6u#Wr,7v#Xs18w#Yt29x#Zu30y#av4Az#bw5B #cx6C.#dy7D,#ez8E1#f 9F2#g.0G3#h,AH4#i1BI5#j2CJ6#k3DK7#l4EL8#m5FM9#n6GN0#Idovh#Jepwi#Kfqxj#Lgryk#Idovh#Jepwi#Kfqxj#Lgryk#Mhszl#Nit m#Oju.n#Pkv,o#Qlw1p#Rmx2q#Sny3r#Toz4s#Up 5t#Vq.6u#Wr,7v#Xs18w#Yt29x#Zu30y#av4Az#bw5B #cx6C.#dy7D,#ez8E1#f 9F2#g.0G3#h,AH4#i1BI5#j2CJ6#k3DK7#l4EL8#m5FM9#n6GN0#Idovh#Jepwi#Kfqxj#Lgryk#Mhszl#Nit m#Oju.n#Pkv,o#Qlw1p#Rmx2q#Sny3r#Toz4s#Up 5t#Vq.6u#Wr,7v#Xs18w#Yt29x#Zu30y#av4Az#bw5B #cx6C.#dy7D,#ez8E1#f 9F2#g.0G3#h,AH4#i1BI5#j2CJ6#k3DK7#l4EL8#m5FM9#n6GN0#Idovh#Jepwi#Kfqxj#Lgryk#Mhszl#Nit m#Oju.n#Pkv,o#Qlw1p#Rmx2q#Sny3r#Toz4s#Up 5t#Vq.6u#Wr,7v#Xs18w#Yt29x#Zu30y#av4Az#bw5B #cx6C.#dy7D,#ez8E1#f 9F2#g.0G3#h,AH4#i1BI5#j2CJ6#k3DK7#l4EL8#m5FM9#n6GN0#Idovh#Jepwi#Kfqxj#Lgryk#Mhszl#Nit m#Oju.n#Pkv,o#Qlw1p#Rmx2q#Sny3r#Toz4s#Up 5t#Vq.6u#Wr,7v#Xs18w#Yt29x#Zu30y#av4Az#bw5B #cx6C.#dy7D,#ez8E1#f 9F2#g.0G3#h,AH4#i1BI5#j2CJ6#k3DK7#l4EL8#m5FM9#n6GN0#Idovh#Jepwi#Kfqxj#Lgryk#Mhszl#Nit m#Oju.n#Pkv,o#Qlw1p#Rmx2q#Sny3r#Toz4s#Up 5t#Vq.6u#Wr,7v#Xs18w#Yt29x#Zu30y#av4Az#bw5B #cx6C.#dy7D,#ez8E1#f 9F2#g.0G3#h,AH4#i1BI5#j2CJ6#k3DK7#l4EL8#m5FM9#n6GN0#Idovh#Jepwi#Kfqxj#Lgryk#Mhszl#Nit m#Oju.n#Pkv,o#Qlw1p#Rmx2q#Sny3r#Toz4s#Up 5t#Vq.6u#Wr,7v#Xs18w#Yt29x#Zu30y#av4Az#bw5B #cx6C.#dy7D,#ez8E1#f 9F2#g.0G3#h,AH4#i1BI5#j2CJ6#k3DK7#l4EL8#m5FM9#n6GN0#Idovh#Jepwi#Kfqxj#Lgryk#Mhszl#Nit m#Oju.n#Pkv,o#Qlw1p#Rmx2q#Sny3r#Toz4s#Up 5t#Vq.6u#Wr,7v#Xs18w#Yt29x#Zu30y#av4Az#bw5B #cx6C.#dy7D,#ez8E1#f 9F2#0CAHGHGCDFIII#80D8EBCAA0#Q#I#J#