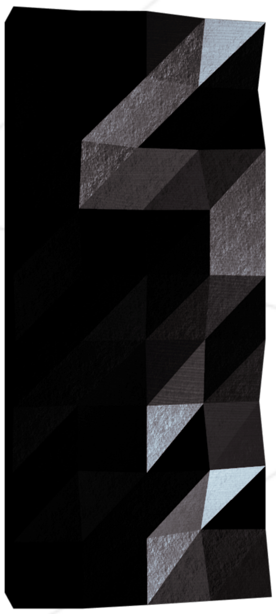

Information Libre

Scatter, learn things, and regroup

Kai kai@42.us.org

Summary: Keep a friend at your side, and study in the path of inspired ones who have given their knowledge out for free.

Contents

Chapter I

Before you Start!

- Download the `setup.sh` script included with this project. Then, in your terminal, run it by typing "sh ~/Downloads/setup.sh". If it gives you error messages, copy and paste the error message in the #setup_help Slack channel and ask your mentors.

- Find a partner to work with on this project. Be advised that Information_Libre may take up to a month, and you'll make faster progress if you pair with someone you can correspond with or meet with outside of 42 easily.

- If you are on our AP Computer Science Principles track, take this practice AP exam for a baseline score: `quiz.42.us.org` (Set aside 1 hour and answer as many as you can. The real exam is 2 hours.)

- Join some useful Slack groups! Click on "Channels" and join `#pdf_questions`, `#setup_help`, `#ruby-help`, `#python-help`, and one of the AP channels if interested.

- Create your project folder:

  1. From your project page on intra, copy the git repository link. Now, in the terminal type "git clone " and paste the link. After the link and before pressing enter, write a name for the new folder. Cloning your git repository always creates a new folder.

  2. cd into the folder you just created and from now on, save your work there. Use the command "mkdir <name>" to create new folders. Put each puzzle from this project in a folder with the same name.

Chapter II

Coding Tools

Set up your programming environment by picking a program for each of three essential functions. Our favorites are listed below.

If you want to install one which is not already on your computer, simply download the program and then drag its icon to your desktop or to a folder of your choice (you won't have access to the Applications folder).

1. Terminal
    - Built-in: Terminal
    - Upgrade: iTerm2
2. Text Editor
    - In the terminal: Emacs or Vim
    - Built-in: Xcode
    - Free: Atom
    - Free: Sublime
    - Free: Brackets
3. Web browser
    - Built-in: Safari
    - Free: Google Chrome
    - Free: Opera
    - Free: Vivaldi

Chapter III

Study One of These

With your partner, choose one from the list of freely offered, noncommercial Intro to Programming courses that follows.

You will use your chosen curriculum to learn about programming. Feel free to either follow it from start to finish or skip around. You need to put together the puzzle pieces from what those tutorials teach you to solve our programming challenges below.

III.1 Ruby

- Chris Pine's Learn to Program: https://pine.fm/LearnToProgram/

- Bastards' Book of Ruby, sections from "Style, Conventions, and Debugging" to "File input/output". http://ruby.bastardsbook.com/toc/ (Don't worry about the tweet fetching scripts, they no longer work with Twitter's new API.)

- Learn Ruby the Hard Way: https://learnrubythehardway.org/book/

- Test-first Ruby: http://testfirst.org/learn_ruby - best for someone with some programming experience, but really fun. The same author wrote http://codelikethis.com/lessons/learn_to_code which provides a beginner overview. Make sure you learn "puts" if you choose the test-first course.

III.2 Python

- IntroPython.org: http://introtopython.org/hello_world.html

- A Byte of Python: https://python.swaroopch.com/

- MIT Gentle Introduction to Programming using Python: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-189-a-gentle-introduction-lectures/

- Google's open Python class: https://developers.google.com/edu/python/ - best for people with a bit of programming experience already.

`In addition to studying these resources and Googling your questions, each challenge chapter contains a "Notes" section which links you directly to the official documentation of Ruby and Python. It's a good idea to get familiar with the layout of official documentation - that means you can find answers directly from the source.`

III.3 Gemstones? Invertebrates?

"Which language should I learn?"

If you do not know either Ruby or Python, you may not have anything to base your preference on.

Programmers love to argue about the superiority of their favorite tools but the truth is that both of these are popular, useful, modern programming languages that have similar abilities. Python is used more widely and Ruby is a bit more "poetic" or aesthetically pleasing.

You can research opinions online if you would like to read something about their characteristics.

If you have no prejudice, I arbitrarily recommend that you go with Ruby =)

It's what Intra is written in and thus it is what we love.

III.4 Going Retro

Ruby and Python are updated on a regular basis with new features.

So, for example, hashtables are written using different symbols in Ruby 1.9.3 than they are in the modern version, Ruby 2.4.2.

Python 3 is a big leap forward from the features included in Python 2. However, it's not supported by all of the tools that interface with Python yet.

`Setup.sh` sets your computer to use Ruby 2.4.2 and Python 2.7 by default. It also installs the tools needed to easily switch between versions of both languages (so please run it regardless of which version you want)!

Some of the freely offered coding classes are written for a different version of the programming language than Ruby 2.4 & Python 2.7. If that's the case, post in the `#setup_help` Slack channel for help using `rbenv` and `pyenv` to switch the version used by your computer.

Chapter IV

Turn In

Turn in your work with the following structure. We have divided the essential topics into seven parts, so you will turn in seven folders. Name them exactly as the example shows.

```
?> pwd
~/kai/hackhighschool
?> ls
first_day
information_libre
?> cd information_libre && ls -a
.
..
.git
4200_io
4201_if
4202_str
4203_num
4204_arr
4205_bool
4206_hash
?> cd 4200_io && ls
4200_io.rb
printing_practice.rb
argv_in_python.py
```

In each folder, turn in the assigned project in a .rb or .py file with a name that matches the folder name. Also turn in at least two other examples of short scripts that use or demonstrate that chapter's coding concept.

You will get bonus points for turning in for than two additional demonstrations of the topic and explaining them to your peer graders.

Each day, turn in your work so far by typing three commands in order:

- git add *
- git commit -m "<your comments here>"
- git push
- If you have an error during the git push, you may need to refresh your authentication ticket. Do this by typing "kinit <username>" and then typing your intra password.

Chapter V

Format your Code

Each 42 challenge you turn in must adhere to the following format:

```
#!/usr/bin/env ruby

# This is what my program does
# By <userid>

def function_a
 #code
end

def function_b
 #code
end

def main(ARGV)
 #main method
 function_a
 function_b
end

main(ARGV)
```

- Always begin with the "#!/usr/bin/env ruby" statement. This tells your terminal to run the program using Ruby. In python, the first line is "#!/usr/bin/env python".
- Always add a comment stating what this program is for, some hints to help others use or understand it, and your name or intra ID.
- Do not write any code outside of functions except for one line, at the end of your program, which calls the main() function.
- The (ARGV) parameter is not always needed. In Python it is sys.argv.

Reference your chosen intro to coding class to learn about functions/methods (The keyword "def" means "define function...").

Chapter VI

Exercise 0: Input, Printing, and ARGV/sys.argv

	Exercise 00
IO: Input/Output, Command Line Arguments, String Interpolation	
Turn-in directory : `4200_io`	
Files to turn in : `4200_io.rb or 4200_io.py, plus 2 or more practice files`	
Notes : `Ruby IO, ARGF Python Built-in Functions, System`	

Create a short Mad-Libs puzzle. It will take a command-line argument which sets the title of the madlib. Then, your program prompts your corrector for an adjective, a business, an animal, and a noise. Then, print a version of "Old MacDonald Had a Farm" using the user input instead of some of the traditional words.

```
?> ruby 4200_io.rb "Ode to Joy"
Please input an adjective:
```

After printing each prompt, the user (yourself or your friend) should type in a word and press 'enter'. When all the prompts are answered, your program prints out an empty line, then the title (from argv) in all caps, and then each line of the poem with the words replaced as below:

```
?> ruby 4200_io.rb "Ode to Joy"
Please input an adjective: fruity
Please input a business: orchard
Please input an animal: bat
Please input a noise: click

ODE TO JOY
fruity Macdonald had a orchard, E-I-E-I-O
and on that orchard he had a bat, E-I-E-I-O
with a click click here
and a click click there,
here a click, there a click,
everywhere a click click,
fruity Macdonald had a orchard, E-I-E-I-O!
```

Chapter VII

Exercise 1: Decision Making

	Exercise 01
IF: If, If-else, Else, and comparison symbols like ==, =>, >, !=	
Turn-in directory : `4201_if`	
Files to turn in : `4201_if.rb or 4201_if.py, plus 2 or more practice files`	
Notes : `Ruby` `Control Expressions` `Python` `Control Flow`	

Create a program which contains a secret five-letter word. When the program runs, it tells the user what letter the word starts with and invites the user to guess.

Use a loop to receive guesses from the user 10 times.

The program says different things depending on the input:

- If the user does not type a word but presses Enter, the program informs them, "You wasted a guess =P"
- If the user types a word that is longer or shorter than five letters long, the program helpfully prints out "0, 1, 2, 3, 4 that's how we count to five!".
- If the user types a word that is five letters long but does not start with the correct letter, the program helpfully prints out the full alphabet.
- If the user types a word that is five letters long and starts with the correct letter but is not the secret word, the program informs them how many guesses they have left.
- If the user guesses the word correctly, the program prints out, "Good Job! You are one with the Source."
- If the user spends all 10 guesses and does not get the question right, exit the program.

```
?> ruby 4201-if.rb
The secret word begins with a D.
GUESS: delighted
0, 1, 2, 3, 4 that's how we count to five!
GUESS: rigor
ABCDEFGHIJKLMNOPQRSTUVWXYZ
GUESS:
You wasted a guess!
GUESS: dolma
You have 6 guesses left!
GUESS: dough
Good Job! You are one with the Source.
```

Chapter VIII

Exercise 2: Strings

	Exercise 02
STR: Strings, Regular Expressions, Strings are an array of Characters	
Turn-in directory : `4202_str`	
Files to turn in : `4202_str.rb or 4202_str.py, plus 2 or more practice files`	
Notes : `Ruby String, Regular Expressions Python String, Regular Expressions`	

Write a progam which performs three tasks in a row on the phrase given as a command line argument.

First, it prints out the phrase with eVeRy OtHeR lEtTeR cApItAlIzEd.
Then, print out the same string with every capitalized vowel replaced by an asterisk.
Next, run a check_parentheses function which determines whether every open paren "(" in the phrase is balanced with a close paren ")". Print out "Balanced? True" or "Balanced? False" accordingly.

```
?> ruby 4202_str.rb "(whats going on()?)"
(WhAtS gOiNg On()?)
(Wh*tS g*iNg *n()?)
Balanced? True
```

```
?> ruby 4202_str.rb "()()()()(((a)b)b)b"
()()()()(((A)b)B)b
()()()()(((*)b)B)b
Balanced? True
```

```
?> ruby 4202_str.rb "((this doesn't match)"
((ThIs DoEsN't MaTcH)
((Th*s Do*sN't MaTcH)
Balanced? False
```

`Use gsub or re.sub. Study a little about regex. Look up what string interpolation means & use it all the time.`

Chapter IX

Exercise 3: Numbers and Casting

	Exercise 03
NUM: Integers, Floats, Doubles, other data types and how to change types	
Turn-in directory : `4203_num`	
Files to turn in : `4203_num.rb or 4203_num.py, plus 2 or more practice files`	
Notes : `Ruby Numeric Python Numeric Types`	

Write a program that takes two numbers as command line arguments.

They will come into your program as Strings. Find a way to turn them into numbers with which you can perform math.

Divide the first number by the second one and print out both the integer quotient and the remainder.

Then, your program should declare and initialize four variables of different numeric types. Print them out and use the built-in functions type() or .class to print the variable type of each.

```
?> ruby 4203_num.rb 142 6
142 divided by 6 equals 23 remainder 4
Variable a contains : 10  which is of type: Integer
Variable b contains: 56.99  which is of type: Float
Variable c contains: 2+3i  which is of type: Complex
...
```

Chapter X

Exercise 4: Arrays/Lists

	Exercise 04
ARR: Arrays in Ruby; Lists (and Tuples, Sets, etc) in Python	
Turn-in directory : `4204_arr`	
Files to turn in : `4204_arr.rb or 4204_arr.py, plus 2 or more practice files`	
Notes : `Ruby Array, Python List, More on Lists`	

Take in a set of numbers as command line arguments. Store them as an array and print out the min, max, mean, median, mode and range of the set.

```
?> ruby 4204_arr.rb 142 6 13 36 54 4 9 78 78 102
Min: 4
Max: 142
Mean: 52.2
Median: 45
Mode: 78
Range: 138
```

Chapter XI

Exercise 5: True, False, and nil/None

	Exercise 05
BOOL: True and False keywords, nil/None, "and" and "or" expressions	
Turn-in directory : `4205_bool`	
Files to turn in : `4205_bool.rb or 4205_bool.py, plus 2 or more practice files`	
Notes : `Ruby nilClass, TrueClass, FalseClass Python Constants, Types`	

Using the three provided arrays:

Ruby
[false,true,true,nil,true,nil,nil,false,false,nil,true,false]
[or,or,or,==,!=,==,and,==,!=,and,!=,and]
[false,false,nil,nil,true,true,false,true,nil,false,true,nil]

Python
[False,True,True,None,True,None,None,False,False,None,True,False]
[or,or,or,==,!=,==,and,==,!=,and,!=,and]
[False,False,None,None,True,True,False,True,None,False,True,None]

Write a program which tests boolean logic by combining one element from each array into an equation and printing the full equation with its result. You can either use the arrays as given or have your program randomize them.

```
?> ruby 4205_bool.rb
false or false => false
true or false => true
true or false => true
nil == nil => true
true != true => false
...
```

There are several ways to do this. Sophisticated ways include .send or getattr(). Simpler ways involve using a series of if statements to perform the operation requested. Practicing these equations in the terminal by typing "irb" or "python" may be fun.

Chapter XII

Exercise 6: Hashes/Dicts

	Exercise 06
HASH: Hashes in Ruby, Dictionaries in Python	
Turn-in directory : `4206_hash`	
Files to turn in : `4206_hash.rb or 4206_hash.py, plus 2 or more practice files`	
Notes : `Ruby Hash Python Dictionary`	

Using the attached file names.txt, store the information in a hash or dictionary where first names are associated with last names.

Use your hashtable to identify which first names are shared by more than one student, mentor or admin in h2s. Print out each first name that repeats in the set followed by an array of the last names associated with that first name.

```
?> ruby 4206_hash.rb
# There are many ways to take the input in... you could use gets() and copy/paste the whole thing in
 the terminal here
Elliot (2): [Tregoning, vanHeuman]
```

* Note: This is not a real example. Eliot vanHeuman spells his name with one 'l' and Elliot Tregoning spells it with two. ;-)