Projets : Une série d'exercices en Ruby

1. Introduction

Bienvenue dans cette nouvelle journée d'exercices. Cette journée te fera creuser les méninges et ne sera pas facile. Heureusement, avec la force de la communauté, elle sera parfaitement à ta hauteur. On y va ?

2. Les exercices

Voici ce qui va se passer pour cet exercice : pour chaque sous-partie, nous allons te demander de créer un programme, et de soit répondre à des questions, soit de le faire marcher.

Nous te conseillons de tout mettre dans un joli repo Git, afin que tu t'entraines avec le programme de versionning.

2.1. Bonjour monde

Créé un programme exo\_01.rb qui affiche "Bonjour, monde !". Voici les lignes qu'il doit avoir d'affichées lorsque tu l'exécutes :

$ ruby exo\_01.rb

Bonjour, monde !

2.2. Un programme qui dit bonjour

Écris un programme exo\_02.rb qui demande le prénom de l'utilisateur, et qui salue l'utilisateur avec "Bonjour, *prénom* !"

On l’a vu hier.

2.3. Un programme qui calcule des âges

Écris un programme exo\_03.rb qui demande son année de naissance à l'utilisateur, puis qui ressort l'âge que l'utilisateur a eu en 2017.

On l’a vu hier.

2.4. Un programme centenaire

Écris un programme exo\_04.rb qui demande son année de naissance à l'utilisateur, puis qui ressort l'année où l'utilisateur aura 100 ans.

On l’a vu hier. Des similarités avec (2.3) mais pas le même calcul.

Il va falloir rajouter une condition je pense par rapport à l’exercice 3.

2.5. Un programme qui répète

Écris un programme exo\_05.rb qui demande un nombre à l'utilisateur, puis qui écrit autant de fois "Salut, ça farte ?

On l’a vu hier.

2.6. Un programme qui répète (bis)

Écris un programme exo\_06.rb qui demande un nombre à un utilisateur, puis qui écrit autant de fois -1 "Bonjour toi !". Ainsi, si l'utilisateur rentre 10, le programme devra écrire 9 fois "Bonjour toi !"

On l’a pas vu hier mais cela a des liens avec le (2.5).

2.7. Compter

Écris un programme exo\_07.rb qui demande un nombre à l'utilisateur, puis qui compte jusqu'à ce nombre.

Cfr exercice 12 hier.

2.8. Compte à rebours

Écris un programme exo\_08.rb qui demande un nombre à l'utilisateur, puis qui affiche un compte à rebours à partir de ce nombre, jusqu'à 0.

C’est le même que (2.7) sauf qu’au lieu de compter de 0, on compte de ce nombre indiqué et on fait le sens inverse.

2.9. Afficher les années

Écris un programme exo\_09.rb qui demande son année de naissance à l'utilisateur, puis qui va ressortir chaque année depuis son année de naissance jusqu'aujourd'hui.

Des liens avec (2.8) et probablement un travail sur les boucles (loops). Ça se répete jusqu’à l’infini, donc il faut bien mettre un début à une fin.

J’arrive à faire le « décompte » mais je n’arrive pas à définir le champ de ma variable et la limite de celle-ci.

Vidéo du gars : <https://www.youtube.com/watch?v=gtGujoBnEsI>

Documentation de ruby : <http://ruby-doc.com/docs/ProgrammingRuby/>

2.10. Afficher tous les âges

Écris un programme exo\_10.rb qui demande son année de naissance à l'utilisateur et qui va **afficher chaque année depuis son année de naissance jusqu'aujourd'hui**. Pour chaque année affichée, le programme devra annoncer l'âge que l'utilisateur avait cette année-là.

C’est comme 2.9, sauf que pour chaque années affichée, il faut rajouter un âge.

2.11. Virer les années

Le programme exo\_10.rb est cool, mais on peut l'améliorer. Écris un programme exo\_11.rb qui va demander son âge à l'utilisateur, et qui, pour chaque année depuis sa naissance, dira "Il y a X ans, tu avais Y ans".

Lien avec l’exercice 10 donc une boucle. Un string avec deux variables de nombres à l’intérieur.

2.12. Annoncer l'âge, option BG

Notre programme exo\_11.rb est devenu beau gosse. Écris un programme exo\_12.rbqui va faire la même chose, sauf que si X et Y sont égaux, il dira "Il y a n ans, tu avais la moitié de l'âge que tu as aujourd'hui".

Il y’a une fonction conditionnelle (contrôle de flux).

2.13. Une liste d'email

Écris un programme exo\_13.rb qui va créer une liste de 50 faux emails et les stocker dans une array (c’est une sorte de tableau, base de données dans ruby). Voici le format que devront avoir les faux emails :

"jean.dupont.01@email.fr"

"jean.dupont.02@email.fr"

etc..

Une boucle, à voir après, … .

2.14. Afficher les bons emails

Prends le programme exo\_13.rb et créé un programme exo\_14.rb qui va reprendre l'array des emails créés, et n'afficher que les emails avec un nombre pair.

"jean.dupont.02@email.fr"

"jean.dupont.04@email.fr"

etc..

Voir le 13 et probablement rajouter une condition en plus.

2.15. La pyramide

Construis un programme exo\_15.rb qui va demander à l'utilisateur un nombre entre 1 et 25 et qui va sortir une pyramide à descendre d'autant d'étages que ce nombre. Voici un exemple :

Salut, bienvenue dans ma super pyramide ! Combien d'étages veux-tu ?

> 5

Voici la pyramide :

#

##

###

####

#####

Histoire d’une boucle avec une limite.

2.16. La pyramide, dans l'autre sens

Reprends ton programme exo\_15.rb et fais un programme pyramide.rb qui montera au lieu de descendre :

Salut, bienvenue dans ma super pyramide ! Combien d'étages veux-tu ?

> 5

Voici la pyramide :

#

##

###

####

#####

Bien que légèrement différent dans l'énoncé, ce programme est bien plus dur que le exo\_15.rb, donc c'est normal de devoir réfléchir à comment le faire 😎

Histoire d’une boucle avec une limite (inversée). Est-ce qu’il faut mettre un nombre entier négatif ?

2.17. La pyramide, version expert

Crée un programme exo\_17.rb qui va demander à l'utilisateur un nombre entre 1 et 25 et qui va sortir une pyramide qui monte et qui descend, comme montré ci-dessous :

Salut, bienvenue dans ma super pyramide ! Combien d'étages veux-tu ?

> 5

Voici la pyramide :

#

###

#####

#######

#########

3. Rendu attendu

Le rendu attendu est un repository GitHub qui contient tous les exercices de cette journée.