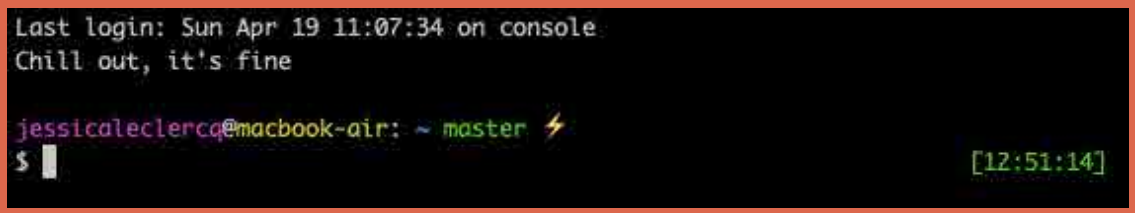
Lexique

The Hacking Project



Sommaire

GIT / GITHUB

Upload repository (local > online)

$ Git init
$ Git remote add origin (url_repo)
$ Git push origin master

Upload repository (online > local)

$ Git init
$ Git remote add origin (url_repo)

$ Git pull origin master

Other random commands

$ Git stash
> (sauvegarde provisoire) + git check-out (pour se balader d'un fichier à l'autre)
$ Git stash pop
> pour récupérer travail

Other random commands

$ Git clone
> cloner repo online en local
$ Git add_mon_fichier
> ajouter un fichier
$ Git commit -m "(...)"
> ajout d'un commentaire
$ Git status
> état des commits
$ Git log
> historique des commits

$ Git checkout SHA
> retour en arrière purement indicatif
$ Git checkout master
> revenir à version actuelle
$ Git reset hard - SHA
> revenir arrière définitivement

$ Git diff
> affiche toutes modifications de tous fichiers entre commits (voir git diff nom_fichier pour git diff limité à ce fichier)

Les Méthodes

Basic knowledge

une entrée

```
def ma_méthode (local_variable)
        #code qui faire qqch lorsque méthode
        sera appelée
end
ma_méthode(une_variable_choisie)
```

Different types of methods

> Les boucles (simples)

```
n.times do
leap_year = 2016
puts "Liste 3 prochaines années bissextiles :"
3.times do
        leap_year = leap_year + 4
        puts leap_year
end
```

```
for count in (1..6)
leap_year = 2016
puts ""Liste 3 prochaines années bissextiles :"
3.times do
        leap_year = leap_year + 4
        puts leap_year
end
```

> Les boucles while

Les boucles while ne s'arrêtent que en fonction condition (boléen qui doit passer de true à false)

Exemple

```
# Count until 5
i = 1
while i <= 5
        puts i
        i += 1
end
```

Arrays

> Les boucles et arrays

Exemple

```
prime_numberes = [2, 3, 5, 7, 11, 13]
result = 0 #variable qui va contenir le résultat

prime_numbers.each do |number| #initialisation de la boucle each
        result = result + number # ajoute à result valeur chaque entrée array - entrée identifiée par variable  |number|
```

Les Méthodes

Exercices sur des boucles

> roll the dice

```
def roll_dice
        rand(1..6)
end

def analyze_dice(roll_dice)
        if roll_dice >=5
                puts "Vous avancez!"
        1
        elsif roll_dice == 1
                puts "Vous reculez!"
        -1
        else
                puts "rien ne se passe"
        0
        end
end

def show_state(analyze_dice)

        puts "Vous êtes sur marche n° #{analyze_
dice}"
end
```

```
def is_over?(analyze_dice)
        if analyze_dice == 10
                true
        else
                false
        end
end

def play
        puts "Bienvenue dans le jeu!!"
                step = 1
        show_state(step)

        while(!is_over?(step)) do
                puts "tapez 'entrée' pour jouer"
                gets.chomp

        step += analyze_dice(roll_dice)
        show_state(step)
        end

        puts "===Vous avez gagné!==="
end

play
```

Les Méthodes

Exercices sur des boucles

> Crypotcurrencies

```
crypto_name = ["Bitcoin","etc."]
crypto_value = ["$6558,07","etc.]

crypto_value = crypto_value.map {|i| i.delete("$").
to_f}

#transform to hash
crypto_currency = crypto_name.zip(crypto_val-
ue).to_h

puts crypto_currency
puts "La valeur la plus haute est #{crypto_curren-
cy.key(crypto_currency.values.max)}"

#nombre de qqch dans le hash
currencies_6000 = [ ]
        crypto_value.each do |k, v|
        if v <= 6000
        currencies_6000 << k
        end
end

puts "Il y a #{currencies_6000.size} cryptomon-
naies dont le cours est en-dessous de 6000"

#La plus haute devise sous 6000
currency_price_6000      =crypto_currency.select
{|k,v| v <= 6000}
max_currency_6000  =  crypto_currency_6000.
max_by{|k,v| v}

puts "La cryptomonnaie la plus haute sous la barre
des 6000 est #{max_currency_6000}"
```

> Journalist array

```
def number_journalists_with_in_it (journalist)
        number_of_numbers = 0 #incrémenteur
> commencer un truc par 0
        journalists.each do  |journalists_handle|
        if journalist_handle.count ("0-9") > 0
        number_of_numbers += 1
        end
end

def underscore_counter (array)
number_of_underscore = 0
        array.each do |element|
number_of_underscore += element.count ("_")
      number_of_underscore = number_of_score
        end
end
```

Interesting classes

Cool classes, frequently used

> Also, always check ruby-doc

index = position dans le array ou string (-1,0,1,2…)

.select = returns new array

.split = chacun des éléments du arrays sont divisés

.map= retourne un array dont chaque entrée est issue des entrées du array initial mais modifiées selon l'instruction de la boucle > map! modifie array initial

.chars = returns array to string

** = Raises int to the power of numeric, which may be negative or fractional. The result may be an Integer, a Float, a Rational, or a complex number.

.reject = Returns a new array containing the items in self for which the given block is not true. The ordering of non-rejected elements is maintained.

.is_a? Returns true if class is the class of ob

Testing

Rspec

> Synthèse

- *require_relative '../lib/hello' dit juste à notre programme "hey, je vais vouloir accéder au contenu du fichier hello.rb qui est dans le dossier lib qui se trouve dans le dossier parent (d'où le ../) de là où tu es".*

- *describe "the hello function" do (...) end permet d'ouvrir un premier groupe de tests que l'on a appelé "the hello function". Comme son nom l'indique, ce groupe de tests automatiques va se focaliser sur ce que doit faire la fonction "hello".*

- *it "says hello" do (...) end permet d'ouvrir un premier test au sein du groupe de test "the hello function". Nous avons intitulé ce test "says hello" car c'est précisément ce comportement qu'on va tester sur la fonction hello.*

- *expect(hello).to eq("Hello world!") est le cœur même du test ! On a indiqué via expect(..) qu'on va exécuter la fonction hello. Puis le .to eq(..) permet de spécifier le résultat attendu : un string "Hello world!"*

> Exemples

```
require_relative '../lib/01_crypto_scrapper'

describe "the crypto scrapper function" do
        it "should be a hash" do
        expect(crypto_scrapper).to        be_in-
stance_of(Hash)
        end

        it "should not be nil" do
        expect(crypto_scrapper).not_to be_nil
        end

        it "should not be empty" do
        expect(crypto_scrapper).not_to      be_
empty
end
        end
```

```
describe "#factorial" do
        it "computes the factorial of 0" do
        expect(factorial(0)).to eq(1)
        end
end
```

Exemples, exemples

Multiples 3 et 5

> Méthode

```
def is_multiple_of_3_or_5?(num)
   (num % 3 == 0) || (num % 5 == 0)? true : false
end

def sum_of_3_or_5_multiples(final_number)
  if final_number.is_a?(Integer) && final_number
>= 0
    sum = 0
    final_number = final_number - 1
    while final_number >= 0 do
      if is_multiple_of_3_or_5?(final_number)
        sum = sum + final_number
        final_number = final_number - 1
      else
        final_number = final_number - 1
      end
    end
    return sum
  else
    return "Prend qu'entier naturels"
  end
end
```

> Testing

```
require_relative '../lib/multiples'

describe "the is_multiple_of_3_or_5? method"
do
 it "should return TRUE when an integer is a mul-
tiple of 3 or 5" do
   expect(is_multiple_of_3_or_5?(3)).to eq(true)
 end

 it "should return FALSE when an integer is NOT
a multiple of 3 or 5" do
   expect(is_multiple_of_3_or_5?(2)).to eq(false)
     end
end

describe "the sum_of_3_or_5_multiples meth-
od" do
  it "should return the sum of the integers (even
0)" do
         expect(sum_of_3_or_5_multiples(10)).to
eq(23)
end
```

Bot Twitter (with .env)

Find online API

> Personal and public

- Enter APIs in .env file + add .env to your .gitignore file
- Find functions on public documentation (example Spotify, Twitter, etc.)

>What your .rb contain

```
require 'dotenv'
require 'twitter'

Dotenv.load('../.env')

puts ENV['TWITTER_API_SECRET']

def login_twitter
  Twitter::REST::Client.new do |config|
       config.consumer_key  =  ENV["TWITTER_
CONSUMER_KEY"]
    config.consumer_secret = ENV["TWITTER_
CONSUMER_SECRET"]
    config.access_token = ENV["TWITTER_AC-
CESS_TOKEN"]
      config.access_token_secret  =  ENV["TWIT-
TER_ACCESS_TOKEN_SECRET"]
  end
end
```