
RUDIMENTS DE LA PROGRAMMATION



APERÇU

1. Concepts fondamentaux de la programmation
2. Éléments du code
3. Conception au moyen de pseudocode
4. Transfert du pseudocode à un code exécutable



CONCEPTS FONDAMENTAUX DE LA PROGRAMMATION

RUDIMENTS DE LA PROGRAMMATION

BUT ET OBJECTIFS D'APPRENTISSAGE DE LA PRÉSENTE SECTION

1. Vous transmettre les concepts fondamentaux qui conviennent à *tout* langage de programmation.
2. Vous transmettre les éléments communs à tous les langages de programmation.
3. Vous aider à *apprendre* tout langage de programmation au moyen des concepts fondamentaux communs.
4. Vous préparer en vue de votre premier exercice – lorsque vous concevrez et créerez des programmes informatiques en langages R ou Python.

DISCUSSION DE GROUPE

Qu'est-ce que le code informatique?

Qu'est-ce qu'un programme informatique?

```
#include <stdio.h>
int main()
{
    double firstNumber, secondNumber, temporaryVariable;

    printf("Enter first number: ");
    scanf("%lf", &firstNumber);

    printf("Enter second number: ");
    scanf("%lf",&secondNumber);

    // Value of firstNumber is assigned to temporaryVariable
    temporaryVariable = firstNumber;

    // Value of secondNumber is assigned to firstNumber
    firstNumber = secondNumber;

    // Value of temporaryVariable (which contains the initial value of firstNumber)
    secondNumber = temporaryVariable;

    printf("\nAfter swapping, firstNumber = %.2lf\n", firstNumber);
    printf("After swapping, secondNumber = %.2lf", secondNumber);

    return 0;
}
```

PROGRAMME INFORMATIQUE : EXEMPLE EN LANGUAGE C

Un algorithme écrit dans un langage informatique donne à un ordinateur des instructions pour exécuter une suite d'opérations.

PROGRAMME INFORMATIQUE : DÉFINITION

Un **algorithme** écrit dans un **langage informatique** donne à un ordinateur des instructions pour exécuter une **suite d'opérations**.

On peut le **compiler** ou l'**interpréter** sous forme d'une suite d'opérations matérielles exécutées par les **composants électriques** d'un ordinateur.

QUELQUES CONCEPTS FONDAMENTAUX

Algorithme

Langage informatique

Langage (normal)

LANGAGE NORMAL : EXEMPLE

Alphabet : {« a », « b », « C », « D », « ! »}

Règles (grammaire) :

- On peut placer une lettre à la gauche ou à la droite d'une autre lettre.
- Une lettre doit toujours avoir une lettre identique à sa gauche ou à sa droite.
- Une lettre majuscule doit toujours avoir une lettre minuscule à sa gauche ou à sa droite.

LANGAGE NORMAL : DÉFINITION

Dans un langage normal :

- les mots sont créés à partir d'un alphabet prédéfini;
- une grammaire définit des règles relatives à la manière de regrouper des lettres pour créer des mots.

LANGAGE INFORMATIQUE : DÉFINITION

Langage (normal) créé pour donner des instructions à **un ordinateur**, de manière qu'on puisse le compiler en instructions de bas niveau que peut **exécuter** le processeur de l'ordinateur.

In the lexical and syntax rules given below, BNF notation characters are written in green.

- Alternatives are separated by vertical bars: i.e., '*a* | *b*' stands for "*a* **or** *b*".
- Square brackets indicate optionality: '*[a]*' stands for an optional *a*, i.e., "*a* | *epsilon*" (here, *epsilon* refers to the empty sequence).
- Curly braces indicate repetition: '*{ a }*' stands for "*epsilon* | *a* | *aa* | *aaa* | ...".

1. Lexical Rules

letter ::= a | b | ... | z | A | B | ... | Z

digit ::= 0 | 1 | ... | 9

id ::= *letter* { *letter* | *digit* | *_* }

intcon ::= *digit* { *digit* }

charcon ::= '*ch*' | '\n' | '\0', where *ch* denotes any printable ASCII character, as specified by **isprint()**, other than \ (backslash) and ' (single quote).

stringcon ::= "{*ch*}", where *ch* denotes any printable ASCII character (as specified by **isprint()**) other than " (double quotes) and the newline character.

Comments Comments are as in C, i.e. a sequence of characters preceded by */** and followed by **/*, and not containing any occurrence of **/*.

LANGAGE INFORMATIQUE : DÉFINITION OFFICIELLE DU LANGAGE C

Langage créé pour donner des instructions à un ordinateur

PROGRAMME INFORMATIQUE : DÉFINITION

Un **algorithme** écrit dans un **langage informatique** donne à un ordinateur des instructions pour exécuter une **suite d'opérations**.

On peut le **compiler** ou l'**interpréter** sous forme d'une suite d'opérations matérielles exécutées par les **composants électriques** d'un ordinateur.

EXEMPLE D'UN ALGORITHME

1. Verser $\frac{1}{2}$ tasse de farine dans un bol.
2. Ajouter un œuf dans le bol.
3. Verser 3 cuillères à table d'huile dans le bol.
4. Verser 1 cuillère à thé de poudre à pâte dans le bol.
5. Mélanger avec une cuillère jusqu'à l'obtention d'une texture lisse.
6. Verser le mélange dans le moule à muffins.
7. Cuire pendant 15 minutes à 350 °F.



ALGORITHME : DÉFINITION

Suite d'instructions comportant au moins un point d'arrêt bien défini.

PROGRAMME INFORMATIQUE : DÉTAILS

Les langages informatiques évolués sont compilés en (ou interprétés sous forme de) code machine, c.-à-d. une suite d'instructions élémentaires qui indiquent au matériel informatique comment se comporter.

Lorsque l'ordinateur exécute les instructions, nous disons qu'il « exécute » le programme – sous forme d'un **processus**.

Nous pouvons indiquer à un ordinateur d'exécuter un programme. Un ordinateur peut aussi lancer lui-même un programme!

```
MONITOR FOR 6802 1.4          9-14-80  TSC ASSEMBLER  PAGE    2

C000                                ORG    ROM+$0000 BEGIN MONITOR
C000 8E 00 70  START  LDS    #STACK

*****
* FUNCTION: INITA - Initialize ACIA
* INPUT: none
* OUTPUT: none
* CALLS: none
* DESTROYS: acc A

0013                                RESETA EQU    %00010011
0011                                CTLREG EQU    %00010001

C003 86 13  INITA  LDA  A  #RESETA  RESET ACIA
C005 B7 80 04                                STA  A  ACIA
C008 86 11                                LDA  A  #CTLREG  SET 8 BITS AND 2 STOP
C00A B7 80 04                                STA  A  ACIA

C00D 7E C0 F1                                JMP    SIGNON  GO TO START OF MONITOR

*****
* FUNCTION: INCH - Input character
* INPUT: none
* OUTPUT: char in acc A
* DESTROYS: acc A
* CALLS: none
* DESCRIPTION: Gets 1 character from terminal
```

PROGRAMME INFORMATIQUE : VUE D'ENSEMBLE

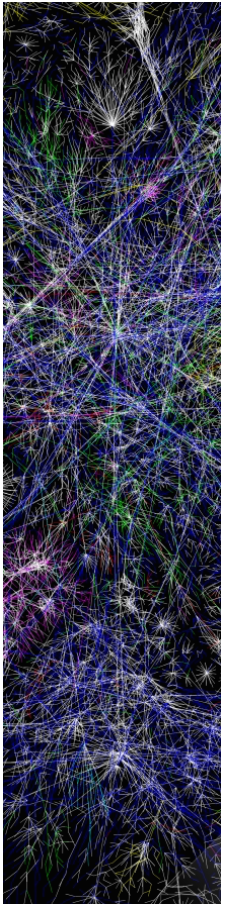
Tous les ordinateurs fonctionnent en exécutant des programmes informatiques (compilés).

Internet est un regroupement d'ordinateurs connectés par des fils ou des émetteurs et des récepteurs radio.

Un ordinateur transmet des signaux à d'autres ordinateurs sur ce réseau et reçoit des signaux d'autres ordinateurs.

Les signaux envoyés d'un ordinateur à un autre, et le traitement effectué sur les signaux reçus, dépendent des programmes exécutés par les ordinateurs.

Le nuage est une partie d'Internet grossièrement définie comme étant un regroupement d'ordinateurs utilisés principalement pour stocker du contenu et en envoyer à d'autres ordinateurs.



ÉLÉMENTS DU CODE

RUDIMENTS DE LA PROGRAMMATION

DISCUSSION DE GROUPE

Quels sont les éléments fondamentaux du code informatique?

ÉLÉMENTS DU CODE INFORMATIQUE

Variables

Structures des données

Opérateurs

Énoncés et expressions

Blocs (et portée)

Fonctions

Flux logique (de commande)

Bibliothèques/trousses/modules

Données d'entrée/données de sortie

Interpréteurs/compilateurs

load additional functions (package/module/
library) from outside the current code

variable

user-defined function
with three arguments

block

```
library(igraph)
```

```
my_graph_function <- function(my_number_nodes, my_colour, my_density)
```

```
{
```

```
my_graph <- sample_gnp(my_number_nodes, my_density, directed = FALSE, loops = FALSE)
```

```
if(ecount(my_graph) >= my_number_nodes){V(my_graph)$color <- my_colour}
```

```
plot(my_graph, layout=layout_fruchterman_reingold, vertex.color=V(my_graph)$color)
```

```
}
```

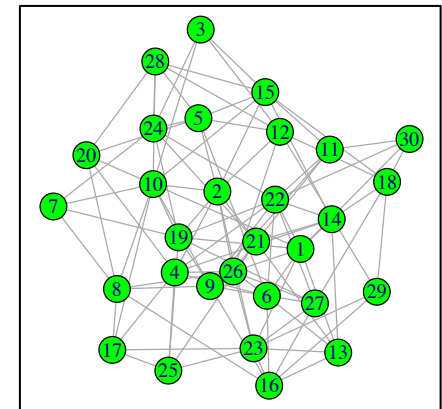
```
my_graph_function(30, "green", 0.3)
```

creating a data structure/
object (a graph)

conditional logic
statement (control flow)

calling the user-defined function

generating output
(a visualization of the graph)



PRÊTS À FAIRE DE LA PROGRAMMATION?

Pas si vite!



CONCEPTION DE CODE (UTILISATION DE PSEUDOCODE)

RUDIMENTS DE LA PROGRAMMATION

DISCUSSION DE GROUPE

Qu'est-ce que ça signifie que de concevoir un algorithme, ou un programme?

ÉLÉMENTS DE LA CONCEPTION

Lors de la conception d'un algorithme, nous devons préciser :

- les données d'entrée;
- les données de sortie;
- la méthode pour transformer les données d'entrée en données de sortie.

D'un point de vue plus général, nous pouvons également parler de la fonction ou de l'objet de l'algorithme.

PSEUDOCODE : EXEMPLE

```
j_cluster(array_of_points, max_n_neighbour_distance)
{
    for each point[i] in array_of_points
    {
        for each remaining point[j] in array_of_points
        {
            distance_between_ij = distance(point[i], point[j])
            if distance_between_ij <= max_n_neighbour_distance
            then neighbours[i] = add_to_neighbours(point[i],point[j])
        }
    }
    ...
}
```

PSEUDOCODE : CE À QUOI IL RESSEMBLE RÉELLEMENT!

~~1-cluster~~
~~BS CAN~~ (any of points, ^{med neighbor distance})

for each point[i] in array-of-points

{ for each remaining point[j]

{ distance-between_{i,j} = distance
(point[i], remaining
point[j])
}

}
if distance-between_{i,j} \leq max-neighbor
distance
neighbor[i] = add-to-neighborhood(point[i], p_j)

PSEUDOCODE : DESCRIPTION

Le terme « **pseudocode** » désigne l'ébauche d'un algorithme et donne une idée générale des données d'entrée, des données de sortie et des étapes, sans tenir compte des détails des fonctions.

À partir des principaux éléments de tout langage informatique (p. ex. variables, fonctions, flux logique, etc.), nous pouvons concevoir un algorithme sans utiliser un langage en particulier.

PSEUDOCODE : STRATÉGIE

Définissez les données d'entrée.

Définissez les données de sortie.

Rédigez un jeu d'instructions qui vous fait passer des données d'entrée aux données de sortie.

N'oubliez pas que vous pouvez « obscurcir » des parties du code – décrire des fonctions de manière générale.

PSEUDOCODE : NIVEAU D'ABSTRACTION

Il faut beaucoup de pratique pour utiliser le bon niveau de détails dans un pseudocode.

Jusqu'à un certain point, tout dépend du niveau d'abstraction du langage de programmation que vous utiliserez (selon toute probabilité) :

- un langage évolué – comporte un grand nombre de fonctions intégrées;
- un langage de bas niveau – vous devrez programmer de nombreux détails et de nombreuses fonctions.

Un langage évolué permet la programmation selon un niveau d'abstraction plus élevé.

Toutefois, vous risquez de sacrifier l'utilité pour la compréhension.

EXERCICE DE CONCEPTION DE PSEUDOCODE ET D'UN ALGORITHME : TRI

Vos données d'entrée sont une liste de nombres sans ordre particulier.

Vos données de sortie doivent être la même liste de nombres triés dans le bon ordre.

Rédigez un pseudocode qui vous fait passer des données d'entrée aux données de sortie.

Retenez que vous pouvez obscurcir des parties du code – décrire les fonctions de manière générale.

En ce qui concerne le niveau de détails, tenez compte de la manipulation de chaque nombre ou d'un groupe de nombres de la liste.

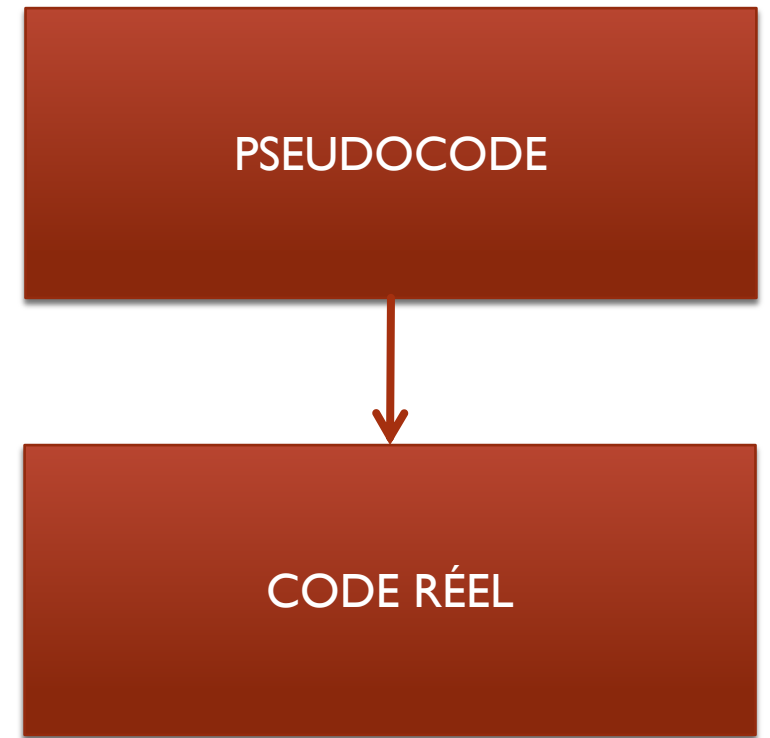
PASSAGE DU PSEUDOCODE AU CODE EXÉCUTABLE

RUDIMENTS DE LA PROGRAMMATION

LA RÉALITÉ

Pour transformer un pseudocode en code réel, vous devez effectuer certaines étapes :

- déterminer la syntaxe appropriée dans le langage que vous voulez utiliser et récrire votre pseudocode sous forme d'un code réel dans ce langage;
- remplacer les fonctions obscurcies par le code réel;
- Déterminer la manière de raccorder le code (le logiciel) à l'ordinateur, pour que vous puissiez compiler/interpréter votre code et l'exécuter sur l'ordinateur de manière à ce que votre code traite les données d'entrée et produise les données de sortie.



DU CODE À L'ORDINATEUR

De nombreux **obstacles** peuvent surgir lorsque vous voulez transformer votre code – qui est en fait un simple fichier texte – en un code informatique que peut exécuter votre ordinateur. Ces obstacles sont :

- l'accès aux bibliothèques;
- l'utilisation des données d'entrée et de sortie, ainsi du système de fichiers;
- l'utilisation des compilateurs et des interpréteurs.

En gros, il doit y avoir une certaine infrastructure!

Nous nous occupons d'une bonne partie de cette infrastructure en configurant pour vous des notebooks.

RESSOURCES DE PROGRAMMATION

La plupart des renseignements sur l'utilisation d'un langage informatique ou sur l'exécution d'un code en fonction d'une configuration matérielle particulière **ne sont pas rédigés** dans quelque **manuel de référence officiel** que ce soit.

La raison est toute simple : les codes et les ordinateurs évoluent trop vite.

Pour créer un code efficace, vous devez vous **joindre à une communauté de codeurs**. Heureusement, Internet facilite cette étape – la plupart des questions sur la création d'un code ont déjà été abordées quelque part sur Internet.

Autrement dit, visitez **STACK EXCHANGE** (et autres sites semblables).

ALGORITHME DE TRI : ÉBAUCHE EN LANGAGE R

Votre défi : Au moyen du notebook R fourni, rédigez (en langage R) et exécutez un programme pour trier les nombres.

R STUDIO

The screenshot displays the R Studio environment. The top toolbar includes icons for file operations and a 'Go to file/function' search bar. The main editor window shows two tabs: 'Untitled1*' and 'nodo_dataset_igraphlab'. The console at the bottom left contains the following text:

```
R version 3.2.1 (2015-06-18) -- "World-Famous Astronaut"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]
```

Below the console, a red message states: "Loading required package: RMySQL".

The Environment pane on the right shows the 'Global Environment' with the following data and values:

Object	Value
data	813 obs. of 2 variables
mydb	Formal class MySQLConnection
namecounts	'table' int [1:256(1d)] 2 8 4 1 1 2 6 1 2 1 ...
namelist	chr [1:256] "Aaron" "Aboriginal" "Adam" "Adrianna" ...
rs	Formal class MySQLResult

The Files pane at the bottom right shows a file explorer view of the 'Home' directory. It lists various files and folders with their sizes and modification dates.

Name	Size	Modified
.RData	6.7 KB	Jul 11, 2018, 4:09 PM
.Rhistory	178 B	Sep 17, 2019, 5:31 PM
Applications		
CHLPA_proc_20150323.pdf	190.4 KB	Nov 1, 2015, 10:02 PM
CHLPA_proc_20150323.vdx	109.6 KB	Nov 1, 2015, 10:02 PM
CHLPA_report20150323.docx	1.4 MB	Nov 1, 2015, 10:02 PM
CHLPA_simple_simulation_20150323.xlsx	84.4 KB	Nov 1, 2015, 10:02 PM
CHLPAdocumentflow_20150323.pdf	109.5 KB	Nov 1, 2015, 10:02 PM
CHLPAdocumentflow_20150323.vdx	95.3 KB	Nov 1, 2015, 10:02 PM

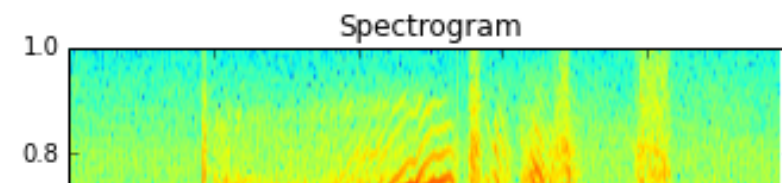
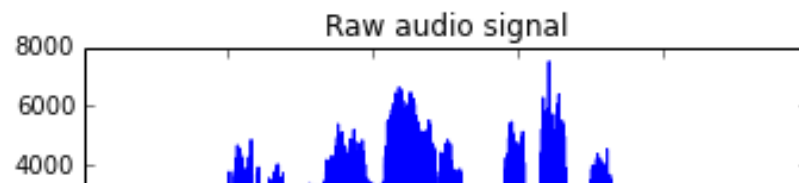
$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{j2\pi}{N}kn} \quad k = 0, \dots, N-1$$

We begin by loading a datafile using SciPy's audio file support:

```
In [1]: from scipy.io import wavfile
rate, x = wavfile.read('test_mono.wav')
```

And we can easily view its spectral structure using matplotlib's builtin specgram routine:

```
In [2]: %matplotlib inline
from matplotlib import pyplot as plt
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.specgram(x); ax2.set_title('Spectrogram');
```



ANATOMIE DU NOTEBOOK JUPYTER

ÉLÉMENTS DU CODE INFORMATIQUE R

Variables

Structures de données

Opérateurs

Énoncés et expressions

Blocs (et portée)

Fonctions

Flux logique (de commande)

Bibliothèques/trousses/modules

Données d'entrée/données de sortie

Interpréteurs/compilateurs

R Reference Card

by Tom Short, EPRI PEAC, tshort@epri-peac.com 2004-11-07
Granted to the public domain. See www.Rpad.org for the source and latest version. Includes material from *R for Beginners* by Emmanuel Paradis (with permission).

Getting help

Most R functions have online documentation.

help(topic) documentation on topic

?topic id.

help.search("topic") search the help system

apropos("topic") the names of all objects in the search list matching the regular expression "topic"

help.start() start the HTML version of help

str(a) display the internal *str*ucture of an R object

summary(a) gives a "summary" of a, usually a statistical summary but it is *generic* meaning it has different operations for different classes of a

ls() show objects in the search path; specify `pat="pat"` to search on a pattern

ls.str() `str()` for each variable in the search path

dir() show files in the current directory

methods(a) shows S3 methods of a

methods(class=class(a)) lists all the methods to handle objects of class a

LANGAGE R : QUELQUES RENSEIGNEMENTS IMPORTANTS (I)

Pour créer une **variable** en langage R, créez simplement un nom et utilisez l'opérateur d'attribution pour attribuer une valeur à la variable.

La valeur peut être une variable de type nombre, caractère, chaîne, vecteur, liste, matrice, cadre de données ou tout autre objet.

Le langage R utilise beaucoup le cadre de données!

```
> my_number <- 5
> my_string <- "Jen"
> my_vector <- c(1,2,3,4)
> my_list <- list(1,2,3,4)
> my_data_frame <-
data.frame(c("Jen", "Pat"), c(4, 2), c(6, 10))
> colnames(my_data_frame) <-
c("Name", "Shoe_Size", "Score")
> my_data_frame
```

	Name	Shoe_Size	Score
1	Jen	4	6
2	Pat	2	10

LANGAGE ORIENTÉ OBJET ET LANGAGE PROCÉDURAL

Les langages R et Python sont des langages orientés objet, plutôt que des langages procéduraux.

Qu'est-ce que cela signifie?

Pour comprendre la réponse, nous devons d'abord comprendre :

- les types de données;
- les structures de données;
- les fonctions.

TYPES DE DONNÉES

Un langage comporte un jeu intégré de variables types de base – p. ex. :

- entier : 5
- caractère : « m »
- liste : (5, 3, 9)

On peut créer d'autres variables types à partir de ces variables de base – p. ex. :

- chaîne = liste de caractères : ('t', 'a', 'b', 'l', 'e')

STRUCTURES DE DONNÉES ET OBJETS

Un utilisateur peut définir son propre jeu de variables connexes – une structure de données :

- `struct myNames = {string firstName, string middleName, string lastName}`
- `jenNames` peut être une variable de type `myNames`, où `firstName = Jen`, `middleName = Adele`, `lastName = Schellinck`

En outre, un programmeur peut vouloir être toujours en mesure d'exécuter un jeu d'instructions prédéfinies, ou fonctions, sur cette structure de données :

- `jenNames.print_middle_name`

Un objet désigne **en gros** une structure de données définie par l'utilisateur et un ensemble de fonctions liées à cette structure.

CADRE DE DONNÉES EN LANGAGE R

L'**objet** de cadre de données (« data frame ») en langage R est structuré comme une feuille de calcul Excel :

- il comporte des lignes et des colonnes, chacune désignée par un nom;
- vous pouvez exécuter des opérations prédéfinies sur des valeurs précises, sur des lignes ou sur des colonnes.

Une personne habituée de travailler avec une base de données OU avec un langage plus orienté vecteur (p. ex. Java) pourrait éprouver de la frustration à utiliser le cadre de données du langage R!

ALGORITHME DE TRI : ÉBAUCHE EN LANGAGE R

Votre défi : Au moyen du notebook R fourni, rédigez (en langage R) et exécutez un programme pour trier les nombres.

ALGORITHME DE TRI : ÉBAUCHE EN PYTHON

Votre défi Au moyen du notebook Python fourni, rédigez (en Python) et exécutez un programme pour trier les nombres.

QUELQUES RENSEIGNEMENTS UTILES

LANGAGE COMPILÉ ET LANGAGE INTERPRÉTÉ

Langage compilé : Le programme est écrit en entier, le compilateur vérifie le code **dans son ensemble** et le transforme en langage de bas niveau.

Langage interprété : L'interpréteur lit le code, le transforme en code de bas niveau et exécute **un énoncé à la fois**.

Avec un interpréteur, vous programmez d'une manière plus libre, presque improvisée – comme jouer du jazz au lieu de la musique classique.

L'interpréteur peut être utile lorsque vous réalisez un travail exploratoire, mais vous risquez de rencontrer des difficultés si vous adoptez cette stratégie pour créer des programmes plus gros ou plus substantiels.

DÉBOGAGE

Le débogage consiste principalement à révéler ce qui se trouve dans la mémoire à divers points du flux de commande du code – est-ce que le code fait ce que vous pensez qu'il doit faire?

Le débogage est un art.

Le débogage nécessite que vous deveniez un détective.

Le débogage vous enseigne la persévérance.

Il existe des outils de débogage qui peuvent vous aider.

QUELQUES CADRES PERTINENTS DE L'INFORMATIQUE

Langages (informatique, de balisage)

Bibliothèques/API

Logiciels (applications, utilitaires, systèmes)

Code (source ouvert, non compilé)

Protocole/norme

Modèles/styles de programmation

EXERCICES ET LECTURES COMPLÉMENTAIRES

RUDIMENTS DE LA PROGRAMMATION

EXERCICES : LABORATOIRES À VENIR!

Au cours de la prochaine séance, vous aurez l'occasion d'en apprendre davantage sur les langages R et Python ainsi que d'exécuter des exercices au moyen de notebooks **préparés**.

À partir de février, vous aurez des laboratoires dans le cadre desquels **vous serez responsable de la programmation**.

Avant ces laboratoires, en guise de devoirs, nous vous encourageons à :

- revoir le notebook « Introduction to R »;
- revoir les notebooks d'introduction au langage Python;
- **essayer de rédiger des énoncés et de courts codes en langage R et en langage Python** dans l'environnement des notebooks, afin de vous familiariser avec ces deux langages.

AUTRES EXERCICES : ESSAYEZ-LES

À partir d'un notebook vide (nouveau) :

- chargez des données à partir d'un fichier dans un cadre de données;
- créez un tracé ou un graphique au moyen des données chargées.

Remarque – Il **est permis d'utiliser** du code d'un autre notebook ou d'une ressource en ligne!

RÉFÉRENCES

RUDIMENTS DE LA PROGRAMMATION

RÉFÉRENCES

Exemple de spécifications en langage C :

<https://www2.cs.arizona.edu/~debray/Teaching/CSc453/DOCS/cminusminusspec.html>

Court code en langage C : <https://www.programiz.com/c-programming/examples/swapping>

Aide-mémoire sur l'application Jupyter Notebook :

https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Jupyter_Notebook_Cheat_Sheet.pdf

Aide-mémoire sur le langage R : <https://cran.r-project.org/doc/contrib/Short-refcard.pdf>

Aide-mémoire sur Markdown :

https://scottboms.com/downloads/documentation/markdown_cheatsheet.pdf

IMAGES

Poteau indicateur :

https://upload.wikimedia.org/wikipedia/commons/2/29/Worden_park_signpost.jpg

Muffins :

https://commons.wikimedia.org/wiki/Category:Muffins#/media/File:Sweet_potato_pecan.jpg

Langage assembleur :

https://en.wikipedia.org/wiki/Assembly_language#/media/File:Motorola_6800_Assembly_Language.png

Carte d'Internet :

https://upload.wikimedia.org/wikipedia/commons/d/d2/Internet_map_1024.jpg