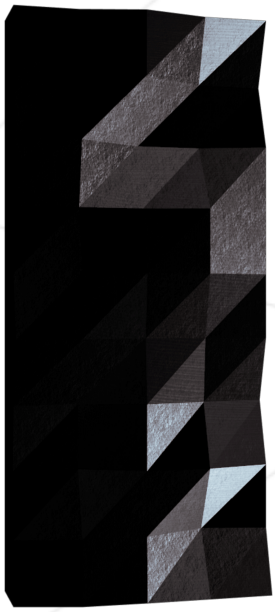

Introduction to Programming - 42

Day 03

Kai kai@42.us.org
Gaetan gaetan@42.us.org

Summary: This document is the subject of the day 03 of the introduction to programming piscine.

Contents

Chapter I

Guidelines

- Corrections will take place in the last hour of the day. Each person will correct another person accoring to the peer-corrections model.
- Questions? Ask the neighbor on your right. Next, ask the neighbor on your left.
- Read the examples carefully. The exercises might require things that are not specified in the subject...
- Your reference manual is called Google / "Read the Manual!" / the Internet / ...

Chapter II

Preamble

This is a real, functional program in the language "lolcode":

```
HAI 1.3

O HAI IM guess
    I HAS A animal

    HOW IZ I new YR animal
        I HAS A new ITZ LIEK guess
        new'Z animal R animal
        FOUND YR new
    IF U SAY SO

    HOW IZ I guessin
        VISIBLE "Is it a " MAH animal "? " !
        I HAS A answer, GIMMEH answer
        BOTH SAEM answer AN NOOB, O RLY?
        YA RLY
            DO NOT WANT finished
        MEBBE NOT answer
            ME IZ guessin, BTW again
        MEBBE BOTH SAEM answer AN "yes"
            VISIBLE "I win!"
            FOUND ME
        NO WAI
            ME IZ learnin, FOUND IT
        OIC
    IF U SAY SO

    HOW IZ I learnin
        VISIBLE ":)I give up. What is it? " !
        I HAS A answer, GIMMEH answer
        I HAS A new ITZ guess IZ new YR answer MKAY

        VISIBLE "Enter a question where the answer is YES for a " answer...
            " and NO for a " MAH animal ": " !
        I HAS A sentence, GIMMEH sentence
        VISIBLE "Thanks!"

        question IZ new sentence AN new AN ME
    IF U SAY SO
KTHX

O HAI IM question
    I HAS A question
    I HAS A yes
    I HAS A no

    HOW IZ I new YR ask AN YR yes AN YR no
        I HAS A new ITZ LIEK question
        new'Z question R ask
        new'Z yes R yes
        new'Z no R no
        FOUND YR new
```

```
    IF U SAY SO

    HOW IZ I guessin
        VISIBLE MAH question " " !
        I HAS A answer, GIMMEH answer
        BOTH SAEM answer AN NOOB, O RLY?
        YA RLY
            DO NOT WANT finished
        MEBBE NOT answer
            ME IZ guessin, BTW again
        MEBBE BOTH SAEM answer AN "yes"
            MAH yes IZ guessin, MAH yes R IT
        NO WAI
            MAH no IZ guessin, MAH no R IT
        OIC
        FOUND ME
    IF U SAY SO
KTHX

HOW IZ I playing YR animals
    VISIBLE "Think of an animal!"
    animals IZ guessin
IF U SAY SO

question IZ new "Does it have wings?"...
    AN guess IZ new "parrot" MKAY...
    AN guess IZ new "rabbit" MKAY
I HAS A animals ITZ IT

BTW baby exception
ME HAS A finished ITZ LIEK BUKKIT

ME IZ frist, O RLY?
YA RLY
    PLZ
        IM IN YR animals, BTW forever
            I IZ playing YR animals
            VISIBLE ""
        KTHX
    O NOES ITZ A finished
        VISIBLE ":):)Thanks for playing!"
    KTHX
OIC

KTHXBAI
```

You can run it yourself with the help of this gem!
(https://github.com/belkadan/lolcode-rb)

Chapter III

Exercise 00 : My First Method

	Exercise 00
Letś traverse the array	
Turn-in directory : $ex00/$	
Files to turn in : `my_first_method.rb`	
Allowed functions : `All`	
Notes : `n/a`	

- Create a script `my_first_method.rb` which includes a method. The method takes a string as an argument. The method must return an uppercase version of the string, if and only if the character string is longer than 10 characters. If the string is 10 characters or less, the method returns nil.

- Your program will call this method and display the return value on the command line. If there are no parameters, display `none` followed by a newline.

```
?> ./my_first_method.rb | cat -e
none$
?> ./my_first_method.rb "alo" | cat -e
nil$
?> ./my_first_method.rb "hello world" | cat -e
nil$
?> ./my_first_method.rb "i'M happY to be hERE" | cat -e
I'M HAPPY TO BE HERE$
?>
```

`Google ruby methods.`

Chapter IV

Exercise 01 : Greetings for All

	Exercise 01
Say hello to the lady	
Turn-in directory : *ex*01/	
Files to turn in : `greetings_for_all.rb`	
Allowed functions : `All`	
Notes : `n/a`	

- Create a script `greetings_for_all.rb` that contains a greetings method. The method takes a parameter name and displays a welcome message with that name. If the method is called without arguments, its default setting will be ñoble stranger". If the method is called with an arguent that is not a string, an error message should be displayed instead of the welcome message.

- So the following script:

```
?> cat greetings_for_all.rb
# your method definition here

greetings "lucie"
greetings
greetings 22
?>
```

  will have the result:

```
?> ./greetings_for_all.rb | cat -e
Hello, lucie.$
Hello, noble stranger.$
Error! That doesn't sound like a name.$
?>
```

`Google is_a.`

Chapter V

Exercise 02 : Help your Professor

	Exercise 02
A little boost to the prof	
Turn-in directory : $ex02/$	
Files to turn in : `help_your_professor.rb`	
Allowed functions : `All`	
Notes : `n/a`	

- Create a script `help_your_professor.rb` which contains a method average_mark. The method will use a hash, associating the first name of the students with their grade, to calculate the average score of the class on that test.

- So the following script:

```
?> cat help_your_professor.rb
# your method definition here$

class_csci101 = {
    "margot" => 17,
    "june" => 8,
    "colin" => 14,
    "lewis" => 9
}
class_csci102 = {
    "quentin" => 16,
    "julie" => 15,
    "mark" => 8,
    "stephanie" => 13
}
puts "Average mark for the CSCI 101 class: #{average_mark class_csci101}."
puts "Average mark for the CSCI 102 class: #{average_mark class_csci102}."
?>
```

has the result:

```
?> /.help_your_professor.rb | cat -e
Average mark for the CSCI 101 class: 12.$
Average mark for the CSCI 102 class: 13.$
```

`Google ruby hashes`

Chapter VI

Exercise 03 : Family Affairs

	Exercise 03
Family Stories	
Turn-in directory : *ex*03/	
Files to turn in : `family_affairs.rb`	
Allowed functions : `All`	
Notes : `n/a`	

- Create a script `family_affairs.rb`. It will contain a find_the_gingers method which takes in as a parameter a hash containing the first names of family members as key and their hair colors as attribute. This method will use the select metho to collect the first names of the redheads in a new array, which it will return.

- So a script like this:

```
?> ./family_affairs.rb | cat -e
# your method definition here

Dupont_family = {
    "matthew" => :red,
    "mary" => :blonde,
    "virginia" => :brown,
    "gaetan" => :red,
    "fred" => :red,
}

p find_the_gingers Dupont_family
?>
```

would have this result:

```
?> ./family_affairs.rb | cat -e
["matthew", "gaetan", "fred"]$
?>
```

`Google ruby hashes, select, to_s`

Chapter VII

Exercise 04 : Persons of Interest

	Exercise 04
People worth knowing.	
Turn-in directory : $ex04/$	
Files to turn in : `persons_of_interest.rb`	
Allowed functions : `All`	
Notes : `n/a`	

- Create a script `persons_of_interest.rb`. It will contain a method `great_births` that takes a hash representing people from history, each entry itself being a hash with keys "name" and "year_of_birth". Display them in order sorted by birth dates.

- A script like this:

```
?> cat persons_of_interest.rb
# your method definition here

women_in_science = {
    :ada => { :name => "Ada Lovelace", :year_of_birth => "1815" },
    :cecilia => { :name => "Cecila Payne", :year_of_birth => "1900" },
    :lise => { :name => "Lise Meitner", :year_of_birth => "1878" },
    :grace => { :name => "Grace Hopper", :year_of_birth => "1906" }
}

great_births women_in_science
```

  has output like this:

```
?> ./persons_of_interest.rb | cat -e
Ada Lovelace is a great person born in 1815.$
Lise Meitner is a great person born in 1878.$
Cecila Payne is a great person born in 1900.$
Grace Hopper is a great person born in 1906.$
?>
```

`Hashes and sort_by are "valu"able.`

Chapter VIII

Bonus: Posse

	Exercise 05
People worth knowing.	
Turn-in directory : $ex05/$	
Files to turn in : `family_list.rb`	
Allowed functions : `All`	
Notes : `n/a`	

Create a script which loops infinitely, taking input from the user. On each iteration, ask the user the name of someone in their family, and how they are related. Build a hash containing this information and print it when the user types DONE.

```
?> ./persons_of_interest.rb
Hello, what is someone's name?: Luna
And who is that person to you?: cousin
Hello, what is someone's name?: Marley
And who is that person to you?: dog
Hello, what is someone's name?: Anna
And who is that person to you?: sister
Hello, what is someone's name?: DONE
Cool, here is your family!
{"Luna"=>:cousin, "Marley"=>:dog, "Anna"=>:sister}
```