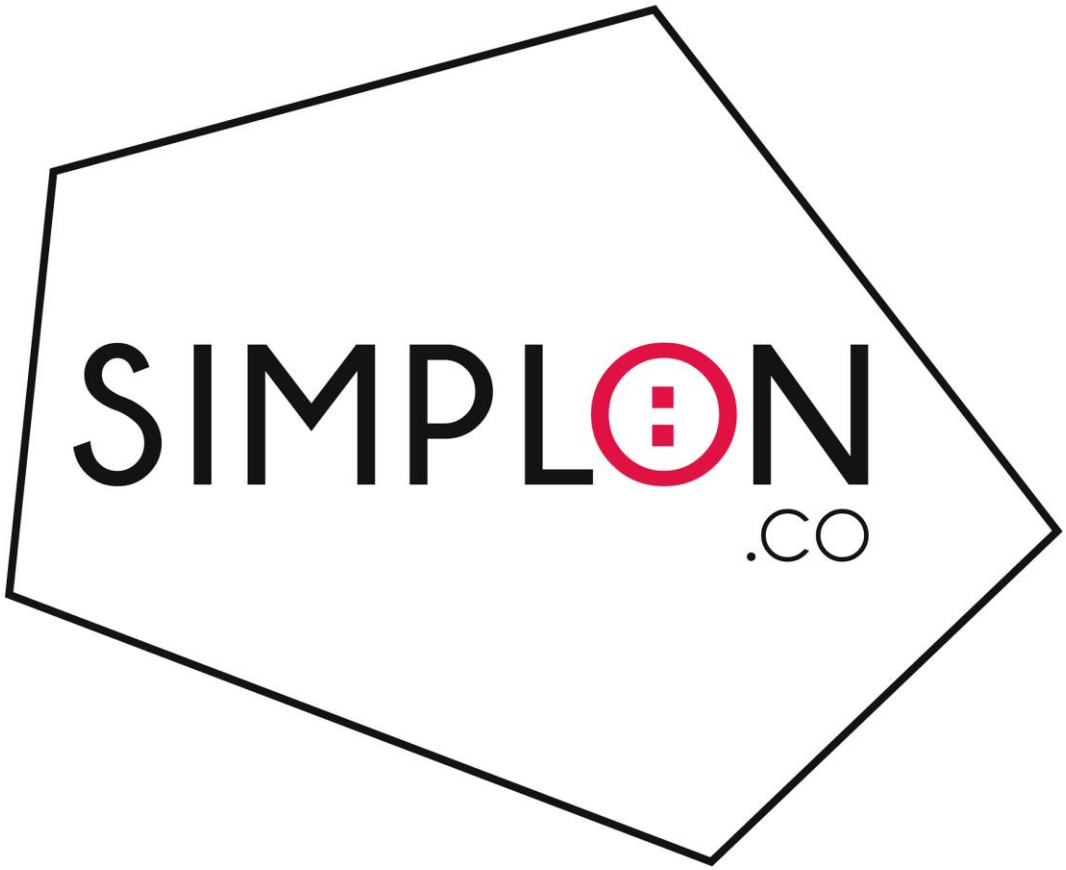

Partie III - Initiation à la programmation (Ruby)

Partie 0 - Programme

Nous vous proposerons chaque jour de la lecture, des activités/exercices et des ressources. Tout est décrit dans les parties correspondantes.

Cette première partie décrit ce que nous attendons de vous aujourd'hui.

1 - Soyez curieux !

Cf. partie "Lecture".

2 - Faites les activités dans l'ordre et mettez en pratique

Et n'hésitez à aller plus loin **si et seulement s**i vous en avez le temps et que vous avancez bien.

Vous allez acquérir des connaissances au fil de votre avancement dans les activités. Utilisez-les pour résoudre un maximum de problèmes parmi les séries d'exercices que nous vous proposons.

3 - Elargissez vos réseaux sociaux et soyez curieux (bis) !

Quelques Twittos en relation avec les sujets du jour : @RubyInside, @HCN_Ruby, @chadfowler, @peterc, @dhh

N'oubliez pas de partager vos découvertes avec les autres via Slack ;-)

Partie 1 - Lecture

Comprendre en quoi consiste la programmation  **3 min vidéo**

Courte vidéo d'introduction à un cours en ligne. Toujours utile. Hélas, le reste des vidéos sont chères.

> Lien

Partie 2 - Activités

Ruby  **1-2 h** **tutoriel**

Ruby est un outil puissant, et pourtant il est facile de commencer à l'apprendre ; il est utilisé pour le développement des sites, partout dans le monde.

Ne faites que les 5 premières parties du cours ! Vous irez jusqu'à la partie "Hashes et symboles".

Entre chaque partie du cours, vous réaliserez un maximum d'exercices des séries d'exercices concernées (voir partie 3).

> Lien

Bonus: Ruby Warrior  **???** **jeu**

Pour vous challenger si vous vous ennuyer… Attention la difficulté augmente très vite, essayez d'aller au moins au niveau 5 ou 6.

N'oubliez pas de finir tous vos exercices… ;-)

> Lien

Partie 3 - Exercices

A faire après la partie "Introduction à Ruby"

Exercice 1 - Surface d'un triangle

Écrire un algorithme qui calcule et affiche la surface d'un triangle connaissant sa base et sa hauteur.

Exercice 2 - Prix TTC

Écrire un algorithme qui, étant donné le prix unitaire d'un produit (hors TVA), le taux de TVA (en %) et la quantité de produit vendue à un client, calcule et affiche le prix total à payer par ce client.

Exercice 3 - Permutation

Écrire une séquence d'instructions qui réalise la permutation du contenu de deux variables.

Exercice 4 - Note moyenne

Écrire un algorithme qui, étant donné les résultats (note entière sur 20) de trois examens passés par un étudiant (exprimés par six nombres, à savoir, la note et la pondération de chaque examen), calcule et affiche la moyenne globale exprimée en pourcentage. Vérifiez bien votre algorithme avec des valeurs concrètes ; il est facile de se tromper dans la formule !

Exercice 5 - Somme des chiffres

Écrire un algorithme qui calcule la somme des chiffres d'un nombre entier de 3 chiffres. Réflexion : l'algorithme est-il aussi valide pour les entiers inférieurs à 100 ?

A faire après la partie "Les structures de contrôle"

Exercice 1 - Maximum de 2 nombres

Écrire un algorithme qui, étant donné deux nombres quelconques, recherche et affiche le plus grand des deux. Attention ! On ne veut pas savoir si c'est le premier ou le deuxième qui est le plus grand mais bien quelle est cette plus grande valeur. Le problème est donc bien défini même si les deux nombres sont identiques.

Exercice 2 - Maximum de 3 nombres

Écrire un algorithme qui, étant donné trois nombres quelconques, recherche et affiche le plus grand des trois.

Exercice 3 - Le signe d'un nombre

Écrire un algorithme qui affiche un message indiquant si un entier est strictement négatif, nul ou strictement positif.

Exercice 4 - La fourchette

Écrire un algorithme qui, étant donné trois nombres, recherche et affiche si le premier des trois appartient à l'intervalle donné par le plus petit et le plus grand des deux autres (bornes exclues). Qu'est-ce qui change si on inclut les bornes ?

Exercice 5 - Nombre de jours dans un mois

Écrire un algorithme qui donne le nombre de jours dans un mois. Le mois est lu sous forme d'un entier (1 pour janvier...). On considère dans cet exercice que le mois de février comprend toujours 28 jours.

Exercice 6 - Année bissextile

Écrire un algorithme qui vérifie si une année est bissextile. Pour rappel, les années bissextiles sont les années multiples de 4. Font exception, les multiples de 100 (sauf les multiples de 400 qui sont bien bissextiles). Ainsi :

- 2010 n'est pas bissextile
- 2012 est bissextile
- 2100 n'est pas bissextile
- 2400 est bissextile

Exercice 7 - Valider une date

Écrire un algorithme qui valide une date donnée par trois entiers : l'année, le mois et le jour.

Exercice 8 - Le stationnement alternatif

Dans une rue où se pratique le stationnement alternatif, du 1 au 15 du mois, on se gare du côté des maisons ayant un numéro impair, et le reste du mois, on se gare de l'autre côté. Écrire un algorithme qui, sur base de la date du jour et du numéro de maison devant laquelle vous vous êtes arrêté, indique si vous êtes bien stationné ou non.

A faire après la partie "Les boucles avec Ruby"

Exercice 1 - Afficher le n premiers chiffres

Écrire un algorithme qui lit un naturel n et affiche :

- les n premiers entiers strictement positifs ;
- les n premiers entiers strictement positifs en ordre décroissant ;
- les n premiers carrés parfaits ;
- les n premiers entiers strictement positifs impairs ;
- les entiers strictement positifs impairs qui sont inférieurs ou égaux à n.

Exercice 2 - Maximum de nombres

Écrire un algorithme qui lit une série de cotes d'interrogation (entiers entre 0 et 20) et affiche ensuite la plus grande. Pour signaler la fin de l'encodage des cotes, l'utilisateur introduit un nombre négatif (valeur sentinelle).

Exercice 3 - Afficher les multiples de 3

Écrire un algorithme qui lit une série de nombres entiers positifs, jusqu'à ce que l'utilisateur encode la valeur 0. Les nombres multiples de 3 seront affichés au fur et à mesure et le nombre de ces multiples sera affiché en fin de traitement.

Exercice 4 - Génération de suites

Écrire un algorithme qui affiche les n premiers termes des suites suivantes :

a) Le pas croissant
1, 2, 4, 7, 11, 16, . . .

b) La boiteuse
1, 2, 4, 5, 7, 8, 10, 11, . . .

c) La suite de Fibonacci
0, 1, 1, 2, 3, 5, 8, 13, 21, . . .

Exercice 5 - Factorielle

Écrire un algorithme qui retourne la factorielle de n (entier positif ou nul). Rappel : la factorielle de n, notée n !, est le produit des n premiers entiers strictement positifs. Par convention, 0 ! = 1.

Exercice 6 - Somme de chiffres

Écrire un algorithme qui retourne la somme des chiffres qui forment un nombre naturel n. Attention, on donne au départ le nombre et pas ses chiffres. Exemple : 133045 doit donner comme résultat 16, car 1 + 3 + 3 + 0 + 4 + 5 = 16.

Exercice 7 - Jeu de la fourchette

Écrire un algorithme qui simule le jeu de la fourchette. Ce jeu consiste à essayer de découvrir un nombre quelconque compris entre 1 et 100 inclus, tiré au sort par l'ordinateur (la primitive hasard(n : entier) retourne un entier entre 1 et n). L'utilisateur a droit à huit essais maximum. À chaque essai, l'algorithme devra afficher un message indicatif « nombre donné trop petit » ou « nombre donné trop grand ». En conclusion, soit « bravo, vous avez trouvé en [nombre] essai(s) » soit « désolé, le nombre était [valeur] ».

A faire après la partie "Les structures de données"

Exercice 1 - Somme

Écrire un algorithme qui calcule et affiche la somme des entiers d'un tableau.

Exercice 2 - Maximum et minimum

Écrire un un algorithme qui affiche le plus grand et le plus petit nombre contenu dans un tableau d'entiers.

Exercice 3 - Nombre d'éléments d'un tableau

Écrire un un algorithme qui affiche le nombre d'éléments qu'il y a dans un tableau.

Exercice 4 - Plus grand écart

Écrire un un algorithme qui calcule le plus grand écart entre deux entiers consécutifs dans un tableau.

Exercice 5 - Tableau ordonné ?

Écrire un un algorithme qui affiche "vrai" si un tableau d'entiers est ordonné (strictement) croissant sur les valeurs, ou "faux" si ce n'est pas le cas.

Exercice 6 - Renverser un tableau

Écrire un un algorithme qui « renverse » un tableau, c'est-à-dire qui permute le premier élément avec le dernier, le deuxième élément avec l'avant-dernier et ainsi de suite.

Exercice 7 - Occurence des chiffres

Écrire un un algorithme qui affiche pour chacun des chiffres le nombre de fois qu'il apparait dans un nombre. Ainsi, pour le nombre 10502851125, l'affichage mentionnera que le chiffre 0 apparait 2 fois, 1 apparait 3 fois, 2 apparait 2 fois, 5 apparait 3 fois et 8 apparait une fois (l'affichage ne mentionnera donc pas les chiffres qui n'apparaissent pas).

A faire après la partie "Blocs et tri"

Exercice 1 - Échange de variables

Écrire une méthode appelée *swap* qui échange le contenu de deux variables entières passées en paramètres.

Ensuite, permettre à l'utilisateur d'entrer deux valeurs, afficher les valeurs pour A et B, utiliser votre nouvelle méthode pour permuter les variables et afficher les nouvelles valeurs pour A et B.

Exercice 2 - Maximum de 2 nombres

Écrire une méthode qui retourne le plus grand nombre des 2 nombres donnés en paramètre.

Ensuite, permettre à l'utilisateur d'entrer deux valeurs et afficher la plus grande des deux.

Exercice 3 - Maximum de 4 nombres

Écrire une méthode qui retourne le plus grand nombre des 4 nombres donnés en paramètre en réutilisant la méthode de l'exercice précédent.

Ensuite, permettre à l'utilisateur d'entrer quatre valeurs et afficher la plus grande des quatre.

Exercice 4 - Le chiffrement de César

Depuis l'antiquité, les hommes politiques, les militaires, les hommes d'affaires cherchent à garder secret les messages importants qu'ils doivent envoyer. L'empereur César utilisait une technique (on dit un chiffrement) qui porte à présent son nom : remplacer chaque lettre du message par la lettre qui se situe k position plus loin dans l'alphabet (cycliquement).

Exemple : si k vaut 2, alors le texte clair "CESAR" devient "EGUCT" lorsqu'il est chiffré et le texte "ZUT" devient "BWV".

Bien sûr, il faut que l'expéditeur du message et le récepteur se soient mis d'accord sur la valeur de k.

On vous demande d'écrire une méthode qui reçoit une chaine ne contenant que des lettres majuscules ainsi que la valeur de k et qui retourne la version chiffrée du message.

Vous utiliserez ensuite la méthode pour chiffrer le message encodé par l'utilisateur, et vous afficherez le message codé.

Exercice 5 - Validité d'une date

Reprendre l'algorithme de validation d'une date développé dans la partie "Les structures de contrôle" et le rendre modulaire. C'est-à-dire que vous aurez plusieurs méthodes dans votre code :

- Une méthode qui vérifie la validité du jour. Elle recevra en paramètre le jour et retournera un booléen ("vrai" si le jour est valide, sinon "faux") ;

- Une méthode qui vérifie la validité du mois (exprimé en nombre, de 1 à 12). Elle recevra en paramètre le mois et retournera un booléen ("vrai" si le mois est valide, sinon "faux") ;

- Une méthode qui vérifie la validité d'une année. Elle recevra en paramètre l'année et retournera un booléen ("vrai" si l'année est valide, sinon "faux") ;

- Une méthode qui utilise les trois précédentes pour vérifier qu'une date est bien valide. Cette méthode recevra en paramètre le jour, le mois et l'année, chacun sous la forme d'un entier.

Vous afficherez "vrai" si la date encodée par l'utilisateur est valide, sinon vous afficherez "faux".

Attention ! On ne tiendra pas compte des années bissextiles dans cet exercice.

Challenge - Afficher un rectangle

Pour ce dernier exercice, on vous donne peu de consignes, peu de pistes, c'est à vous de vous débrouiller pour trouver une solution qui fonctionne et construire un code propre, lisible, modulaire et réutilisable.

Écrire un programme qui affiche un rectangle. L'utilisateur encodera la longueur et la largeur.

Vous écrirez une méthode qui recevra en paramètre la longueur et la largeur et affichera le rectangle avec les bonnes dimensions.

Exemples :

rectangle(6,3) affiche :

```
o----o
|    |
o----o
```

rectangle(3,4) affiche :

```
o-o
|  |
|  |
o-o
```