Chapitre - Les données structurées et leur traitement

Repères historiques

1930: utilisation des cartes perforées, premier support de stockage de données;
1956: invention du disque dur permettant de stocker de plus grandes quantités de données, avec un accès de plus en plus rapide ;
1970: invention du modèle relationnel (E. L. Codd) pour la structuration et l'indexation des bases de données ;
1979: création du premier tableur, VisiCalc ;
2009: Open Government Initiative du président Obama ;
2013: charte du G8 pour l'ouverture des données publiques.

Introduction

Les données constituent la matière première de toute activité numérique. Afin de permettre leur réutilisation, il est nécessaire de les conserver de manière persistante. Les structurer correctement garantit que l'on puisse les exploiter facilement pour produire de l'information.

I. Les données et l'information

1. Que sont les données?

Une *donnée* est en informatique la représentation d'une information, formatée de manière adaptée à sa nature et pour être exploitée par des applications. Toutes les informations humaines se codent en binaire ; bien entendu ce n'est pas l'objet réel, ce n'est que son reflet numérique.

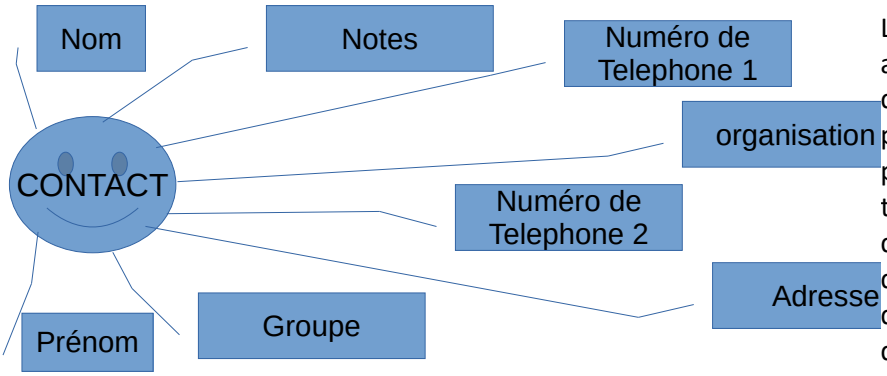
Exemple : le numéro de téléphone d'un contact est une donnée.

Dans de nombreux cas, cette représentation prend la forme d'une valeur numérique. Mais les données sous-jacentes sont très diverses : textes, image, sons.

Pour connaître le **type de donnée** à laquelle on a affaire:
l'instruction type()-->

type(nom_de_la_variable)
nous donne <class 'str' ou int ou ...>

Comme les données peuvent être des objets relativement complexes, on utilise des ***descripteurs*** **qui sont utiles pour décrire cet objet.**
Petite application : prenons une donnée comme un **contact** (dans votre téléphone), combien de descripteurs pensez-vous devoir utiliser au minimum, lesquels?.
--> NOM, Prénom, numéro, mail, adresse , renseignements sup....



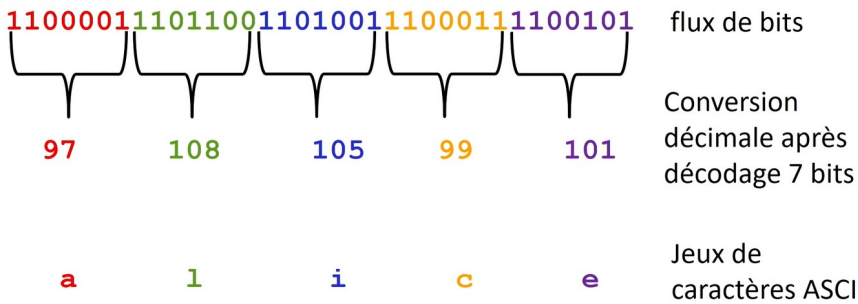
Le fichier contact est donc associé à un certain nombre de descripteurs qui vont permettre à des programmes de pouvoir travailler avec, retrouver des informations.... Ces descripteurs se doivent donc d'être rangés toujors de la même façon!!!
Pas comme votre chambre!

Une ***collection*** regroupe des objets (élémentaires ou complexes) partageant le même descripteur (par exemple, la collection des vidéos d'un Youtubeur aura un descripteur commun: leur auteur).

2. Les formats de données

Un format de données est la façon dont est représenté (codé) un type de données. Au niveau inférieur : une information est toujours codée sous forme d'une suite de bits (dans les machines, il n'y a que des bits, soit des séries de 0 et de 1). Par commodité, on interprète cette suite de bits comme un nombre binaire, et on dit par raccourci que la donnée est représentée comme un nombre.

Exemple de codage (ASCII - American Standard Code for Information Interchange - car il existe plusieurs conventions de codage des caractères) des caractères de l'alphabet :
Le flux de bits est décodé de la façon suivante



En **ASCII** (American Standart Code for Information Interchange.)on code 95 caractères: les 10 chiffres, les 26 lettres minuscules, les 26 lettres majuscules, 32 symboles (@, <, >..), l'espace; et 33 symboles de mise en page (passage à la ligne, saut de page...), soit 128 caractères.
Il faut donc 7 bits, mais on code sur 1 octet, le premier bit étant toujours 0 et sert au contrôle de parité (pour éviter des erreurs)

- Les caractères de numéro 0 à 31 et le 127 ne sont pas affichables ; ils correspondent à des commandes de contrôle de terminal informatique.
- Le caractère numéro 127 est la commande pour effacer.
- Les chiffres sont codés par les nombres de 48 à 57
- Les lettres majuscules par les nombres de 65 à 90
- Les minuscules par les nombres de 97 à 122
- @ est codé par 64
- Dans un texte le premier octet donne la longueur du texte , le dernier est 00000000 indique la fin du texte. Un texte de n caractères occupe donc une place de n+2 octets

TEST/ vous pouvez faire l'expérience en écrivant un texte dans notepad++ ou enregistré en format text ASCII et en l'enregistrant avec l'extension .text, regarder ses propriétés et la place occupée...
Le recopier dans un éditeur de texte (word ou openoffice.org) regarder la place occupée...

Dec.	Car.	Binaire	Dec.	Car.	Binaire	Dec.	Car.	Binaire	Dec.	Car.	Binaire
000	NUL	00000000	032	SP	00100000	064	@	01000000	096	`	01100000
001	SOH	00000001	033	!	00100001	065	A	01000001	097	a	01100001
002	STX	00000010	034	"	00100010	066	B	01000010	098	b	01100010
003	ETX	00000011	035	#	00100011	067	C	01000011	099	c	01100011
004	EOT	00000100	036	$	00100100	068	D	01000100	100	d	01100100
005	ENQ	00000101	037	%	00100101	069	E	01000101	101	e	01100101
006	ACK	00000110	038	&	00100110	070	F	01000110	102	f	01100110
007	BEL	00000111	039	'	00100111	071	G	01000111	103	g	01100111
008	BS	00001000	040	(	00101000	072	H	01001000	104	h	01101000
009	HT	00001001	041	)	00101001	073	I	01001001	105	i	01101001
010	LF	00001010	042	*	00101010	074	J	01001010	106	j	01101010
011	VT	00001011	043	+	00101011	075	K	01001011	107	k	01101011
012	FF	00001100	044	,	00101100	076	L	01001100	108	l	01101100
013	CR	00001101	045	-	00101101	077	M	01001101	109	m	01101101
014	SO	00001110	046	.	00101110	078	N	01001110	110	n	01101110
015	SI	00001111	047	/	00101111	079	O	01001111	111	o	01101111
016	DLE	00010000	048	0	00110000	080	P	01010000	112	p	01110000
017	DC1	00010001	049	1	00110001	081	Q	01010001	113	q	01110001
018	DC2	00010010	050	2	00110010	082	R	01010010	114	r	01110010
019	DC3	00010011	051	3	00110011	083	S	01010011	115	s	01110011
020	DC4	00010100	052	4	00110100	084	T	01010100	116	t	01110100
021	NAK	00010101	053	5	00110101	085	U	01010101	117	u	01110101
022	SYN	00010110	054	6	00110110	086	V	01010110	118	v	01110110
023	ETB	00010111	055	7	00110111	087	W	01010111	119	w	01110111
024	CAN	00011000	056	8	00111000	088	X	01011000	120	x	01111000
025	EM	00011001	057	9	00111001	089	Y	01011001	121	y	01111001
026	SUB	00011010	058	:	00111010	090	Z	01011010	122	z	01111010
027	ESC	00011011	059	;	00111011	091	[	01011011	123	{	01111011
028	FS	00011100	060	<	00111100	092	\	01011100	124	|	01111100
029	GS	00011101	061	=	00111101	093	]	01011101	125	}	01111101
030	RS	00011110	062	>	00111110	094	^	01011110	126	~	01111110
031	US	00011111	063	?	00111111	095	_	01011111	127	DEL	01111111

Un format de données est ainsi une convention (éventuellement normalisée) utilisée pour représenter des données — des informations représentant un texte, une page, une image, un son, un fichier exécutable, etc. L'existence d'une convention, si elle est bien respectée, permet d'échanger des données entre divers programmes informatiques. On

Tableau des codes ASCII

En premier-->		000	001	010	011	100	101	110	111
En second		0	1	2	3	4	5	6	7
0000	0	NUL	DLE	SP	0	@	P	`	p
0001	1	SOH	DC1	!	1	A	Q	a	q
0010	2	STX	DC2	"	2	B	R	b	r
0011	3	ETX	DC3	#	3	C	S	c	s
0100	4	EOT	DC4	$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v
0111	7	BEL	ETB	'	7	G	W	g	w
1000	8	BS	CAN	(	8	H	X	h	x
1001	9	HT	EM	)	9	I	Y	i	y
1010	10	LF	SUB	*	:	J	Z	j	z
1011	11	VT	ESC	+	;	K	[	k	{
1100	12	FF	FS	,	<	L	\	l	!
1101	13	CR	GS	-	=	M	]	m	}
1110	14	SO	RS	.	>	N	^	n	~
1111	15	SI	US	/	?	O	-	o	DEL

01000001

appelle interopérabilité cette possibilité d'échanger des données entre différents logiciels.

Activité principale du chapitre: le codage en binaire des nombres entiers --> T02-01

Point histoire… pour aller plus loin
Un peu d'histoire des nombres en cliquant sur le lien

Vous avez probablement quelques souvenirs de primaire: pour compter, nous regroupons les unités par paquets de 10.
Le nombre de symboles pour faire ces paquets s'appelle la base. Nous comptons donc en base dix en utilisant des symboles que sont les chiffres de 0 à 9 (cela fait bien 10 possibilités), mais on peut compter en n'importe quelle base supérieure ou égale à deux. Bien sûr, les autres bases peuvent nous paraître étranges tellement nous sommes habitués à notre bonne vieille base dix, mais passé le premier étonnement, le mécanisme est vraiment identique.

Écritures d'un entier naturel positif
Il existe trois principaux codages utilisés pour coder les nombres entiers.

A. Comment décomposer un nombre en base décimale ?
Exemple: décomposez 45620 en puissances de dix.

Chaque « paquet » de 10^n est affecté d'un coefficient compris entre 0 et 9

B- décomposons un nombre en base binaire
Exercice 1: En base 10, un paquet de 2 caractères peuvent représenter 100 nombres entiers différents (de 0 à 99). Sur le même principe combien de nombres entiers naturels peut on représenter avec un nombre n de caractères en binaire :

Valeur de n	Nombre d'entiers naturels représentables.
1	2
3	8
8	256
n	2^n

Exercice 2:
D'abord ensemble :

79 en B10 équivaut à :	$1x2^6$	$+0x2^5+$	$0x2^4+$	$1x2^3+$	$1x2^2+$	$1x2^1+$	$1x2^0$
valeur :	64	32	16	8	4	2	1

représentation de 79 en base binaire→ 1 0 0 1 1 1 1

Voici 5 nombres en base 10: 5, 16, 57, 248 et 348. Donnez leur représentation en base 2 : → Aide possible (sous forme d'un tableur)

A retenir: chaque caractère 0 ou 1 est nommé un bit. Au sein d'un programme informatique, ils sont regroupés en entités de huit bits nommées octet (comme huit!)

Exercice 3:
Décomposez 4620 afin d'exprimer sa valeur en binaire. Quel est l'inconvénient mis en évidence de ce type de codage? Quelle solution peut-on imaginer?

A retenir: pour coder le nombre 4620l faut treize bits ! Donc 2 octets (2 paquets de 8, puisqu'un seul ne suffit pas). La représentation des nombres entiers d'une certaine taille va donc nécessiter de nombreux octets.

C- La représentation d'un nombre en base hexadécimale
La base 16 utilise les symboles que sont les chiifres de 0 à 9 et les lettres de A à F: 0123465789ABCDEF

Exercice 4
Combien de nombres entiers naturels peut on représenter avec un nombre n de caractères en base hexadécimale : si n=1, si n=3, si n=8 ou pour tout n.

Exercice 5:
D'abord ensemble : en base 16, les puissances de 16 peuvent être multipliées par des valeurs de 0 à 15 représentées :

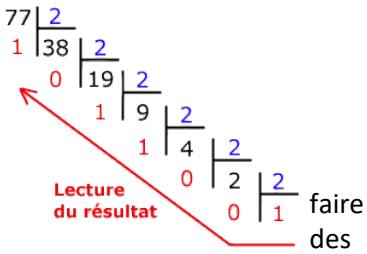
Coefficient par lequel on multiplie la puissance 16	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Symbole utilisé en base hexadécimale	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

438 en B10 équivaut à : $1x16^2+ 11x16^1+6x16^0$
Valeur en B10 : 256 176 6
représentation de 438 en base hexadécimale→ 1 B 6
Voici 5 nombres en base 10 : 5, 16, 57, 548 et 3852. Représentez-les en base 16 (Aide possible avec un tableau ou un tableur)

Il existe de nombreux autres « codages ». Tous fonctionnent sur le même principe.
Alors POURQUOI avoir choisi le binaire pour coder les nombres en informatique ?
A retenir: le codage binaire a été choisi car il est facile de faire lire des trous ou des pleins à des machines… Cela correspond aussi à des signaux lumineux ou électriques (0: pas de courant ou pas de lumière/1: on enregistre un courant ou une lumière).

D- traduire pour communiquer/ Comment convertir d'une base à l'autre
Passer du décimal au binaire

Le nombre en base 10 est $2^6 + 2^3 + 2^2 + 2^0 = 64 + 8 + 4 + 1 = 77$.
Allons maintenant dans l'autre sens et écrivons 77 en base 2. Il s'agit de faire une suite de divisions euclidiennes par 2. Le résultat sera la juxtaposition des restes. Le schéma ci-dessous explique la méthode:



77 en base 10 s'écrit donc en base 2: 1001101.

Exercice 6 : vous allez écrire l'algorithme (en français) qui permet de passer du décimal au binaire.
Pour aller plus loin: on peut à partir de là écrire un programme python s'effectuant sur un nombre entré par l'utilisateur. Voici des éléments qui pourraient vous servir:
L'instruction **//** donne en python le quotient d'une division euclidienne
L'instruction **%** donne le reste d'une division euclidienne.
L'instruction **reverse** exécutée sur une liste vous permet d'inverser les éléments d'une liste, mais elle ne fonctionne pas sur une chaine de caractère.

II. Les données sont structurées

1. Les tables

La structure de *table* permet de présenter une collection: les objets en ligne, les descripteurs en colonne et les données à l'intersection. Les données sont alors dites structurées.
Pour assurer la persistance des données, ces dernières sont stockées dans des fichiers. Le format CSV

(Comma Separated Values, les données avec des séparateurs) est un format de fichier simple permettant d'enregistrer une table.

2. Les métadonnées

À toute donnée ou tout fichier de données sont associées des *métadonnées* qui permettent d'en décrire le contenu. Ces métadonnées varient selon le type de fichier (date et coordonnées de géolocalisation d'une photographie, auteur et titre d'un fichier texte, etc.). Les données comme les métadonnées peuvent être capturées et enregistrées par un dispositif matériel ou bien renseignées par un humain. Elles sont de différents types (numériques, textes, dates). Certaines collections typiques sont utilisées dans des applications et des formats standardisés leur sont associés: par exemple le format ouvert vCard (extension .vfc) pour une collection de contacts.

3. Les bases de données

Lorsqu'une collection complexe utilise les éléments d'une autre collection, il est intéressant de lier ces deux collections pour simplifier la structure de la *base de données* et ainsi:
- Eviter les doublons dans la base,
- ajouter/modifier/supprimer à un seul endroit lorsque I'on a besoin d'ajouter/modifier/supprimer une donnée,
- éviter les erreurs.

Une base de données (ou database, ou BDD par commodité) regroupe ainsi plusieurs collections de données reliées entre elles. Les informations y sont organisées afin d'être facilement consultables, gérables et mises à jour.
Elles sont indexées afin de pouvoir facilement trouver les informations recherchées à l'aide d'un logiciel informatique. Chaque fois que de nouvelles informations sont ajoutées, les données sont mises à jour, et éventuellement supprimées, toutes les collections liées entre elles sont mises à jour.
Une base de données est représentée dans un format spécifique, faisant apparaître les collections (sous forme de tables) et les liens (appelées relations) entre ces collections.
Par exemple, la base de données d'une bibliothèque conserve les données sur les livres, les abonnés et les emprunts effectués.
Pour aller plus loin: visitez les centres de données Google en cliquant sur ce lien
voir les deux vidéos du dossier

4. Enregistrer, accéder aux données et les modifier....

Activité 02-02 manipuler des données.

Partie 1: créer un fichier de données, le modifier, le lire: la BASE!

1- Ressources: Diaporama

Partie 1 : créer un fichier de données, le modifier, le lire: la BASE!

1- Les instructions de base

a- conseil de sioux: **créer un répertoire courant de travail à accès facile** pour le fichier à travailler:

```
import os  #on importe le module nécessaire pour travailler les fichiers et dossiers(=répertoires)
os.chdir("C:/tests python")  # indique le répertoire existant de travail courant 'tests pythons' à la racine de C:
print(os.getcwd()) # pour vérifier qu'on a bien changé le répertoire de travail courant...
```

b- Pour **nommer un fichier** dans python, il faut décrire le **chemin absolu** pour s'y rendre: exemple

```
C:/mon_dossier/prog_python/mon_fichier.txt
```

c- Il faut toujours ouvrir le fichier selon un mode d'ouverture particulier(lecture 'r', écriture 'w', ajout 'a'):

```
mon_fichier = open("fichier.txt", "r") #on affecte le fichier ouvert à une variable mon_fichier pour la LIRE facilement
print(mon_fichier) # pour indiquer les caractéristiques de ce fichier
<_io.TextIOWrapper name='fichier.txt' encoding='cp1252'>
```

Rq: 'r': le fichier doit être existant, 'w' le contenu du fichier est écrasé, 'a' le contenu n'est pas écrasé, on peut le modifier. Dans les cas de 'w' et 'a', si le fichier n'existe pas, il est créé.

On peut ensuite affecter le contenu de notre fichier à une variable, la modifier, l'afficher....
contenu = mon_fichier.read() # affecte à la variable contenu le contenu du fichier
print(contenu) # affiche le contenu du fichier(en fait de la variable mais c'est la même chose ici…)
mon_fichier.close() # pour fermer le fichier et **le rendre disponible aux autres applications(indispensable!!!).**

Alternative plus rapide mais moins pédagogique...:

```
with open('fichier.txt', 'r') as mon_fichier:
      texte = mon_fichier.read()
      print(texte)
```

→ complétez votre mémo python des nouvelles instructions utilisées....
open("fichier.txt", "r"), open("fichier.txt", "w"), open("fichier.txt", "a") ; close("fichier.txt") ; with....
la façon d'adresser un fichier F:/dossier1/dossier1,1/dossier1,1,1/fichier.txt

2- Exercice d'application:
En utilisant l'interface python:
- Dans un dossier "mes fichiers python" situé à la racine de votre clef, créer un fichier .txt vide
- afficher son contenu
- le fermer(vous essaierez ensuite sans le fermer pour constater le conflit...).
- ouvrez le avec un traitement de texte... et inscrivez y votre NOM et votre prénom et un mot secret, enregistrez, fermez le.
- afficher son contenu

correction:

```
import os   #on importe le module nécessaire pour bouger les fichiers
print(os.getcwd())
os.chdir('K:/mes fichiers python') #on indique le répertoire de travail
#courant tests pythons à la racine de  votre clef...(il faut le créer avant!!)
print(os.getcwd())  # vérifier qu'on a changé le répertoire de travail courant...


#partie 1 créer un fichier dans le répertoire de travail courant
mon_fichier = open("fichier.txt", "w") #on affecte le fichier ouvert à une
#variable mon_fichier pour la LIRE facilement
#ou
mon_fichier = open("fichier.txt", "a")#on affecte le fichier ouvert à une
#variable mon_fichier pour la LIRE facilement
mon_fichier.close() # pour fermer le fichier et le rendre disponible aux autres applications.

mon_fichier = open("fichier.txt", "r")
contenu = mon_fichier.read()
print(contenu)
mon_fichier.close() # pour fermer le fichier et le rendre disponible aux autres applications.
```

III. Les opérations sur les données

1- Créer et modifier un fichier texte.

Activité 02-02 manipuler des données.

Partie 2: opérations sur les données

1, Quelques opérations de base à effectuer sur les données.

1- Un exemple: créer un fichier texte liste de course....

Ressources:

liste.write(variable à ajouter) #ajoute la variable dans le fichier 'liste', ici ce sera une chaîne de caractères(str)

liste.write('\n') #va à la ligne

l'instruction '{}'.format() → print('{}, c'est à présent ton essai n{}'.format(NOM, i)) # permet d'afficher JEAN RENE, c'est à présent ton essai n°5 si NOM= JEAN RENE et i = 5....

2- Exercice d'application:

En utilisant l'interface python:

Niveau 1:

- Créer une programme python qui vous permettra de dresser une liste de course lisible enregistré dans un fichier 'listedecourse.txt'

- ouvrir le fichier 'liste de course'

Niveau 2:

- Créer une programme python qui demande à l'utilisateur:

- combien de produits il désire mettre sur sa liste de course

- quels sont ces produits

- et que ce programme crée un fichier 'liste de course.txt' dans lequel on trouve la liste.

- ouvrir le fichier 'liste de course'

Niveau ultime:

Niveau 2 + le programme doit dresser la liste afin que les produits soit numérotés alors que l'utilisateur n'a entré que le nombre de produits total et les nom de chacun.

exemple de rendu au sein du fichier quel que soit le niveau:

Ma liste de course
1- œufs
2- viande
3- un arbuste pour le jardin

A faire:

On peut donc créer un fichier qui renferme des données afin de les conserver pour les consulter(read) ou les modifier(append) --> Allez jeter un coup d'œil au fichier créé, complétez votre mémo python des nouvelles instructions utilisées....

fichier.write(' ')

fichier.read()

print('{}e de la course mixte et {} garçon'.format(position mixte, position garçon))

2- Créer et modifier des listes/tables: (HPorg)

1-Très utile pour stocker les bases de données, on peut créer des tables, les variables vont donc être définies par des descripteurs. Le plus compatible est fichier de type csv.

Ressources:

Liste=[variable1, variable2, variable3] # on affecte à Liste une liste de trois variables.

Si on a une chaîne de caractères de type chaîne='NOM, PRENOM, AGE'

Liste=chaine.split(',') # permet de transformer la chaîne en liste de 3 variables séparées par des virgules.

ListeTot=[[liste1],[liste2],[liste3]] # permet de créer une liste de liste et de générer un tableau lisible par un tableur.

Variable=m[j][i] #ressort l'élément i dans la liste j de la liste de liste m.

Grrrrrr.insert(*i*, *x*) #Insère x en i dans la liste Grrrrrr, ajoute un élément dans la liste à une position précise(décale le reste!!!)

Pour créer un fichier csv, c'est comme pour un txt mais il faut réfléchir à la structure:

a. lors de la lecture, chaque ligne pourra être transformée une liste par python à condition de séparer les éléments par un caractère précis(une virgule, un point virgule, une tabulation…), on se servira de la fonction split(« ? »)

b. penser, si l'on veut représenter un tableau, à renseigner la première ligne comme intitulés de colonnes.

c. Il faut que chaque ligne possède le même nombre d'éléments et qu'ils soient rangés dans le même ordre que les intitulés.

Un exemple:

```
f=open('test.csv', 'w') # lien vers un fichier on l'affecte la variable f en écriture, on écrase le contenu
f.write('intVar1,intVar2,intVar3\n') #on crée les intitulés et on va à la ligne
f.close() #on ferme dans la config 'w'

f=open('test.csv', 'a') # lien vers un fichier on l'affecte la variable f en ajout, on écrase pas
f.write(var1+','+var2+','+var3+'\n') #on ajoute à la liste 3 valeurs de variables var1 à 3 et on va à la ligne
f.close() #enferme dans la config 'a'
```

Très facultatif, à ne pas traiter.

Pour travaillller un fichier csv, on peut importer un module(bibliothèque) csv sous python:

```
import csv
f=open('test.csv', 'w') # lien vers un fichier on l'affecte la variable f
fichierscv=csv.reader(f) #lit le contenu du fichier et l'affecte à la variable 'fichiercsv'
for ligne in fichiercsv:
        print(ligne)
table = csv.writer(open("MONFICHIER.csv", "wb"))
fichierscv=csv.reader(f) #lit le contenu du fichier et l'affecte à la variable 'fichiercsv'
for ligne in fichiercsv:
        print(ligne) #affiche toutes les lignes du contenu du fichier
```

exemple:

contenu du fichier

Scores des joueurs :
Alice : 257692
Bob : 199827

```
f = open("scores.txt", "r")
f.readline() # on saute la première ligne
t=f.readline()
```

```
print(t)
t = f.readline()
print(t)
f.close()
```

Résultat :
Alice : 257692
Bob : 199827

2- Exercice d'application:
Niveau 1:
Créer à présent un programme permettant de créer, en interaction avec un utilisateur, une base de donnée concernant ce client.
Le client devra renseigner son nom, prénom, son sexe(M ou F), son age, son adresse mail, sa pointure de chaussure. Le fichier se nommera basededonneesclients.csv.

Niveau 2:
Créer à présent un programme permettant de créer, en interaction avec X(à demander) utilisateurs successifs, une base de donnée clients.
Chaque client devra renseigner son nom, prénom, son sexe(M ou F), son age, son adresse mail, sa pointure de chaussure. Le fichier se nommera basededonneesclients.csv.

3- Rechercher, filtrer, trier ou calculer sur une ou plusieurs tables des listes/tables:
1- La recherche dans des données structurées a d'abord été effectuée selon une indexation préalable faite par l'homme lorsqu'il crée ses listes ou tables. Des algorithmes peuvent ensuite d'automatiser l'indexation à partir de textes, d'images ou de sons. Une table de données peut faire l'objet de différentes opérations: rechercher une information précise dans la collection grâce aux descripteurs, trier la collection sur une ou plusieurs propriétés, filtrer la collection selon un ou plusieurs tests sur les valeurs des descripteurs, effectuer des calculs, mettre en forme les informations produites pour une visualisation par les utilisateurs.

Objectifs du projet: lire un fichier csv afin de trier son contenu et d'utiliser une partie des données qu'il comporte.
Ressources : *(tout n'est pas forcément utile...)*
votre mémo Python.
list.append(*x*) **#**Ajoute un élément à la fin de la liste 'list'
list.insert(*i*, *x*) **#**Insère un élément à la position indiquée. Le premier argument est la position de l'élément courant avant lequel l'insertion doit s'effectuer
list.remove(*x*) **#**Supprime de la liste le premier élément dont la valeur est égale à *x*
list.clear() **#**Supprime tous les éléments de la liste
list.index(*x*[, *start*[, *end*]]) **#**Renvoie la position du premier élément de la liste dont la valeur égale *x*
list.count(*x*) **#**Renvoie le nombre d'éléments ayant la valeur *x* dans la liste.

2- Exercice d'application:
Nous travaillerons sur le fichier nat2018.csv qui recense tous les prénoms nés en France, par année depuis 1900.
Votre programme python devra demander le prénom que vous recherchez et afficher les lignes correspondantes au prénom ainsi que la somme de toutes les naissances sous ce prénom depuis 1900.

Aide Pas à pas:

a- On l'invente pas, ce fichier nécessite un traducteur, codec, commencez le programme ainsi:
import codecs # le fichier nécessite des traducteurs, il est codé en UTF-8
monfichier=codecs.open('K:/nat2018.csv','r','utf-8') #on affecte la variable 'monfichier' du fichier à travailler
b- Il va falloir séparer cette immense chaîne de caractères en listes: d'abord une liste par ligne, puis pour chaque ligne, la transformer en liste qui comprend autant de variables que de colonnes.

c- Il va falloir faire autant de boucles qu'il y a de lignes(moins l'entête des intitulés)
Mon programme fait 14 lignes et fonctionne à merveille, alors courage !!!

Autre idée...

Jouer et enregistrer ses scores:

Etape 1: créer le jeu "tu brules, tu refroidis":
Un nombre de 1 à 9999 est tiré au sort par l'ordi, le joueur doit le retrouver en limitatn le nombre d'essais. LA seule aide de l'ordi est la suivante: il dit "tu te réchauffes" lorsque la proposition est plus proche du nombre mystère que le précédent essai, "tu refroidis" dans le cas inverse et "t'avances pas là!" lorsqu'on est aussi loin...
Une fois le nombre trouvé, l'ordi annonce la victoire, donne le nombre mystère et le nombre d'essais qu'il a fallu au joueur pour le trouver.

Etape 2: faire en sorte que les scores soient gardés
Ensuite, l'ordi demande le nom du joueur et l'enregistre, associé à son score(nombre d'essais) dans un tableau de type csv où s'accumule tous les scores des joueurs.

IV- Le stockage des données.

1, Les supports

Les fichiers de données sont stockés sur des supports de **stockage: internes (disque dur ou SSD) ou externes (disque dur , clé USB, CD et cartes mémoires)**
Le **disque dur** est constitué de **disques magnétiques** qui tournent en continu à très grande vitesse et permet de stocker des quantités importantes de données (de façon standard de 1 à 2 To, pour info : 1 téra-octet (To) = 2^{10} giga-octets = 2^{40} octets).
La **clef USB** est très facile à transporter, utilisant des technologie de **mémoire flash**, particulièrement rapide mais c'est un support peu fiable (manipulé souvent, de qualité variable et dont la durée de vie est relativement courte). Sa capacité de stockage est assez faible.
Le **SSD ou Solid State Drive** utilise la **même technologie que la clef mais est un dispositif de stockage interne aux machines**. Du fait de la rapidité de l'écriture et de la mobilisation des données qui y sont stockées; il va être l'emplacement de prédilection des données fréquemment mobilisées, comme celles du système d'exploitation.
Tous ces supports pouvant subir des dommages matériels entraînant des altérations ou des destructions des données, il est nécessaire de réaliser des sauvegardes multiples.

Stocker des données implique de faire un choix entre stockage local (sur un ordinateur local, sur **un serveur local mais commun à plusieurs machines**) ou distant (d**es serveurs externes ou le *cloud***).
Les plus grandes bases de données sont souvent implémentées sur des serveurs dédiés (machines puissantes avec une importante capacité de stockage sur disques) et souvent distants dans ce que l'(on nomme des **Data Center.**

Les critères importants pour faire ce choix sont ceux de l'accessibilité des données et de leur sécurité. Des solutions locales assurent l'accessibilité mais ne sont pas toujours les plus sûres (fait-on assez de sauvegardes, est-on protégé contre les attaques informatiques, les pannes électriques, etc.?). Les solutions distantes posent d'autres questions.
Notamment les centres de données doivent être alimentés en électricité en continu et maintenus à des températures suffisamment basses pour fonctionner correctement. Ainsi le secteur numérique dans sa globalité (fabrication et utilisation de serveurs, réseaux, terminaux, etc.) représente à lui seul pas moins de 10% de la consommation énergétique mondiale. Avec le développement du cloud, les besoins en hébergement de données augmentent régulièrement. Conséquence directe : d'ici 2020, la France devrait posséder au moins 200 centres de stockage de données qui sollicitent de grosses ressources énergétiques pour leur alimentation comme pour leur refroidissement (3 TWh en 2015, soit l'équivalent de la ville de Lyon, selon l'Union française de l'électricité). L'impact écologique du numérique est massif et doit probablement être repensé.

2, La prolifération des données

L'évolution des capacités de stockage, de traitement et de diffusion des données fait qu'on assiste aujourd'hui à un phénomène de surabondance des données. Les objets connectés (nos téléphones, les montres, les ampoules, les voitures, les lampes , les frigo et autres capteurs multiples) génèrent de façon automatique des millions de données.
Cela exige le développement de nouveaux algorithmes capables de les exploiter et des solutions pour les stocker: Les bases de données structurées sont souvent inopérantes pour des données dont l'évolution est aussi rapide.
L'exploitation de données massives (Big Data) est en plein essor dans des domaines aussi variés que les sciences, la santé ou encore l'économie. Les conséquences sociétales sont nombreuses tant en termes de démocratie, de surveillance de masse ou encore d'exploitation des données personnelles. Certaines de ces données sont dites ouvertes (OpenData), leurs producteurs considérant qu'il s'agit d'un bien commun.
Néanmoins, il existe aussi un marché de la donnée où des entreprises collectent et revendent des données sans transparence pour les usagers. D'où l'importance d'un cadre juridique permettant de protéger les usagers, préoccupation à laquelle répond le règlement général sur la protection des données (RGPD), texte réglementaire européen qui encadre le traitement des données de manière égalitaire sur tout le territoire de l'Union Européenne. Il est entré en application le 25 mai 2018.
Le RGPD s'inscrit dans la continuité de la Loi française Informatique et Libertés de 1978 (et dont l'application est surveillée par la CNIL, commission nationale de l'Informatique et des Libertés). Il a été conçu autour de 3 objectifs :

- renforcer les droits des personnes
- responsabiliser les acteurs traitant des données
- crédibiliser la régulation grâce à une coopération renforcée entre les autorités de protection des données.

Tout acteur du numérique opérant sur le territoire de l'Union ou de l'étranger mais proposant des services à des résidents européens doit s'y conformer.

Contenus	Capacités attendues
Données	Identifier les principaux formats et représentations de données.
Données structurées	Identifier les différents descripteurs d'un objet. Distinguer la valeur d'une donnée de son descripteur. Utiliser un site de données ouvertes, pour sélectionner et récupérer des données.
Traitement de données structurées	Réaliser des opérations de recherche, filtre, tri ou calcul sur une ou plusieurs tables
Métadonnées	Retrouver les métadonnées d'un fichier personnel.

Données dans le nuage (cloud)	Utiliser un support de stockage dans le nuage. Partager des fichiers, paramétrer des modes de synchronisation. Identifier les principales causes de la consommation énergétique des centres de données ainsi que leur ordre de grandeur.

Notions fondamentales et lexique

Données, descripteur, collection, base, binaire, hexadécimal, table, métadonnée, base de donnée, indexation, tri, CD, disque dur, SSD, clé USB, serveur, *cloud* ou nuage, Datacenter ou centre de données, Big Data ou données massives, RGPD.