**Proposition de séquence pédagogique NSI Première Cours Hattemer**

**Document rédigé par Mr OBELE MVOMO**

**Thème : représentation des données : types construits**

**Sous-thème : Dictionnaire par clés et valeurs**

**Pré requis :**

* Les bases d’algorithme et de programmation Python
* Notion de listes et de fonctions

**Capacité attendues :**

* Construire une entrée de dictionnaire
* Itérer sur les éléments d’un dictionnaire

**Situation :**

On effectue un sondage au sein du groupe d’élève de NSI pour connaitre le style de musique préféré des élèves parmi :

Pop, rock, électro, jazz, reggae, urban… (la liste est ouverte pour d’autres propositions éventuelles).

Remarque : attention un seul choix est possible !

1. Q1 Donner la définition d’un dictionnaire (en programmation) (recherche par les élèves)

Synthèse de l’enseignant

Ecriture d’un dictionnaire ; Notion clés et de valeurs

Les dictionnaires sont des collections non ordonnées d’objets ; il n’y a pas d’index et l’ordre dans lequel les objets apparaissent n’est pas important. Les dictionnaires sont représentés entourés par des accolades {} avec pour chaque terme une données sous la forme clé :valeur dic={clé :valeur, …}

dic\_vid={}

Notion de clé unique dans un dictionnaire de types pour les clés et les valeurs…

1. Q2 : Entrer le dictionnaire suivant en mode console.

StyleMusic={« pop » :valeur, ………………}

1. Q3 : Tester les commandes suivantes dans la console. Expliquer leur rôle, leurs fonction.

|  |  |
| --- | --- |
| Comande | Son rôle, sa fonction |
| styleMusic[« pop »] |  |
| styleMusic[« rap »] |  |
| StyleMusci[« rap »]=2styleMusic |  |
| styleMusic.keys() |  |
| styleMusic.values() |  |
| styleMusic.items() |  |

1. Q4 : Ecrire une fonction effectif(dic) qui étant donné un dictionnaire, renvoie la somme des valeurs.

effectif(styleMusic) devra renvoyer effectiftotal+2

1. Q5 : Ecrire une fonction nbCle(dic) qui renvoie le nombre de clés du dictionnaire.

nbCle(styleMusic) devra renvoyer 8.

1. Q6 : Encire une fonctionajout(dic, cle, valeur) qui ajoute un couple (musique,1)
   1. Attention 🡪 Valeur du couple augmenté de 1 si la musique déjà présenté dans le dictionnaire
      1. Ajout d’une nouvelle clé si la musique n’est pas présente
   2. Tester cette fonction avec l’ajout des musiques préférées des 2 enseignants (choix des enseignants donnés aux élèves sur une demande).
   3. Ecrire une fonction stylePreference(dic) qui renvoie le style de la classe
   4. Tester cette fonction

**Chapitre I :**

Def m :

Toute machine informatique ne manipule que des représentations de données : des suites formatées de de bis (contraction de binary digits). C’est-à-dire contractions de 0 et de 1. Les différents types de données (nombres entiers (python : integer) ou réel (python : flotants)). Chaînes de caractères, valeurs booléennes, images, son) sont donc représentés selon un codage dépendant de leur nature. Le codage des images et des sons ne sera pas étudié dans ce cour. Nous nous limiterons dans une première partie au nombre des caractères et dans une seconde partie au fonctions logiques.

1) Représentation des nombres entiers

a) écriture d’un nombre entier dans une base quelconque

**définition 1.1 :** Ecriture d’un entier en base A. a étant un entier supérieur ou égal à 2, un entier positif quelconque n s’écrit en base a : à l’aide des -a chiffres inhérents à sa base sous la forme d’addition des puissances successives de cette base multiplié par des coefficients ak<a :

* Décomposition :

n=a0\*a0+a1\*a1+a2\*a2+…+ap\*ap.

* Écriture condenses de:
* n=ap …. .a2a1a0.
* Écrire 459(10) (10)⬄sur une base de 10. Il faut l’écrire sous forme décondensée.

(r)écriture d’un entier en base a.

n=a0\*a0+a1\*a1+a2\*a2+…+ap\*ap.

n=ap…a2a1a0 est indicée n=ap…a2a1a0(a)

condensée

Remarque : Les chiffres inhérents à la base a<10 sont les chiffres ovales de 0 à a-1. En base 8, par exemple, les chiffres sont 0 ;1 ;2 ;3 ;4 ;5 ;6 ;7.

Les chiffres binaires (a=2) sont : 0 et 1. Pour une base au>u10, on ajoute les lettres ont (pour 10), b (pour 11) etc…

Ainsi en base de 16, on utilise les chiffres  : 0 ;1 ;2 ;3 ;4 ;5 ;6 ;7 ;8 ;9 ;a ;b ;c ;d ;e ;f

La notation de la base utilisée peut se faire de différentes façons mais nous utilisons ici la notation indicée, la base étant indiquée en indice et entre parenthèse.

Quand le contexte rend cette notation inutile (la base étant indiquée dans le texte) On la supprime.

Ainsi, On écrira 459(10); 110011001(2) et 1af(16) les nombres qui s’écrivent en base dix 459, 409 et 431. En python un nombre binaire (base deux se préfixe par ’ob’ et un nombre hexadécimal (en base 16 se préfixe ‘0x’ ; ainsi en python 110011001(2) et 1af(16) se notent ob 110011001 ox 1af.

Ce type de notation s’impose du fait de l’écriture linéaire des instructions. La décomposition qui explique la valeur d’un nombre s’écrit généralement en base dix. Ainsi dans les trois exemples précédents, nous avons :

1. 459(10) :
   1. 459=9\*100+5\*101+4\*102=9+50+400= 459
2. 110011001(2) :
   1. 110011001=1\*20+1\*23+1\*24+1\*27+1\*28=1+8+16+128+256=409
3. 1af(16) :
   1. 1af=15\*160+10\*161+1\*162=15+160+256=431

Méthode : Traduction décimale d’un nombre donné en base a.

La méthode dérive de la définition : on utilise la décomposition selon les puissances de la base. Le chiffre le plus à droite représente les unités, ensuite le deuxième chiffre représente les paquets de « a » unités (diraines en base dix, deuzième en base deux, troisième en base vingt…). Le troisième chiffre représente les paquets de a² unités (centaine en base dix, ; quatraine en base deux, quatre centaines en base vingt…) etc.

Les termes ak\*ak ne sont pas écrits lorsque le chiffre ak est nul ( le but étant de singulier l’écriture décomposée)

Remarquons que, comme les nombres 999…9 en base dix, il est facile de connaitre\savoir la valeur de bbb…b en base b+1.

Exemples :

999(10)=103-1 ; 666(7)=73-1=342 ;111111111(2)=29-1=512-1=511 ; ffff(16)=164-1=65536-1=65535

Remarquons une autre caractéristique du système décimal qui se transpose aisément à toutes les bases. Lorsqu’on multiple un entier décimal par dix, on ajoute un zéro à droite.

Transposé à une base quelconque, cette propriété s’énonce ainsi : lorsqu’on multiplie par a un entier écrit en base a, on ajoute un 0 à droite. Ainsi, on a :

* 6660(7)=6\*101+6\*102+6\*103=~~60+120+180=360~~ 7\*342=2394 Car 666=73-1=342
* 111 111 111 000(2)=23\*511=8\*511=4088
* ffff0(16)=1\*161+2\*162+3\*164+0\*165=16+….=… =16\*65535=1048560

ffff0(16)=fffff(16)-f

= 165-16

= 1048576-16

= 1048560

En python, la conversion décimale s’écrit int(‘n’,a) où ‘n’ est le nombre écrit sous la forme d’une chaîne de caractères et a est la base. Par exemple : int(‘ob110011001’,2) renvoie 1\*20+1\*23+1\*24+1\*27+1\*28 = 1+8+16+128+256=409

int(‘Ox1af’, 16) renvoie 431.

Pour la transposition inverse, afin de déterminer le nombre de paquets de a unités, on commence par effectuer une division euclidienne par a ( conduisant à un quotient et à un reste entier).

Le reste de cette division est le chiffre des unités, le quotient est le nombre de paquets de a unités qu’il convient de diviser à nouveau par a s’il atteint ou dépasse a (car le plus grand chiffre utilisable en base a à pour valeur a-1).

* Comment s’écrit 100 en base 16 ?
  + 100(16)=100/16=6\*16+4, or 6<16, donc 100(10)=64(16)
* Comment s’écrit 1000 en base 16 ?
  + 1000(16)=1000/16=3e8(16)

(r)Méthode (Traduction en base a d’un nombre donné en base dix) (/r)

On effectue la cascade des divisions euclidiennes par a décrite sur l’illustration précédente, cette cascade s’arrêtant lorsque le quotient obtenu est nul, il suffit de (r)remonter la suite des restes(/r) à l’envers pour obtenir l’écriture condensée en base a. Cette démarche algorithmique se programme aisément :

* On entre les nombres n et a et on obtient l’écriture Condensée en base a
* Une restriction s’impose la base ne doit pas dépasser 36 (les dix chiffres arabes et les 26 lettres minuscules), mais le programme ci-contre en python ne teste pas la valeur de l’entrée.

def conversionBase ():

reste,quotient,num,ecriture=list(),nb,nb,""

while quotient>0:

quotient=num//base

rest.append(num%base)

num=quotient

for j in range(len(reste)): ecriture=chiffre[reste[j]]+ecriture

return ecriture

chiffre=["0","1","2","3","4","5","6","7","8","9","a","b","c","d","e","f","g","h",\"i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z"]

nb=int(input("Entrer le nombre (en base de 10):"))

base=int(input("Entrer la base (en base de 10):"))

print("Le voici écrit dans cette base: ()".format(conversionBase()))

en python toujours, la conversion du décimal n au binaire s’écrit « bin(n) » et la conversion du décimal n à l’hexadécimal s’écrit hex(n). Ce sont des chaînes de caractères qui sont renvoyées. Par exemple « bin(100) » renvoie la chaine « 0b1100100 » et « hex(1000) » renvoie « 0x3e8 ».

ex 1 : 12,87510 = 1\*101+2\*102 | 8\*101+7\*102+5\*103 = 10+200 | 80+700+5000 = 210,5780

ex 2 1100,1112=1\*21+1\*22 | 1\*21+1\*22+1\*23 = 2+4 | 2+4+8 = 6,14

ex 3 :

1. Bin en hex : 1111|0000|111|000|11|00|102 = 0x9a2ea63cd6b643ba
2. En bin : abcdef = 10+11+12+13+14+15 = 75
3. En hex : 111|000|111|000|111|000|1112=0x6046f9cd9ce8baa2f

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 1100 | 0111 | 0001 | 1100 | 0111 |
| ||  ||  \ / | ||  ||  \ / | ||  ||  \ / | ||  ||  \ / | ||  ||  \ / | ||  ||  \ / |
| 0001 | c | 7 | 1 | c | 7 |
| ||  ||  \ / |  | | | | |
| 1 |

(r)codage sur p bits (/r)

Les informations manipulées par un ordinateur sont des mots binaires de taille fixe, souvent multiple de 8 bit. 8 bit correspond à 1 octet

Sur p bits on peut stoker des entiers positifs entre 0 et 2p-1.

Les entiers codés sur un octet sont donc compris entre 0 et 28-1=255.

Avec 2 octets, on va jusqu’à 216-1=65534, sur 4 octets (32 bits) : 4294967295.

Sur 8 octets (64 bits) jusqu’à : 1,84…1019.

En python, version 2 … . Il existait deux sortes d‘entiers à savoir les int : qui allaien jusqu’à 2 147 483 648 ( correspond environ à la moitié du codage sur 4 octets) les longs au-delà. Dans python 3, la taille des entiers n’est plus limitée (pour de très grands nombres) La mémoire de l’ordinateur va être le seul facteur limitant.

En Java, on a quatre types d’entiers : le byte (1 octet), le short(2 octets), le int (4 octets), le long (8 octets).

Le Transtypage (changement de type) est automatique quand la taille de l’entier manipulé augmente, mais dans l’autre sens, comme il peut avoir une perte d’information, elle doit-être provoquée par une instruction de cast (forcer).

En hex : D47= 13\*162+4\*161+7\*160= 3399

3399=1\*211+1\*210+1\*28+1\*26+1\*22+1\*21+1\*20 = 1\*2048+1\*2024+1\*256+1\*64+1\*4+1\*2+1\*1= 110 101 000 111(1)

Conversions binaire / octal / hexadécimal

Les nombres binaires manipulés par l’ordinateur étant très long, on trouve souvent avantageux à les convertir en hexadécimal (en base 16) ou parfois en octal (en base 8). Les chiffres hexadécimaux se codent sur 4 bits (un quartet) car le plus grand chiffre hexadécimal s’écrit avec 4 bits. f(16)=15(10)=1111(2) car 1\*23+1\*22+1\*21+1\*20 = 8+4+2+1

Convertir du binaire en octal (base 8) : 111 000 111 000 111 000 111(2)=7070707(8)

Python écrit les nombres octaux avec le préfixe (r)Oo(/r) et la conversion en octal est prévue par les fonctions oct(n) (du décimal vers l’octal) ; et int(‘Oon’, 8) de l’octal vers le décimal.

Par exemple oct8 donne ‘(r)0à10’(/r)

int(‘0o777’, 8) donne 7\*8²+7\*81+7\*80 = 7\*64+7\*8+7\*1 = 448+56+7=511.

[https://Repl.it/@L33tHax0rz/conversionbase](Https://Repl.it/@L33tHax0rz/conversionbase)

<http://Repl.it/repls/CriminalPricklyExponent>

<https://repl.it/@L33tHax0rz/conversionBase>

(r)Application de l’octal :(/r)

Les droit d’accès sur linux définissent les action qu’un utilisateur à le droit d’effectuer sur un fichier :

* R pour read (lecture)
* W pour write (écrire)
* X pour excecute (exécuter), selon qu’il est propriétaire du fichier, membre du groupe propriétaire du fichier, ou ni l’un ni l’autre.
* Chacun des 9 droits d’accès est stocké sur 1 bits :
  + Si le bit vaut 1 le droit correspondant est accordé
  + S’il vaut à le droit correspondant est refusé.

rwxr-xr—indique que le propriétaire du fichier à tout les droits.

La représentation binaire de ces droits, s’écrit 111 101 100 et linux admet une notation octale (base 8). Qui tient sur 3 chiffres est 754.

Partie 2 : Programmation

Objectifs :

Les deux objectifs du programme, (type construits et traitement de donnée en table) seront traité en rappelant les principes de base.

Introduction au fonctionnalités de base de python 3.3. … (Affectations, variables, séquences d’instruction, instruction conditionnelle, boucle bornée et non bornée, définition et appel de fonctions).

Représentation de données (type construits) : les p-uplets (tuples) p-uplets nommés, tableau indexé, tableau donné en compréhension dictionnaires par clés et vecteurs.

Traitement de donné en table : descripteur, notion de base de données (tableau doublement inversé ou tableau de p-uplets) :

* Indexation de tables
* la recherche dans une table
* le tri d’une table
* fusion d’une table

Plan du cour :

1. Introduction à la programmation en python :
2. Séquence conditionnelles et boucles
3. Type de données construits
4. Fonctions itératives et récursives

1. Introduction à la programmation en python

Aperçus historique et généralités :

La langage python doit tout à Guido von Rossum (1956-…), son principal auteur et régulateur et il se retire du projet en 2018. Le nom python vient de la troupe monty python. La première version de python date de 1991, la version 0.9.0, la version 1.0 en janv 1994, version 2.0 2000, la version 3.0 décembre 2008, 2019, version 3.8.0 est travaillé par les développeurs. Le python est un langage de programmation interprété sur multiplateformes (win, mc, lin). Facile d’accès, riche de potentialité et opensource (libres de droits et gratuit. Il est utilisé autant dans l’industrie et dans l’information (université et lycée). Le nom de l’éditeur IDLE (interpreted development Environment).

Qu’est-ce que IDLE : IDLE est une environnement de développement assez simple pour python.

Pour en apprendre plus : [RDV ICI](https://docs.pyhon.org/3/library/idle.html)

Le mode interactif :

Le Shell de python (la Comande) se présente avec une invite des commandes à trois chevrons. (>>>).

* Les opérations usuelles peuvent être effectuées dans ce mode comme dans une calculatrice :
* Addition : >>> 2+3 retourne 5 (un entier) ;
* Soustraction : >>> 4.5-0.5 retourne 4.0 (un flottant repéré par le point décimal) ;
* La multiplication : >>> 10\*5 retourne 50, >>> 10\*0.5 retourne 5.0 ;
* La division : >>> 10/5 retourne 2, >>> 10/4 retourne 2.5, -10/3 retourne -3.3333333333333335 (petit problème d’arrondi).
* Division entière : (partie entière de la division) : >>> 10//4 retourne 2, >>> 10//3 retourne 3, avec un quotient t négatif, >>> -10//4 retourne -3 (la valeur entière planchée) et non (-2)
* Le reste de la division entière : >>> 10%5 retourne 0, >>> 10%3 retourne 1, >>> 5.3%2 retourne 1.2999999999999998.

Définition 2.1 (Variables) :

Une variable est un nom donné à une valeur. La déclaration du nom et la première affectation (est appelée une initiation) se font en python d’une manière simultanée. Une fois qu’une variable a été initialisée, elle peut être réutilisée dans un calcul.

Remarque : Un Identificateur de variable dénote un emplacement dans la mémoire dans lequel une valeur est stockée. Les noms de variables sont sensibles à la case : Les noms Eff et eff désignent deux variables différentes. Il faut choisir des noms significatifs (r)(préférer best\_player à bp)(/r) en évitant les accents et les caractères spéciaux. Minuscules, majuscules, chiffres (mais pas au début du nom) et le tiret bas/hui \_ uniquement. L’opération de base pour modifier la valeur d’une variable est l’affectation et s’écrit avec le signe (r)=(/r) (/!\ à ne pas confondre avec l’opérateur d’égalité dans un test) qui s’écrit en double signe égal : (r)==(/r)

Exemple : >>> a=4 (r)ne retourne rien car c’est une initialisation(/r)

>>> a=a+1 (r)(c’est une affectation on ajoute 1 la valeur de a et le résultat remplace cette valeur)(/r) a, retourne 5.

En python il est possible d’utiliser un raccourci pour effectuer une opération simple. Sur une variable.

(r)L’incrémentation ou la décrémentation a=a+1 peut s’écrire a+=1 et a=a-2 peut s’écrire a-=2.(/r)

Le même principe est valable pour les autres opérations : a=a\*3 peut s’écrire a\*=3. A=a//4 peut s’écrire a//=4 ; a=a%5 peut s’écrire a%=5 ; a=a=a+« ! » peut s’écrire a+=« ! ». (Cela ne donnera pas d’erreur seulement si a est de type string).

(r)Définition 2.2 : Les types (/r)

Les valeurs manipulées par python sont typées, les types de bases sont :

* (r)Les entiers sont notés int (integer) : (/r)
  + Sont des entiers relatifs sur lesquels portent les opérateurs +, -, \*, //, %, \*\*, etc…
* (r)Les flotants notés float :(/r)
  + Sur lesquels portent les opérateurs +, -, \*, /, etc …
* (r)Str : (/r)
  + Sur lesquels se trouvent des chaines de caractères constitués d’une séquence de chaine de caractères entres apostrophes ou guillemets, sur lesquels s’appliquent l’opérateur + (r) de concaténation (/r)
  + Concaténation : fait de relier deux chaines informatiques dans le but d’en créé une nouvelle.
* (r)Les booléens (True ou False) noté bool : (/r)
  + Sur lesquels portent les opérateurs : (r) and, or, not (/r) et qui peuvent-être retournées par des opérateurs de comparaison : (r) <, <=, >, >=, ==, != (/r)
* (r)L’utilisation du type flottant pour représenter les nombres réels n’est pas sans poser quelques problèmes de conversion (confer chapitre 1, problèmes de précision), le type enter est le plus sûre e, ce qui concerne les nombres, surtout en python car il n’y a pas de limites physique, mais il est limité. (/r)
* Pour le string python utilise l’ASCII ou L’UTF-8.
* Pour connaitre le type d’une variable, il suffit d’utiliser la fonction type.
  + (r)type() : >>> type(« 5 »)(/r)
  + Retourne le message <class 'str'>
  + Tandis que >>> type (5) retourne <class 'int'>
  + Tandis que >>> type (5.) retourne <class 'float'>
  + type(« 5 »)==str retourne (r)True(/r)

(r) b. Mode de programmation (/r)

Pour passer en mode programmation sur IDLE Il suffit de sélectionner new file. Et pour enregistrer il suffit de rentrer .py.

L’affichage avec print, le ou les résulrats de l’éxécution s’affichent dans la console si le programme contient au-moins une instruction utilisant la fonction print ().

Affichage avec print

Le(s) résultat(s) de l’éxécution s’affichen(nt) dans la console si le programme contient au moins une instruction utilisant la fonction print().

Le programme suivant va provoquer une écriture sur la console :

|  |  |
| --- | --- |
| a=3  b=a\*\*2  print (« le carré de », a, « est », b) | >>>  Le carré de 3 est 9 |
| *Programme test.py* | *Affichage console* |

Syntaxe de l’argument de print() : plusieurs variables ou constantes (pas nécessairement du même type) peuvent être affichées à la suite les unes des autres. Python intercale un espace entre ces valeurs et place à la fin par défaut, un caractère de retour à la ligne (\n).

Si on veut supprimer ce retour à la ligne, on doit utiliser l’option end=«  », placée à la fin de l’argument (end= «  » a pour effet d’insérer un espace à la place de \n).

Voici un 2eme programme qui exploite cette possibilité, et qui montre qu’une opération sur les variables peut être effectuée juste au moment précédent l’affichage :

|  |  |
| --- | --- |
| a=3  b=a\*\*2  print (« le carré de », a, « est », b, end= «  »)  print(« et son cube est, a\*b) | >>>  Le carré de 3 est 9 et son cube est 27. |
| *Programme test.py* | *Affichage console* |

Une autre possibilité pour formater un affichage : utiliser la méthode format() de la classe string en insérant un coupke d’accolades à la place d’une variable au sein d’une chaîne de caractères, la variable étant passée en argument de la méthode format(). Cela parait compliqué mais en réalité, c’est simple et proche de l’écriture réelle. Constatez-le par vous-même sur ce 3e exemple :

|  |  |
| --- | --- |
| a=3  b=a\*\*2  print (« le carré de {} est {} et son cube est {} ».format(a,b,a\*b)) | >>>  Le carré de 3 est 9 et son cube est 27. |
| *Programme test.py* | *Affichage console* |

**Les modules avec import :**

Les fonctions Python utilisées jusque-là sont représentées par défaut dans l’interpréteur : ce sont des fonctions built-in. La liste de ces fonctions est assez grande, puisqu’il y en a plus de soixante :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Abs | All | Any | Ascii | Bin | Bool | Bytearray | Bytes | Callable |
| Chr | Compile | Complex | Delattr | Dict | Dir | Divmod | Enumerate | Eval |
| Exec | Filter | Float | Format | Frozenset | Getattr | Globals | Hasattr | Hash |
| Help | Id | Input | Int | Isinstance | Issubclass | Iter | Len | List |
| Locals | Map | Max | Min | Next | Object | Oct | Open | Ord |
| Pow | Print | Property | Range | Repr | Reversed | Round | Set | Setattr |
| Slice | Sorted | Str | Sum | Super | Tuple | Type | Vars | Zip |

Bien sûr, nous n’allons pas ici les étudier toutes.

Aussi nombreuses que soient ces fonctions, il en manque pour de nombreuses applications :

Les fonctions utiles en mathématiques sont regroupées dans une bibliothèque spéciale, un module que l’on doit importer si nécessaire.

La fonction import est spéciale : on l’écrit généralement en début de programme pour importer tout ou une partie du module (l’importation est alors effectuées pour toute la durée d’utilisation du programme) mais on peut écrire l’importation n’importe où dans le programme (l’imortation ne concerne pas la partie du programme située en avant).

Si on a besoin d’une seule fonction du module, par exemple de la fonction sqrt (racine carrée) du module math, on peut écrire from math import sqrt et utiliser cette fonction telle quelle : par exemple a=sqrt(2).

Si on veut utiliser deux fonctions du même module, ou davantage, on peut importer tout le module en écrivant from math import \* ou import math mais cela peut être gênant car certaines fonctions du module peuvent avoir des noms qu’on souhaite redéfinir. Le risque de conflits entre les fonctions imortées et les fonctions que l’on va créer ne doit pas être négligé. Généralement, on n’importe que les fonctions nécessaires, en les séparant par une virgule. Supposons qu’on utilise dans un programme les fonctions sin et cos : il suffit d’écrire from math import sin,cos. Notez qu’il est possible d’importer le module en écrivant import math et d’utiliser les fonctions nécessaires en les préfixant, par exemple math.cos(0) retourne1.

Donnons un exemple, en important les fonctions du modules math au moyen de l’instruction from math import\* et en définissant dans le programme la fonction factorial. Cette fonction étant définie dans le module, selon la place de l’instruction d’importation il va y avoir un écrasement de notre fonction factorial (ce qui n’est pas souhaitable) par celle du module ou bien le contraire (notre fonction écrase celle du module, ce qui n’est pas souhaitable non plus).

0 !=1

2 !=2\*1

3 !=3\*2\*1

= 3\*2 !

4 !=4\*3\*2\*1

= 4\*3 !

5 !=5\*4\*3\*2\*1

= 5\*4

6 !=6\*5\*4\*3\*2\*1

= 6\*5 !

|  |  |
| --- | --- |
| Def factorial (n) : math.factorial l’emporte  If n%2==1 : return n  Else : return n//2  From math import \*  a=6  print(« factorial ({})={} ».fromat(a,factorial(a))) | >>> factorial (6)=720 |
| From math import \* factorial l’emporte  Def factorial (n) :  If n%2==1 : return n  Else : return n//2  a=6  print (« factorial ({})={} ».format(a, factorial(a))) | >>> factorial (6)=3 |
| Def factorial (n) :  If n%2==1 : return n  Else : return n//2  Import math  a=6  print (« factorial ({})={} ».format(a,math.factorial(a))) | >>> factorial (6)=720  factorial (6)=3 |

Remarque : on peut obtenir une aide pour un module importé en utilisant le built-in help.

2.1. Introduction La Programmation en Python

Si je tape dans la console import math, le module étant imorté je peux obtenir l’aide sur ce module en tapant help(math) : cela affiche un texte assez long qui décrit toutes les fonctions du module qui commence par :

Help on built-in module math :

NAME math

DESCRIPTION this module is always avaidable. It provides acces to the mathamtical functions defined by the C standard. FUNCTIONS acos(…)…

(tan(s)=y ⬄ x=arctan(y)

Voici la liste des fonctions du module math :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Acos | Acosh | Asin | Asinh | Atan | Atanh | Atan2 | Ceil | Copysign |
| Cos | Cosh | Degrees | Erf | erfc | Exp | Expm1 | Fabs | Factorial |
| Floor | Fmod | Frexp | Fsum | Gamma | Hypot | Isinfinite | Isinf | Isnan |
| Indexp | Igamma | Log | Log10 | Log1p | Log2 | Modf | Pow | Radians |
| Sin | Sinh | Sqrt | Tan | Tanh | Trunc | E | pi |  |

Les modules présents dans la distribution Python dès votre ordinateur sont certainement nombreux. Citons le module random qui contient, entre autres, des fonctions générant des nombres aléatoires (random, randint, randrange, …) ou permettant d’obtenir un élément tiré au hasard dans une liste (choice, …). Certains modules permettent de dessiner (modules turtle ou tkinter), d’autre donnent des informations sur des dates ou les durées (modules datetime ou calendar), etc. Dans 99,9% des cas, quelle que soit l’utilisation que l’on veut faire de Python, il y a un module spécialisé qui existe déjà.

Il est possible cependant, et même très facile de créer de nouveaux modules. Si vous avez de nombreuses fonctions qui sont utilisées par plusieurs programmes, il est intéressant d’enregistrer ces fonctions à par dans un programme utilitaire (c’est-à-dire un module) et de faire appel à lui dans d’autres programmes, grâce à la fonction import.

Si j’ai enregistré, entre autres, une fonction longueur() dans un fichier dimension.py, je peux faire appel à cette fonction dans un autre programme (situé dans le même dossier que dimensions.py), en l’important dans mon programme par l’instruction from dimension import longueur. J’accède alors cette fonction comme si était dans mon programme.

**Les entrées avec input**

L’interaction entre l’utilisateur et le programme ne se limite pas à l’affichage sur la console : il est possible d’entrer des données à l’intérieur d’un programme en cours d’exécution par l’intermédiaire de la fonction built-in input. Cette fonction a pour syntaxe input(<chaine>)où <chaine> est une suite de caractères entrée au clavier (l’entrée standard). Si un programme contient l’instruction t=input(<chaine>), lors de l’exécution il va y avoir **chaine** qui s’affiche à la console, le programme est en attente d’une entrée au clavier. La saisie est terminée lorsque l’utilisateur appuie sur la touche Enter, le programme continue alors son exécution et affecte ici la chaîne de caractères, saisis la variable t. **input**(<chaine>) retourne toujours une chaîne de caractères, même si les caractères sont des chiffres : c’est à l’utilisateur de convertir la saisie en entier ou en flottant, selon les attentes du programme. On peut tester si la saisie est un nombre entier avec la méthode de la classe string **isnumeric**() : par exemple **« 123 ».isnumeric()** retourne **True** alors que **« 1.3 ».isnumeric()** retourne **False.** Pour convertir la saisie en entier ou en flottant, utiliser l’opérateur de cast adapté **int()** ou **float()**.

**Les commentaires**

Des commentaires peuvent être ajoutés en fin de ligne, après le caractère #. Si on veut écrire un commentaire sur plusieurs lignes, il vaut mieux utiliser la syntaxe « « «  <commentaire sur plusieurs lignes> » » ».

Les commentaires font partie des bonnes pratiques de programmation, au même titre que des choix explicites pour les noms de variables : si le programmeur veut espérer se comprendre lui-même et être compris par les autres, il doit absolument respecter ces recommandations. Dans l’exemple ci-dessous j’utilise un long commentaire pour dire comment s’appellle le programme, ce qu’il fait, ce qu’il prend en entrée, etc. On peut ajouter d’autres informations comme le nom de l’auteur, la date de création, un historique des différentes versions… Cela est surtout important pour les grands projet impliquant plusieurs personnes et de nombreux programmes dont le développement et la maintenance s’étale sur plusieurs années.

Pendant la phase d’écriture d’un programme, on peut aussi mettre en commentaire une séquence d’instruction afin d’en isoler une partie à tester. On peut commenter aussi les parties obsolètes ou utiles seulement dans certains cas. Par exemple, on peut être amené à afficher le contenu de certaines variables pour tester le bon fonctionnement d’un programme. Quand ce besoin ne se fiat plus sentir, on peut mettre l’instruction d’affichage en commentaires. Attention cependant, dans la phase finale, supprimer tous les commentaires devenus superflus.

Exemple 1 :

Ecrivons un programme qui demande une vitesse donnée en km/h et qui l’affiche en m/s après l’avoir arrondi au centimètre près.

Il y a un petit traitement mathématiques pour connaître la réponse : on doit multiplier par 1000 et diviser par 3600. Le résultat étant en m, il suffit de l’arrondi à 2 chiffres après la virgule pour obtenir l’arrondi au centimètre.

La fonction d’arrondi est une des fonctions built-in, round(), qui prend deux valeurs en argument : le nombre à arrondir et le nombre de chiffres après la virgule, séparés par une virgule.

Comme un peu plus haut, ce programme utilise un test pour afficher la bonne réponse si un entier est entré et un message indiquant l’erreur si un autre caractère qu’un chiffre est entré. L’étude de la syntaxe

|  |  |
| --- | --- |
| From random import random  P=0.8  If random()<P :  Print (1)  Else :  Print (0) | def balayage(f,k,a,b,n) :  fa,fb=f(a),f(b)  h=10\*\*(-n)  x=a  if fa<fb:  while f(x)<k :  x=x+h  else :  while f(x)>k :  x=x+h  return (round(x-h,n)round(x,m))  Tester pour balayage(f, 20, -3, -1, 2) |

(r)Représentation des entiers relatifs (/r)

Il existe plusieurs façons de prendre en compte le signe d’un entier relatif. Historiquement, on a utilisé la méthode (r)du complément à 1, dite aussi du complément logique.(/r) avec ce codage, pour changer de signe, Il faut inverser la valeur de chaque bit. Par exemple : 11 sur un octet binaire : 1011, 11 s’écrivant 00001011, comment code-t-on -11 ? 11110100. 11 110 100 : 27+26+25+24+22=128+64+32+16+4=244

La représentation actuelle, utilisée dans tous les processeur est la méthode des complément à deux dite aussi du complément arithmétique.

(r) Définition 1.2 : Les entiers signés en compléments à 2. (/r)

Le codage des entiers relatifs se décompose en deux parties :

Un terme négatif sur un seul bit (le bit le signes) et un terme correctif, positif (écrit sur tout les bit de poids faible « Le premier bit (r)(celui de gauche, il est dit « bit de poids fort ») Indique le signe, (/r) si c’est 1, le nombre sera négatif sinon, il sera positif.

(r)Remarques : (/r) pour un nombre binaire s’écrivant sur un octet n=a7a6a4a3a2a1a0, la décomposition donnant la valeur de n s’écrit : (r)[a0+a1\*2+a2\*2²+a3\*23+a4\*24+a5\*25+a6\*26]-a7\*27. (/r)

Par exemple, le nombre codé 10001100 en compléments à 2 sur 1 octet vont donner : [22+23]-27=12-128=-116. 1001 en complément 2 sur 4 bits : [20]-23=1-8=-7.

Pour déterminer une représentation en complément à 2 d’un nombre relatif, il suffit d’ajouter (r) 1 au complément logique (le complément à 1) (/r) de la valeur absolue de ce nombre. Par exemple, pour coder -7 sur 1 octet, on ajoute 1 au complément à 1 de 7(10)=00000111(2). C’est-à-dire : (r) 11111000+1=11111001. Vérification (/r) : La valeur de 11111001 se calcule [20+23+24+25+26]=-27=[1+8+16+32+64]-128=-7.

L’opposé d’un nombre se trouve en ajoutant 1 au complément à 1 du nombre. En effet, additionner un nombre et son complément donnera toujours -1 (codé rien qu’avec des 1, (r) quelque soit la taille de la représentation) ; en ajoutant 1 à cette somme, on obtient bien 0. (/r)

Cette méthode fonctionne dans les deux sens : pour savoir le nombre représenté en complément a2 par le mot binaire : 10111001, on ajoute 1 au complément à 1, soit 01000110+1=01000111. Comme 01000111(2)=71(10) on en déduit que le mot dbianire 10111001 code le nombre -111

Capacité de l’encodage en complément à deux pbits :

Du fait de l’utilisation d’un bit de signe, il ne reste que p-1 bits pour coder les valeurs.

* Les entiers codés sur 1 octet (8 bits) sont compris entre -27=-128 et 27-1=127
* Les entiers codés sur 2 octets (16 bits) sont compris entre -215=-32768 et 215-1=32767
* Les entiers codés sur 4 octets (32 bits) sont compris entre -231≈ 2,1\*109 et 231-1≈un peu plus de 2 milliards.
* Les entiers codés sur 8 octets (64 bits) sont compris entre -263≈-9,2\*1018 et 263-1≈un peu plus de 9 trillions.

Extensions de la capacité d’encodage :

Il est possible d’étendre la capacité de représentation d’un codage en ajoutant ver la gauche autant de fois que l’on souhaite le bit de poids fort (le bit de signe) par exemple, les entiers codés sur 1 octet sont étendus sur 2 octets en écrivant 8 fois le bit de signe à gauche de la représentation sur 1 octet :

* L’entier positif 01100111 est naturellement étendu sur 2 octets en écrivant : 0000000001100111.
* L’entier négatif 11100111 est étendu sur 2 octets en écrivant : 1111111111100111.

Comment se convaincre qu’il s’agit du même nombre ?

* L’opposé de 11100111 est 00011000+1=00011001 (on ajoute 1 au complément à 1), Celui de 1111111111100111est 0000000000011000+1=00011001, ces nombres ont les mêmes opposés donc ils sont égaux.
* En ajoutant rien que des 1 à gauche d’un nombre négatif (son bit est 1), l’opposé n’esyt pas modifié donc le nombre ne l’est pas non plus.

Addition et Soustraction d’entiers relatifs :

L’addition sur des entiers relatifs binaires en complément A2 suis les mêmes algorithmes que ceux définis pour les entiers naturels :

* On additionne les bit entre eux en propageant la retenue éventuelle La somme 1+1=10 par exemple, conduit à une retenue de 1. Pour effectuer une soustraction, on la transforme en une addition de l’opposé (on ajoute alors le complément à 2). Par exemple, la soustraction de 12-15, on effectue l’addition de l’opposé 12+(-15) . Le complément à 2 sur 1 octet de 15(10)=1111(2) étant 11110001 comme 12(10)=1100(2) il faut effectuer l’addition :11110001+00001100. Il n’y a aucune retenue dans cette somme, effectuons-la en ligne : On trouve 11111101, soit l’opposé de 00000011(2)=3(10) c’est-à-dire -3.

Bien sûr, en procédant sur des mots binaires de tailles fixées, ces méthodes peuvent entraîner des dépassements de capacité que l'on peut détecter en examinant le bit de signe: en ajoutant deux nombres positifs, si le bit de signe est 1, cela indiquera une retenue sortant du dernier rang non nulle et donc un dépassement de capacité. De même, si on ajoute deux nombres négatifs et qu’on obtient un bit de signe nul.

Exercices :

Addition :

Significations :

45+27=

Retenues :

-83+(-37)=

Retenues :

45+91=

Retenues :

Signification : 45+27=72

Retenues

01111110

0100100

00101101

0100100

NB : 45-27

Pour trouver un résultat positif, le bit de signe du nombre négatif (-27) doit bien être neutralisé par une retenue entrante.

Cela provoque de ce fait, une retenue sortante qui n’a aucune signification.

Les retenues entrantes sont représentées en rouge.

Signification : 45+91**=**

**De dépassement**

Retenues

Soustractions :

Signification 45-27=18

Transformation 45+(-27)=18

Retenues :

-83-(-37)=-46

-83+37=-46

Retenues :

Significations : -83+(-37)

(83)10=(01010011)2 ~~test~~ = 10101100=-83

(37)10=(0100101)2 ~~test~~ = 11011010=-37

01010011

(01111000)2=(+120)10

(32+16+8)-64

56-64=-8

Autre méthode :

-83+(-37)=-120

Retenues :

0000001

10000111

10001000

45+91

45=10111

91=1011011

10001000

Autre technique :

Retenues 1010011

37|- = 1

|-=0

|-=0

|-=1

|-=0

|-=1

83|-=1

41|-=1

20|-=0

10|-=0

5|-=1

2|-=0

1|-=1

1-(101001)

⬄(010110)

⬄01011

-(1010011)

⬄-(0101100)

⬄0101101



Fraction dyadique

Une fraction dyadique est un nombre rationnel qui peut s’écrire sous forme de fraction (r)avec pour dénominateur une puissance de 2. D=

+=

-=

\*=

[www.tropheestangente.com](http://www.tropheestangente.com)

[www.tropheetangente.com/PTL2019.php](http://www.tropheetangente.com/PTL2019.php)

[www.tropheetangente.com/photos\_dossier\_de\_presse\_2017.php](http://www.tropheetangente.com/photos_dossier_de_presse_2017.php)

[www.tropheetangente.com/index.php](http://www.tropheetangente.com/index.php)

Rappels : Codage binaire

Des bits : d’un point de vue matériel, un ordinateur est un ensemble de composants électroniques, parcouru par des courants électriques. Par convention, Le passage d’un courant dans un composant, est codé par le chiffres 1, L’absence de courant est codée par le chiffres 0. Ainsi, toute information, stockée dans un ordinateur, peut être codé par une suite fini de 0 et de 1 appelé suite de bits (binary digits). Une convention fixe la taille de ces suites fini de bits, un codage sur 4 bits, signifie qu’une information est représenté en machine à l’aide de 4 0 et/ou 1. L’ordre de ces bits importe de sorte que 011 ne code pas la même information que 101. En outre, les 0 présents en début de codage sont indispensables. Ainsi, sur 4 bits, 24=16 entiers naturels peuvent être codé. Par exemple les entiers de 0 à 15 :

1. ⬄0000
2. ⬄0001
3. ⬄0010
4. ⬄0011
5. ⬄0100
6. ⬄0101
7. ⬄0110
8. ⬄0111
9. ⬄1000
10. ⬄1001
11. ⬄1010
12. ⬄1011
13. ⬄1100
14. ⬄1101
15. ⬄1110
16. ⬄1111

Nombres entiers codés sur n bits :

Si n appartient à ℕ\*, 2n entiers peuvent être codés en binaire sur (r)n bits(/r).

1. 🡪…🡪(2n-1)

* Pour n=4 bits : on a 24-1=16 entiers possibles de 0 à (r)24-1=15(/r)

Pour n=8 bits : On a 28= (24)² = 16²=256 entiers possibles. De 0 à (r)28-1=255(/r)

Pour n=16 bits : on a :

216=(r)65536 entiers possibles(/r)

De 0 à 216-1=65 535.

(r)Arithmétique élémentaire.(/r)

Cela signifie que les entiers plus grand que 2n ne peuvent pas être codé sur n bits. La capacité de codage sur nbits est dépassée. Soit on ajoute des bits pour coder des plus grand entiers, soit-on se content de coder les entiers sur nbits.

(r)Illustration :(/r)

Pour n=4bits et deux entiers naturels :

* n1=3
* n2=4

L’addition de ces entiers est l’entier n3=7

En binaire, les additions sont effectuées suivant les règles suivantes. A savoir : (r)\*0+0 donne 0 avec une retenue égale à 0.

\*0+1 donne 1 avec une retenue égale à 0.

\*1+0 donne 1 avec une retenue égale à 0.

\*1+1 donne 0 avec une retenue égale à 1. (/r)

Le codage de n1 et n2 sur 4 bits donne respectivement 0011 et 0100. L’addition de ces codages peut être posée et s’écrit :

(r)Retenue 0000 (/r)

0011

+0100

\_\_\_\_\_

00111

(r)Le résultat obtenu est 0111 sur 4 bits correspond au codage binaire de 7 (/r)

n1=11 :

11|-2=5|-2=2|-2=1

1 1 0

Résultat 1101

n2=13 :

13|-2=6|-2=3|-2=1|-2=0

1 0 1 1

1011

1101

1101+1011=0

(r)Ecriture d’un entier relatif en binaire sur n bits :

Méthode :(/r) On considère un nombre relatif n en base 10.

1. Si n≥0, On utilise la méthode de conversion en base 2.
2. Si n est négatif, on code la valeur absolue de |n| en binaire puis :
   1. Etape 1 : on prend le complément à 1 (non logique) de l’écriture binaire, C’est-à-dire que l’on remplace les 1 par des 0 et les 0 par des 1.
   2. Etape 2 : On ajoute 1 au résultat, plus exactement (0000 0001)2 si on est sur 8 bits. On remarque que l’écriture binaire de l’entier négatif n correspond à celle de (r)n+2N.

(r)exercice : représentation binaire de (-8)10 sur 8 bits. (/r)

Etape 1 : On prend le non logique encore nommé complément à 1 Sur 8 bits.

|-8|=8 :(8)10=(0000 1000)2 🡺 (1111 0111)2

🡺 = par complément à 1.

Etape 2 : On ajoute 1 au résultat (d’où le complément à 2).

(1111 0111)2+(0000 0001)2=((r)1111 1000)2(/r)

Donc (8)10=(0000 1000)2 🡺 (-8)10=(r)(1111 1000)2(/r)

🡺 = par complément à 2

L’écriture binaire de l’entier négatif n correspond à celle de (r)n+28

(1111 1000)1=1\*23+1\*24+1\*25+1\*26+1\*27=8+16+32+64+128=(248)10 🡺 248-28=248-256=-8.(/r)

(r)Définition :(/r) Soit un entier naturel sur n bits :

((r)xn(/r)xn-1…x1)2

* xn est le bit de poids fort (indique le signe)
* x0 est le bit de poids faible.

(r)4) Représentation booléenne(/r)

Le type de donnée envisagée ici est le type booléen nommé d’après l’inventeur de l’algèbre qui porte son nom, Boole, ce type de donné ne prend que deux valeurs : soit oui/non ou true/false, 1 ou 0 pour les ordinateurs. Une donnée booléenne ne tient donc que sur 1 seul bit. Ce qui nous intéressera tout au long de l’année, c’est de comprendre les fonctions booléennes. Les fonctions booléennes : qui associes un booléen à (r) un ou plusieurs booléens.(/r) Une fonction booléenne se définie d’une façon symbolique comme (r) les fonctions numériques qui utilise des symboles opératoires.(/r) Mais aussi, à l’aide d’un table, dite table de vérité. Nous allons travailler sur quelques exemples fondamentaux de cette table de vérité.

(r)a. Fonctions booléennes de base (/r)

La plus simple des fonctions booléennes est la fonction non (false), elle associe vrai à faux et faux à vrai. On pourrait l’appelée la fonction contraire ou complement ou encore not (c’est d’ailleurs celle utilisée en python). Ainsi, non (0)=1 et non(1)=0. En python, not(True) donne False et note (False) donne True. Comme le test de l’égalité est noté ‘==’(2 symboles d’égalité), l’instruction a%2==0 donnera True si et seulement si a est pair. (a%2 est le reste de la division euclidienne par 2).

(r)Table de vérité de cette fonction (/r)

|  |  |
| --- | --- |
| Non | |
| x | non(x) |
| 0 | 1 |
| 1 | 0 |

|  |  |  |
| --- | --- | --- |
| ET | | |
| x | y | x et y |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |
| 0 | 0 | 0 |

|  |  |  |
| --- | --- | --- |
| Ou | | |
| x | y | x ou y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Le fonction booléenne ‘et’ agit sur agit sur deux booléens, noté x et y. Pour que ‘et(x,y)=1’, il faut que x=1 et y=1. Dans tous les autres cas x et y = 0.

En python, cette fonction est notée (r)and(/r) et on a, par exemple, (r)True and True qui donne True.(/r) Pour être plus concret, a<0 and b==1, donnera (r)True si et seulement si le contenu de A est négatif et le contenu de b est égal à 1.(/r)

(r)Propriété 1 : (Les Lois de Morgan) :(/r)

Pour tout couple (x,y) de bits, on a : non(x et y)=non(x) où non(y), et non(x ou y)=non(x) et non(y). Autrement dit :

1. (r) L’expression x et y peut être remplacée par l’expression non(non(x) ou non(y))(/r).
2. L’expression x ou y peut être remplacée par l’expression (r)non(non(x) et non(y))(/r)

Démonstration :

Dressons les tables de vérité de n(r)on (non(x) ou non(y)) et non(non(x)etnon(y))(/r), Puis les comparer avec celles de x et de y et x ou y.

Voici la table de non(non(x) ou non(y)) que nous allons dresser progressivement en utilisant la fonction non, et ensuite la fonction ou (r)et enfin la fonction non à nouveau.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Non(non(x) ou non(y)) | | | | | |
| x | y | Non(x) | Non(y) | Non(x) ou non(y) | Non(non(x) ou non(y)) |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 |

On constate que pour chaque couple (x, y) possible, les valeurs de vérité coïncident dans cette table et dans (r)celle de x (/r)ou(r) y.(/r)

La fonction ou est un ou inclusif qui apparait dans l’expression : « je serai là, qu’il pleuve ou qu’il vente » (r)(je serai là aussi s’il pleut et vente, simultanément)(/r)

La fonction ou exclusif de la langue, par exemple dans un menu de restaurant : « dessert ou café » **(sous-entendu, ce n’est pas prévu de commander les deux)**

**Cette fonction peut se noter oux ou xor en anglais : informatique.**

|  |  |  |
| --- | --- | --- |
| Oux(xor) | | |
| x | y | x et y |
| 0 | 0 | **0** |
| 0 | 1 | **1** |
| 1 | 0 | **1** |
| 1 | 1 | **0** |

**Quelle est l’expression du oux à l’aide des fonctions de base ?**

**D’après la table, pour avoir x ou y=1, il faut et il suffit d’avoir non(x) et y ou x et non(y), d’où l’équivalence x oux y=(non(x) et y) ou (x et non(y)).**

On peut traduire cette expression sans le « et » :

(r)x oux y=[non(non(non(x)) ou non(y))] ou [non(non(x) ou non(non(y))]

« Pour un humain normal, cette écriture peut s’avérer incompréhensible »(/r)

(r)b. Algèbre de Boule :(/r)

On Appelle (r)addition(/r) la fonction « ou » et (r)multiplication (/r) la fonction « et ».

On définit l’ordre 0<1 sur les éléments booléens.

Cet ordre permet de redéfinir :

(r)\*le ou (+) : x ou y=x+y=max(x,y)

\*le et(x) : x et y = x\*y = min(x,y).(/r)

Les propriétés, connues pour la plupart dans les ensembles de nombres, de l’addition et de la multiplication se retrouve ici, ce qui justifie l’emploi (r)de ces symboles :(/r)

(r)1) Commutativité (/r)

x+y=y+x, le ou(+) est commutatif ; x\*y=y\*x, le et (\*) est commutatif.

(r)2) Associativité(/r)

(x+y)+z=x+(y+z), le ou(+) est associatif.

(x\*y)\*z=x\*(y\*z), le et(\*) est associatif.

(r)3) élément neutre (/r)

On appelle « élément neutre » d’une opération un élément particulier de l’ensemble sur lequel porte l’opération qui ne modifie (r)pas l’autre élément.(/r)

Comme x+0=0+x=x, 0 est l’élément neutre de ou(+).

Comme x\*1=1\*x=x, 1 est l’élément neutre de et(\*).

(r)4) élément absorbant(/r)

On appelle « élément absorbant » d’une opération, un élément particulier de l’ensemble sur lequel porte (r)l’opération qui ne modifie pas l’autre élément. (/r) Comme x\*0+0\*x=0, 0 est l’élément absorbant de et(\*).

**5) Distributivité**

Il y a 2 sorte de distributivité (pour les nombres, il n’y en a qu’une, c’est x qui est distributive par rapport à « + » comme dans l’exemple : 3m(5+6)=3\*5+3\*6 ;

**-** x(y+z)=**(x\*y)+(x\*3)** est distributive par rapport à + ;

- x+(y\*z)= **(x+y)\*(x+z)** + est distributive par rapport à \*.

**6) Complément**

En notant (barre au-dessus (bad))x le contraire (complément de x, on constate que : (r) x+(bad)x=1 et x\*(bad)x=0. (/r)

Le complément est dit (r)involutif car ((bad)\*2)=x.

(r)7) Idempotence(/r)

L’addition et la multiplication sont (r)idempotentes(/r) car x+x=x et x\*x=x. Cette propriété permet les simplification comme par exemple (r) : x\*(x+y)=x+x\*y.(/r)

(r)Priorité : (/r)

Pour faciliter leur compréhension, on a décidé que ces opérations, seraient soumises aux mêmes règles que les opérations sur les nombres, la fonction « et », la multiplication logique, est (r)ainsi prioritaire par rapport à la fonction ou, la somme logique. (/r)

Par exemple (exercice) :

Ecrire les tables de vérité des trois expressions suivantes sous python :

x+y\*z ; x+(y\*z) et (x+y)\*z.

Nous constatons que : (r)x+y\*z=x+(y\*z)

Alors que x+y\*z≠(x+y)\*z (/r)

Ce qui confirme que les opérateurs (r)and(/r) et (r)or(/r) de python suivent bien la règle de priorité donnée précédemment.

**C. Multiplexeur**

La fonction de multiplexage **mux** agit sur trois booléens, notés **x, y et z.** Cette fonction est définie par les deux conditions : **mux (o,y,z)=y et mux(1,y,z)=z.** Cette définition conduit à dresser la table de cette fonction. On pourra alors dresser une deuxième table, celle de l’expression **(non(x) et y) ou (x et z) qui peuvent s’écrire \*y+x\*z**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| MUX | | | | | | | |
| x | y | z | mux(x,y,z) | non(x) | non(x) et y | x et z | (non(x) et y) ou (x et y) |
| 0 | 0 | 0 | 0 | **1** | **0** | **0** | **0** |
| 0 | 0 | 1 | 0 | **1** | **0** | **0** | **0** |
| 0 | 1 | 0 | 1 | **1** | **1** | **0** | **1** |
| 0 | 1 | 1 | 1 | **1** | **1** | **0** | **1** |
| 1 | 0 | 0 | 0 | **0** | **0** | **0** | **0** |
| 1 | 0 | 1 | 1 | **0** | **0** | **1** | **1** |
| 1 | 1 | 0 | 0 | **0** | **0** | **0** | **0** |
| 1 | 1 | 1 | 1 | **0** | **0** | **1** | **1** |

**NB : nous constatons que ces deux tables sont égales, ce qui signifie que :   
mux(x,y,z)=\*y+x\*z**

Cette fonction permet au niveau du processeur, de définir une (r)commande qui produit, (/r) en sortie, la valeur de l’entrée y(si la commande x=0) ou la valeur de l’entrée z (si la commande x=1).

Une autre application théorique cette fois de cette fonction : elle permet de montrer que toutes les fonctions booléennes peuvent s’exprimer avec seulement les fonctions (r) »non », « et » et « ou » (/r) par récurrence sur le nombre de booléens en entrée :

* S’il n’y a qu’un seul booléen, il n’y a que 4 fonctions possibles : la fonction (r)« non » ;(/r) la fonction constante (r) »n »(n(0)=0 et n(1)=0) ;(/r) la fonction (r) « identité » (id(0)=0 et id(1)=1) ;(/r) la fonction (r) « constante y » (y(0)=1 et y(1)=1). (/r) Id(x)=x pour n, on a n(x)=(r)x et non (x) ;(/r) pour y on a (r)y(x)=x ou non(x).

1) Quelle est l'autre nom de la norme ISO-8859-1?

l'ANSI, latin-1,

Décoder le texte suivant écrit en ISO-8859-1 avec des caractères convertis en entiers décimaux:

82 101 110 100 122 45 118 117 115 32 58 10 97 110 110 117 108 233

R e n d z - v u s " " : " " a n n u l é

R e n d z \_ v u s " " : " " a n n u l é

Technique pour décrypter l’iso-8859-1 :

82=16\*5+2 🡪 C5 L2 = R

101=16\*96+5 🡪 C6 L5 = e

110=16\*6+14 🡪 C6 L14 = n

100=16\*6+4 🡪 C6 L4 = d

122=16\*7+10 🡪 C7 L10 = z

45=16\*2+13 🡪 C2 L13 = \_

118=16\*7+6 🡪 C7 L6 = v

117=16\*7+5 🡪 C7 L5 = u

115=16\*7+3 🡪 C7 L3 = s

32=16\*2+0 🡪 C2 L0 = " "

58=16\*3+10 🡪 C3 L10 = :

10=16\*0+10 🡪 C0 L10 = " "

97=16\*6+1 🡪 C6 L1 = a

110=16\*6+14 🡪 C6 L14 = n

110=16\*6+14 🡪 C6 L14 = n

117=16\*7+5 🡪 C7 L5 = u

108=16\*6+12 🡪 C6 L12 = l

233= = é

2) D'après ce qui précède, quelle est la réponse de l'interpréteur python aux instructions :

- chr(233)

- ord ('é')

- list('é'.encode('latin1'))

>>> chr(233)

'é'

>>> ord ('é')

233

>>> list('é'.encode('latin1'))

[233]

>>> list('é'.encode('utf8'))

[195, 169]

3) Décoder le texte suivant encodé en ASCII complété à 1 octet:

01001000 01100101 01101110 01110010

H e n r

01101001 00100000 01001001 01010110

i " " I V

exemple pour décoder de l’ASCII :

01001000 🡪 0100/1000 🡪 100+1000 🡪 C4 L8 = H

01100101 🡪 0110/0101 🡪 110+0101 🡪 C6 L5 = e

01101110 🡪 0110/1110 🡪 110+1110 🡪 C6 L14 = n

01110010 🡪 0111/0010 🡪 111+0010 🡪 C7 L2 = r

01101001 🡪 0110/1001 🡪 110+1001 🡪 C6 L9 = i

00100000 🡪 0010/0000 🡪 010+0000 🡪 C2 L0 = " "

01001001 🡪 0100/1001 🡪 100+1001 🡪

01010110 🡪 0101/0110 🡪 101+0110 🡪

Lire le livre AL-Jabr-Muqabala d’Alkwarizmi.

Lire le livre D’Alkhwarizmi à Descartes.

Lire Gorgias de Platon

liste = (82 ,101 ,110, 100, 122 ,45 ,118 ,117 ,115 ,32 ,58, 10 ,97 ,110 ,110, 117 ,108, 233)

for z in liste:

print(chr(z)) #Quamé

<https://fr.wikipedia.org/wiki/UTF-8>

dresser les tables de vérité des expressions :

* nand(a,b)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | b | a et b | nand(a,b) | nand(a,a) |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 |

* nand(a,a)
* nand(nand(a,b),nand(a,b))

|  |  |  |  |
| --- | --- | --- | --- |
| a | b | nand(a,b) | nand[nand(a,b),nand(a,b)] |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 |

Compo dure 2 heures, coix entre 2 sujets comme dans un DST,

combo conseillé pour conserver NSI c’est NSI-MATHS

|  |  |
| --- | --- |
| Enseignement commun | coef |
| Franc écr+orl | 5+5 |
| Philo | 8 |
| HG/LV1/LV2/EPS/ENS | 40 avec spé 1ère |
| Épreuve orale terminale | 10 |

|  |  |
| --- | --- |
| Enseignement de spécialité | |
| HGGSP | coef 16 pour chacune des 2 spé term |
| Maths |
| Physique-chimie |
| SES |
| LLCE |
| NSI |
| HLP |
| SVT |

Correction de al composition :

Ex 1 :

a)

= += 3+1\*=(10,1)3

b)

(12)3 = 1\*3+2\*30=5

(101)3=1\*32+1\*30=10

(20,2)3=2\*31+2\*3-1 = 6+ =

c)

2121

+1212

11110 en base 3

n=a0+a0+a1\*a1+…

ex 2 :

a)

72=1\*22+1\*21+1\*20 = 1112

25=

(1010)(2)=1\*23+1\*21+0\*20 = 8+2 = 10(10)=a(16)

-27=-128à27-1=128-1=127

-1|-1|=1 = 0000 0001+1111 1110 + 0000 0001 / 1111 1111.

-64

|-64|=64=1\*26=0100 0000 = 1011 111+0000 0001

111 111

1011 1111

+0000 0001

\_\_\_\_\_\_\_\_\_\_

1100 0000

-100 = |-100|=100= 64+32+4 = 1\*26+1\*25+1\*22 = 0110 0100

Complément à 1

1001 1011

+0000 0001

\_\_\_\_\_\_\_\_\_\_

1001 1100

1000 0001 = 1\*27+1\*20 = -128+1 = -127

-127 on ajoute 8 bits égaux de poids fort.

1111 1111 1000 0001.

1110 0101

+1100 0011

\_\_\_\_\_\_\_\_\_\_

1 0110 1111

(r)II Séquences conditionnelles et boucles.(/r)

Les outils de programmation sont indispensables et existent dans tous les langages de programmation. Il y a des différences de syntaxe entre les langages. Ici comme dans tout les languages. Nous nous limiterons au language python.

a) Séquences conditionnelles.

Les instructions conditionnelles utilisent des test(r), donc des booléens.(/r)

Une instruction conditionnelle permet de programmer ce type de situation :(r)

* si un test est vrai, (le booléen vaut « True ») (/r) alors on effectuera un bloc d’instructions. (r)
* Sinon (le booléen vaut « False »)(/r) alors on effectuera un autre bloc d’instructions.

Dans tous les cas, le programme continue avec ce qui suit en bloc d’instruction.

Une instruction conditionnelle s’écrit dans le langage python

(r)if <test> :

<instructions 1>

else :

<instruction 2>

La partie (r)else :…(/r) est une option

Elle peut ne pas être présente. Il est possible d’imbiquer une instruction conditionnelle dans une autre. S’il n’y a qu’une seule instruction dans un bloc, on peut la mettre après les deux points, sinon il faut mettre un bloc d’instruction sur d’autres lignes, en respectant (r)l’indentation (décalage régulier d’une tabulation vers la droite) (/r) car c’est l’indentation d’un bloc qui permet au programme de savoir où se situe (r)la fin du bloc (/r) Si on est amené à écrire un nouveau test, juste après un (r)else(else : if <test>…),(/r) python dispose de l’instruction (r)elif<test> :…(/r) qui permettra de simplifier l’écriture car une nouvele indentation n’est pas requise. On peut enchainer plusieurs instruction (r)elif <test> :…(/r) Nous verrons dans quel but plus loin. Les instructions (r)elif <test> :..(/r) peuvent toujours être suivies (ou pas ) d’une instruction else :… . Dans certains langages (C, C++, java, javascript) il existe une forme d’instruction conditionnelle qui examine l valeur d’une variable et qui exécute le bloc d’instructions correspondant à la valeur. Généralement, le mot clé utilisé alors est (r)switch et s’utilise ainsi :(/r)

Switch <variable> :

Case (Valeur 1) :…break

Case (Valeur 2) :…break

Default :…

(r)NB : python n’a pas implémenté cette fonctionnalité.(/r)

Un test ne se limite pas au seul comparaisons <, <=, >, >=, == et !=. On peut utiliser l’opérateur (r)disjonctif or (v)et l’opérateur conjonctif(/v) and, (/r) ainsi que la négation (r) not,(/r) pour réaliser des (r)tests complexes(/r).

Pour clarifier l’écriture de ce type de test, les parenthèses sont autorisées mais il faut savoir l’ordre des (r)priorités décroissante : la comparaison, not, and, or.(/r)(v)on pourra ainsi éviter les pertes de parenthèses qui alourdissent l’écriture. (/v)

(r) Types de données construits (/r)

(r) Définition : (/r) P-uplet

Un P-uplet-« tuple » en python est une suite de valeurs séparées par des virgules et délimitées par des parenthèses (ce délimiteur est facultatif). La déclaration du tuple vide (r)t1(/r) s’effectue en écrivant (r)t1=()(/r) (ce genre de déclaration est utile quand on doit déclarer le tuple alors qu’on ne sais pas les valeurs a y mettre, par exemple avant une boucle ou elles sont définies). Le tuple (r)t2(/r) déclaré par l’instruction (r)t2=(-1,1)(/r) contient deux valeurs. On accède en lecture à la première valeur du tuple (r)t2(/r) en écrivant (r)t2[0](/r). L’instruction (r)print(t2[0]) (/r) affiche : -1. La deuxième valeur de t2 étant t2[1] on peut affecter à une variable (r)b=t2[1](/r) On peut parcourir la liste des indices à l’envers, t[-1] retourne le dernier élément du tuple (r)t.(/r) On ne peut pas modifier les valeurs d’un tuple, (r)c’est une liste non-muable.(/r). Par contre on peut réaffecter un tuple. Par exemple t1=(0,1) alors que t1 était le tuple vide. Mais l’instruction t2[0]=0 conduirais à l’erreur :

l (r)’tuple’ object does not support item assignment. (/r)

On peut omettre les parenthèses qui sont facultatives.

Le tuple (r)t3(/r) déclaré par l’instruction (r)t3=-1,0,1 est tout à fait valide(/r).

Il contient trois valeurs qui peuvent être identifiées par les notations (r)t3[0],t3[1] et t3[2](/r)

Si un tuple contient n valeurs, on peut affecter n variables en les disposants comme dans un tuple.

On peut par exemple affecter trois variables avec le tuple t3 précédent, en écrivant a,b,c=t3. Dans ce cas, (r) a contient t3[0], b contient t3[1] et c contient t3[2]. (/r)

Cette particularité a déjà été utilisée dans le chapitre précédent, dans le chapitre 1.

Quand on écrit : x1, x2, y1, y2=20,300,100,500.

Pour initialiser quatre variables sur une seule ligne. C’est un tuple de variable qui est initialisé aec un tuple de valeurs. En procédant sans tuple, il faut utiliser une variable (r) tampon c=a, puis a=b et enfin b=c. (/r)

Avec un tuple il suffit d’écrire a,b=b,a.

La structure de tuple quand on veut enregistrer les données variées concernant un objet, constitué d’une sorte de base de données. Supposons que l’on créé une sorte de carnet d’adresse sous la forme d’une liste d’enregistrement où l’enregistrement correspondant à une personne contient, son nom, son prénom, son numéro de téléphone, et sont adresse e-mail. L’utilisation d’un tuple (r)registre(/r) pour les enregistrements permet de récupérer chacune des informations avec (r)l’instruction nom, prénom, tel, e-mail = registre.(/r)

**CHAPITRE III: ARCHITECTURE et OS (operating system)**

**objectif:** L'architecutre des machines et de leurs systèmes d'exploitation constitue un ensembmle de connaissances dont on peut se passer au niveau supeficiel (du point de vue d'un utilisateur ou d'un programmeur occasionnel) mais qui devient rapudement indispensable quand on approfondit le sujet. Ce cour ne fera qu'introduire les différentes notion qui sont, en alland du centre vers la périphérie:

**1) Le processeur et le langage machine**

Au cœur de tout machine informatique il y a des circuits électroniques (processeur, unité de stockage) combiné dans des circuits logiques dont on peut appréhender le fonctionnement sur des modèles simplifiés.

**2) Système d’exploitation et la ligne de commande**

Le système d’exploitation d’un ordinateur gère ces différentes fonctions ; elles s’appréhende en mot de terminal par la ligne de commande.

**3) Réseaux et protocoles :**

L’interconnexion des ordinateurs entre eu et de nos jours omniprésente. L’architecture des réseaux et des protocoles de communication sont des éléments de compréhension indispensables.

**4) La robotique :**

Les systèmes embarqués, les objets connectés et les robots sont en interaction avec les ordinateurs avec lesquels ils ont un lien de parenté.

=========================== fin objectifs ==============================

**1) architecture :**

**a) Le modèle de Von Neumann.**

Un ordinateur est principalement composé de deux circuits, le processeur et la mémoire, connectés entre eu par des « bus » de communications. Le processeur (CPU ()) est l’unité de traitement de l’information (l’instruction et les données).

Il exécute des programmes ((r)suite d’instructions qui définissent un traitement à appliquer à des données).(/r)

La mémoire centrale ((r)RAM (Random Access Memory) dite aussi mémoire « vive »)(/r) est une unité de stockage temporaire des informations nécessaires à l’exécution d’un programme externe au processeur, elle stocke en particulier les instruction du programme (r)en cour d’exécution et ses données (nombres, caractères (alphanumériques), des adresses mémoires, etc…).

Les « bus »(/r) de communication sont les supports physiques des transferts d’information entre les différents unités((r)bus de données, des bus d’adresse, des bus de commande).

Les périphériques(/r) sont des unités connexes permettant de communiquer avec l’ensemble (r)processeur-mémoire :

(v)le clavier, l’écran, le disque dur, le réseau, l’imprimante, scanners, etc…(/v)

(/r) Le processeur exécute sans fin la suite des opérations suivantes : (v)dite « boucle d’exécution ». A savoir : (r)

1) Lire une instruction en mémoire (Instruction Register)

2) Décoder l’instruction

3) Exécuter l’instruction

4) Calculer l’adresse de l’instruction suivante (mise à jour PC, Program Counteur)(/r)(/v)(/r)

L’unité de commande récupère les instructions de la mémoire et les analyses, puis séquence les actions élémentaires pour leurs réalisation. Elle orchestre donc les différentes actions à réaliser par le processeur et commande les interactions entre le chemin des données et la mémoire. Elle est composée d’un ou plusieurs automates séquentiel enchainant les actions à réaliser.

La partie opérative est au service de l’unité de commandes. Elle contient les outils pour réaliser les actions élémentaires ordonnées par l’unité de commande, notamment une unité arithmétique et logique que l’on note ALU et des registres internes. Un banc de registres désigne un ensemble de registres, Identifiés chacun par un nom ou un numéro, et pouvant stocker un mot binaire de n bits un registre n bit est composé de la mise en parallèle de ne registres 1 bit(registre 1 bit : cellule mémorisant élémentaire). Ces registres sont utilisés pour stocker les opérandes des opérations, ainsi que leurs résultat. Le chemin de données contient le banc de registres, l’ALU et la connectique permettant de relier les registres en entrée et sortie de l’ALU((r)multiplexeurs).(/r) La mémoire peut être vu comme un tableau de n lignes ou mot de p bit.

Schéma circuit

La mémoire peut être vue comme un tableau de n lignes ou mots de p bits.

L’index d’un mot définit son adresse :

Adresse du 1er mot=0, adresse du 2e = 1, …, adresse du dernier mot=n-1

Les mots sont en général des octets (soit p=8) ; on dit que l’unité adressable est l’octet. Les cases mémoires peuvent parfois mémoriser des mots de seize, trente-deux ou soixante-quatre bits. Les données et les instructions du programme en cours d’exécution sont stockés dans des zones différentes de la mémoire, notées « Données » et « Code » sur le schéma ci-dessus.

Il existe différents types de mémoire :

* La mémoire non réinscriptible ou ROM : assez lente d’accès, elle contient le code de démarrage ou l’ensemble du code dans certains petits systèmes embarqués.
* La mémoire vive ou RAM : volatile et rapide d’accès, elle contient le code et les données manipulées par le processeur pendant l’exécution d’un programme.
* La mémoire Flash : non volatile, réinscriptible (mais nombre d’écritures limitées) et rapide d’accès, elle est utilisée pour stocker du code ou les données non modifiables d’un programme ou bien dans les clés USB ou certains appareils numériques pour du stockage à long terme.

La capacité mémoire correspond au nombre de mots (octets) que la mémoire peut stocker. Un ordinateur ayant une capacité de stockage de 16 Go (C’est aujourd’hui, un ordinateur individuel ayant couramment plus d’un tera-octet de mémoires de soixante-quatre bits, il y en a 2147483648 pouvant chacune mémoriser un mot de soixantes-quatre bits.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 kilo-octet | 1 ko | 210 octets | 1 024 octets |
| 1 mega-octet | 1 Mo | 220 octets | 1 048 576 octets |
| 1 giga-octet | 1 Go | 230 octets | 1 073 741 824 octets |
| 1 tera-octet | 1 To | 240 octets | 1 099 511 627 776 octets |

Le processeur initie les transferts de données entre le processeur et la mémoire et indique à la mémoire : l’adresse du mot à transférer, la taille du mot à transférer, le sens du transfert (store ou load) et la donnée à écrire en cas d’écriture (store).

Lors d’un transfert de données : le décodeur d’adresse sélectionne la ligne correspondant à l’adresse demandée sur le bus d’adresse.

La commande indique l’opération : lecture ou écriture. Si il y a lecture, les données lues sont mises sur le bus de données ; s’il y a écriture : les données à écrire sont présentes sur le bus de données.

(r)b) Jeu d’instructions (/r)

La vue externe d’un processeur peut être définie par l’ensemble des instructions qu’il est capable de traiter. Ce jeu d’instructions est étroitement lié à l’architecture interne de l’ordinateur. Il précise aussi quels registres du processeur manipulables par le programmeur.

(r)Définition 1 : (JEU D’INSTRUCTIONS)(/r)

Le jeu d’instructions d’un processeur est défini par :

* L’ensemble des instructions qu’il peut effectuer
* Le système d’encodage de ces instructions en binaire.

Remarque :

Il existe de multiples jeux d’instructions, autant qu’il y a de types de micro processeurs. Les jeux d’instructions principaux des années 1980 à 2010 ont pour nom X86, X64, Power PC, SPARC, ARM, Alpha, VAX, 68000, MIPS.

Le X86 regroupe les microprocesseurs compatibles avec le jeux d’instruction du processeur (r)Intel 8086 (/r)(Cette série est nommée IA32 à partir (r)du pentium)(/r). Les processeurs fabriqués selon l’architecture MIPS, ont été utilisé dans les systèmes SGI et plusieurs systèmes embarqués comme les ordinateurs de poche, les routeurs Cisco et les consoles de jeux vidéo. Nous allons étudier un jeu d’instruction MIPS (r)(Microprocessor without interlocked pipeline stages)(/r)

Searche MIPS

Les type instructions : (Opcode 000 000) : <https://www.d.umn.edu/~gshute/mips/rtype.xhtml>

TYPE instructions A11 opcodes exception 000000,00001x,and 01 00xx

A récupérer sur école directe

DEF 2 (suite) :

Instructions en langage machine, une instruction est une commande formatée donnée au processeur qui définit :

* **Le traitement à effectuer maintenant**
* **La prochaine instruction à exécuter**

(r)Remarques : (/r)

Un traitement à effectuer est une opération (addition, opération logique, opération mémoire) accompagnée des opérandes sur lesquels elle porte (la ou les opérandes sources et l’opérande de destination s’il y en a une).

* La prochaine instruction à exécuter est celle implantée en mémoire à la suite de l’instruction courante (succession implicite) ou bien celle définie explicitement par des instructions de saut.

(r)Codage des instructions :(/r)

Le jeu d’instruction définit le format de codage binaire des instructions. La table donne le principe d’encodage de l’architecture MIPS32 où l’instruction à effectuer est associé à un mnémonique indiquant (r)l’opération, par exemple : add, audi, 1w, SW, j, beq. (/r) Ces mnémoniques sont convertis en codes binaires selon le format correspondants. L’opcode est écrit sur les six premiers bits(bits 31 à 26) Il permet de distinguer les instructions au format I ou J, pour celles qui sont au format R, il vaut 0. Les différents opérandes qui suivent l’Opcode sont soit des (r)constantes, nommées « immédiats »(/r) (-1, 0xFFFF, 3, etc…) soit des registres du processeur, identifiés par un numéro ($0, $1, …, $31), soit encore des codes fonctions.

(r)Exemples :(/r)L’instruction addi $1,$2, 3 permet d’ajouter l’immédiat 3 (en binaire (r)0000 0000 0000 00 11)(/r) au registre n°2 (r)(00010)(/r) et met le résultat dans le registre n°1 ((r)00 00 1)(/r). Cela est codé en binaire, selon le format I du MIPS32, (r)001 000 – 00001 – 00010 – 00 00 00 00 00 00 00 11 (/r). On a ajouté les tirets de séparation pour distinguer le codage de l’opération, le registre où stocker le résultat, registre où prendre (r)la donnée et immédiat (immédiat 3) à ajouter ; (/r) mais pour l’unité de commande et le processeur cette instruction est simplement écrit sur : 0010 0000 0010 0010 0000 0000 0000 0011 écrit sur 4 octets.

Pour le simplifier, on utilise parfois le système hexadécimal : dans ce cas, il faut regrouper les bits par 4 et traduire chaque quartet (r) par un caractère hexadécimal (/r) L’instruction addi $1, $2, 3 devient en quartets 0010-0000-0010-0010-0000-0000-0000-0011, d’où en hexadécimal (r)2022 0003.

Instructions de transfert de mémoire :(/r)

Ces instructions lisent((r)load)(/r) ou écrivent (r)(store)(/r). Des données en mémoire. La syntaxe des instructions d’accès mémoire est au format I, mais notée (r)opcode rt, l’immediate(rs).(/r) Le sens du transfert est défini par l’opcode : (r) lw(load word) ou sw(store word).(/r) L’adresse accédée est en définitive, obtenue en ajoutant le contenu du registre (r)rs(/r) à l’immédiat (r)immediate (/r) sur 16 bits (L’ALU est utilisé pour le calcul d’adresse). Le registre (r)rt(/r) contient la valeur à écrire en mémoire (s’il s’agit d’une écriture) ou recevra la valeur lue en mémoire (s’il s’agit d’une lecture).

(r)Exemple 2 :(/r)

* lw $4, 0($3) : lecture du mot de la mémoire contenu à l’adresse (r)0+$3(/r) et rangement dans le registre $4.
* sw$4, 0($3) : écriture du mot contenu dans le registre (r)$4 à l’adresse 0+$3 de la mémoire.(/r)

L’encodage MIPS32 de ces instructions suit le même principe que précédemment :

Le code opération de (r)lw est 100011, celui de sw est 101011(/r) ces instructions codées en hexadécimal sont :

(r)FICHIER CSV

Objectifs :

* Importation des tables(/r)
* Importer une table depuis un fichier texte tabulé ou un fichier CSV (r)
* Manipulation de fichiers CSV(/r)
* Lire un fichier CSV ou écrire dans un fichier CSV

Exercice 1 :

1. Créé un fichier avec blocnote (Wordpad sous windows et TextEdit sous mac) et recopier mes trois lignes suivantes : (Nom, Prénom, Classe).
2. Sauvegarder votre fichier (au format .docx) ouvrez -le avec open office calc ou excel (open office calc et/ou Excel)
3. Faire de même en renomant votre fichier .CSV.

Exercice 2 :

1. Créé un fichier nomé *Notes Eleves.csv* avec le bloc-notes contenant les données du tableau ci-dessous.

|  |  |  |  |
| --- | --- | --- | --- |
| NOM | MATHS | NSI | ANGLAIS |
| KUATE | 17 | 20 | 12 |
| ZHANG | 16,5 | 15 | 17,5 |
| LETELLIER | 09 | 18 | 17 |

1. Vérifier en ouvrant votre fichier texte avec open office Calc et/ou Excel.

Exercice 3 : faire un programme capable d’ouvrir ce fichier

Exercice 4 : écrire une fonction qui *convert(dico)* qui va créé un nouveau dictionnaire avec les notes des dictionnaires de notre table précédente converties en entiers.

Exercice 5 : Ecrire une fonction (r)stats\_csv(fichier)(/r) où le fichier est le nom d’un fochier csv et qui affiche le nombre de lignes et de nombre de colonnes de al table stockée dans le fichier.

(r)Ecriture (export) d’un CSV avec python(/r)

On peut écrire le contenu d’un dictionnaire dans un fichier csv. (r)With open(/r) permet d’éviter d’avoir à signifier la fermeture du fichier, elle se fera automatiquement à la fin (r)de l’indentation.(/r)

Exercice 5 : Ecrie une fonction (r)stats\_csv(fichier)(/r) où fichier est le nom d’un fichier csv et qui affiche le nombre de lignes et de colonnes de la table stockée dans le fichier.

(r)Les commandes (/r)

Taper (r)man(/r) <commande> pour obtenir la notice : Son nom (Name), sa syntaxe (Synopsis), un descriptif plus ou moin détaillé (Description), son auteur (Author), l’adresse où signaler une erreur (Reporting Bugs), le copyright et des liens (See also). Par exemple man ls nous donnera, dans son descriptif, (r)la liste des options possibles pour cette commande(/r).

(r)Premières commandes :(/r)

* Elle permet de savoir qui est connecté sur l’ordinateur (local et distant) ;.
* echo permet d’afficher du texte sur la sortie standard ( le terminal)
* Expr permet d’évaluer une expression entre deux entiers avec un opérateur ((r)+, -, \*, /, %) ou avec un comparateur (<, <=, >, >=, =, !=), (/r) certains caractères ayant une signification particulière, il faut les échapper par un anti-slash : pour obtenir le résultat de 3\*5, on tapera expr 3\\*5.
* Ls affiche le contenu du répertoire courant, tandis que ls<repertoire> adffiche le contenu dur répertoire spécifié.
* Pwd permet de savoir le chemin du répertoire courant ; au login, nom du répertoire courant et le répertoire utilisateur (/home/om en ce qui me cocerne).
* cd permet de changer de répertoire (directory) : ou indique le chemin absolu ou relatif du répertoire vers lequel on souhaite se diriger.

Ex :

Une entreprise a vingt ordinateur, pour identifier chaque machine d’une manière unique, combien de bit faut-il ?

Il nous faut cinq bit pour l’identificateur de machine car 25 = 32 qui est plus grand que vingt.

L’administrateur doit demander un bloc CIDR, un bloc CIDR de 27 bits.

Si l’entreprise obtient l’adresse réseau, (125.93.64.96) avec un préfix réseau sur 27 bits. (r)L’adresse est composée de la partie réseau (netid) et de 27 bits suivi d’une partie et d’une partie machine (hostid) null sur 5 bits :

(/r)L’adresse réseau 125.93.64.96 (r)sous forme binaire est : 125 :26+25+24+23+22+20=01111101

93 :26+24+23+22+20=01011101

64 :26=01000000

96 :26+25=01100000

01111101.01011101.01000000.01100000

Les 5 dernier bits sont différents et permettent d’aligner toutes les machines.

Les adresses ayant le bit du suffixe (mis tous à 0 et tous à 1) ne peuvent pas être attribué à une machine. C’est adresses sont réservées pour des but spécifiques. Si les bits du suffixe machine sont tous null il s’agit de l’adresse du réseau. Si les bits de l’adresse machine sont tous également à 1 il s’agit de l’adresse de diffusion dans le réseau. Un paquet IP avec l’adresse de diffusion comme l’adresse de destination est destiné à toutes les machines du réseau. Pour extraire l’adresse réseau à partir d’une adresse IP, il faut connaître sont masque de sous-réseau ((r)SubnetMask(/r)) il indique sur combien de bits est codé le préfixe réseau. Considérons le réseau 142.68.2.0 ayant un préfixe réseau codé sur 24 bits, Il reste 8 bits pour identifier les machines du réseau. On dira que son masque de sous-réseau sera : 11111111.11111111.11111111.00000000 soit en notation 255.255.255.0 en notation décimale. L’adresse du réseau est identifiée par l’adresse du réseau 142.68.2.0 et (r) le masque 255.255.255.0(/r) Une fois obtenu l’adresse réseau, l’administrateur réseau doit, en général, organiser sou réseau. Les sous réseau sont interconnectés par des routeur qui sont installé et configurés par l’administrateur réseau. S’il y a quatre sous réseaux, il faut lui prendre deux bits (de 0 à 3, 2² ce qui fait quatre numéros), et pour cinq sous réseaux il lui faudra en prendre 3. La taille du suffixe machine qui set à identifier les macines dans chaque sous réseau va de plus en plus se réduire.

Le mask peut être représenté de deux manières :

* sous la forme d’une adresse IP (sur une suite de 32bit, mais vec les bits de la partie préfixe réseau tous mis à 1 et les bit de la partie machine tous mis à 0). Si l’on considère le réseau du réseau, le mask de sous réseau sera : 11111111.11111111.11111111.00000000 soit 255.255.255.0

(r)Interractioon homme-machine :

Objectif : 1) (/r) Les modalités des interactions homme-machine qui tiennent principalement à l’effet conjoint des composants graphiques mis en place par le code html. Des dispositifs de style prévu par le code CSS ; du dynamisme apporté par le code javascript.

(r)2) L’interaction client-serveur (/r) que se soit par l’intermédiaire d’une requête (r)http(/r) ou l’envoie d’un formulaire, pose toujours des questions relevant de la sécurité.

Plan du cour :

Nous nous contenterons d’une approche des trois langages utilisé dans une page web chez le client. Le premier est html pour le texte brut et les principales balise qui permettent de le structurer.

Le CSS pour la mise en forme du texte selon les balises qu’il contient et les actions de l’utilisateur.

Le javascript pour la réaction au évènements déclenché par les actions de l’utilisateur.

L’interaction avec un serveur distant est rtraité principalement coté client réservant les mécanismes du serveur. Le code php, les requêtes sql vers les bases de données et no sql.

Introduction historique : Le web (la toile) désigne un système donnan accès à un ensemble de données reliée par les liens hypertexte et accessible sur le réseau internet. Les textes, photo vidéos, graphiques, les son, les graphiques qui circulent par le web sont régulées par le W3C ((r)World Wide Web Consortium(/r)). En 1965, invention du concept hypertexte par Théodore Holm Nelson. 1989 création du World Wide Web et de l’html par Tim Bernes-Lee (1955-). Eric Bina (1964-) cofondateur de Netscape. Inventeur du langage javascript (Brendan Eich).

(r)I) Structuration html :(/r)

Les suites de caractères, encodés en ascii, en latin-1 (ISO-8859) ou en utf-8 sont des textes brut. Les logiciels de traitement de texte les enrichissent pour introduire des qualifications particulière (ex : le soulignage, l’italique, le gras, le barré), il permet aussi de les structurés (en les subdivisant en parties chapitres, en paragraphes ou en format de tables ou de listes), il peut apporteer des informations sur le texte. L’auteur du texte, la date, le lieu, il peut créé des liens entre certaines parties du texte ou avec d’autres texte. Il peut ajouter des images ou même des vidéos. Les formats spécifiques de ces textes enrichit peuvent être fermés, le proprio en garde le secret ou alors ils sont ouverts (libre office).

Ainsi, les pages web sont écrites dans le langage de balises HTML (Hyper Text Markup Language). Les balises vont deux par deux comme les parenthèses : après une balise ouvrante <namebalise> après une balise fermante </namebalise>. On peut emboiter des balises les unes dans les autres mais pas les fairent se chevaucher. Si on veut metre un passage en italique on mettra <i> et </i>, pour du gras <B> et </B>, pour le souligné <U> et </U>. Le texte html s’écrit HTML(<i><b>H</b>yper<b>T</b>ext <b>M</b> arkup <b>L</b>angage</i>)

La syntaxe générale d’une balise est <name attribut1=’valeur’ attribut2=’valeur’ …> ou name est le nom de la balise et attribut1, attribut2 sont des attributs, on peut aussi dire des options dont il faut choisir les valeurs. La balise <html> admet un seul attribut alors langue pour indiquer que la langue utilisé est soit fr pour du français <html langue=« fr »>. Cela peut servir dans certains cas, pour lire la page avec un logiciel de synthèse vocale. Dans d’autres cas, la liste des attributs possibles est plus importante. La balise d’entête <meta> en accepte 19 dont 4 seulement sont spécifiques à cette balise (name http-equiv, content et charset), les autres pouvant se retrouver dans d’autres balises (acceskey, class, contenteditable, dontextmenu, dir, draggable, dropzone, id, lang, spelcheck, style, tabindex, title et translate).

Il est assez instructif de consulter le code html d’une page web existante. Il y a généralement un outil de votre navigateur qui le permet : sur mozilla Firefox, il faut ouvrir le <<menu>> en haut à droite (l’icône consiste en trois traits horizontaux superposés), choisir <<Développement web>>, puis <<Code source de la page>> [David Roche](https://pixees.fr/informatiquelycee/n_site/nsi_prem_js.html)

(<https://pixees.fr/informatiquelycee/n_site/nsi_prem_js.html>)

2) Catalogue de Balise d’entête

<title>, …</title> : Titre de l’onglet

<style> …</style> : Permet d’inclure du code CSS dans la page.

<link/> : Permet d’indiquer certaines informations comme :

- Inclure une page de style <link rel= « stylesheet » type=« text/css » href= « style.css »/>

- Déclarer une icone pour le site : <link rel=« sortcut icon » type=« image/x-icon » href=« icon.ico »/>

<script>…</script> : permet d’inclure un script comme : \*Du code : <script type=« text/javasript »>…</script> ;

\*Un fichier script :<script type=« text/javascript » src=« script.js »/>

<meta …> : qui permet de définir différentes propriétés comme :

- L’auteur de la page : <meta name=« author » content=« obele julien »/>

- La description de la page : <meta name=« description » content=« Cours NSI »/>

- Les mots-clés de la page : <meta name=« keywords » content=« Web HTML,CSS,Javascript »/>

- Une adresse de contact : <meta name=« reply-to » content=« js.obele@free.fr »/>

- La spécification du système d’encodage des caractères : <meta charset=« utf-8 »/>

- L’instruction d’empêcher la mise en cache de la page : <meta http-equiv=« pragma » content=« no-cache »/>

NB : Pour obtenir des propositions plus complètes, je vous conseille de vous rendre sur le site de Jacques MARTINET (<https://jaetheme.com/balises-htm15/#html>) ou sur le site de david ROCHE.

Balises de contenus :

<p>…</p> : Paragraphe

<pre>..</pre> : Paragraphe affiché tel qu’il a été tapé dans le code (avec les espaces)

<h#>…</h#> : titre de niveau : <h1> (le plus important), …, <h6> (le moins important).

<a>…</a> : permet de pointer vers une page d’adresse url <a href= « url »>…</a>.

Un nom précédé par un dièse (#) indique une cile interne au document (u ID)

La cible est repérée grâce aux balises <a name= »nomCible »>…</a>

Pour charger la réponse dans une fenêtre utiliser l’option target= »-blank »

<br> : passage à la ligne

<hr> : la ligne de séparation horizontale.

<img> : insertion d’une image :

- <img src=« lien de l’image/image1.png » alt=« coucher de soleil sur la banquise »/> insère l’image « image1.png » situé dans le dossier image du même dossier de la page.

La valeur de alt est le texte alternatif utilisé lorsqu’il est impossible d’afficher l’image.

Les attributs width et height donne les dimensions de l’image en pixels ce qui permet de redimensionner. Par exemple : <img src= « image.png » width=« 200 » heigth= « 200 »/>

<div>…</div> : création d’un bloc de contenu de type blok (les div se positionnent les un au-dessous des autres comme h1,h2,p,pre,ul,li,tab,etc…)

<span>…</span> : Création d’un bloc de contenu de type in-line (les span se positionnent les uns à sôté des autres comme a, img, em, input, label, etc…)

<sup>…</sup> : la mise en exposant

<sub>…</sub> : la mise en indice

<em>…</em> : passage sur lequel porte une emphase accentuée. Exemple de leins interne et externe : l’appel à la note du bas est ici doublé par un appel retour du bas vers le haut.

**(mécanisme permettant de faciliter la lecture d’une note de bas de page sans perdre le fil du texte).**

**NB : dans un long texte il y a parfois des indications à apporter comme une référence ou une indication. Cette indication créant une sorte de rupture dan le texte peut-être reportée en bas de page, ce que nous avons fait ici.**

**De nombreux exemples peuvent se voir sur des pages web.**

**Listes et tableaux :**

<ul>…</ul> : liste à puces non numérotée

<ol>…</ol> : Liste à puces numérotée (option par défaut type=« 1 »)

\*Ajouter l’option type=« A » pour une numérotation alphabétique majuscule ;

\*Ajouter l’option type=« I » pour une numération romaine majuscule.

<li>…</li> : Item de liste à puce (numéroté ou non).

<dl>…</dl> : Liste de définitions.

<dt>…</dt> : Titre de la définition

<dd>..</dd> : Définition

<table>…</table> : Déclaration d’un tableau

<caption>…</caption> : Donner le titre du tableau

<tr>…</tr> : La ligne du tableau

<td>…</td> : La case du tableau

(<th>…</th> : Pour une case d’entête (par défaut en gras)

\*<td colspan=n> : La case occupe n colonne

\*<td rowspan=n> : La colonne occupe n lignes

Extrait de page concernant les listes et tableaux. Une mise en forme plus poussée du tableau relèverait d’éléments de style que volontairement nous n’avons pas implémentés ici (pour les réserver aux feuilles de style css que nous étudierons plus loin), exceptée l’option border=5 qui ajoute une bordure.

Trois types de listes :

Liste non numérotée :

* premier élément de ma liste non numéroté
* second élément de ma liste non numéroté

Liste numérotée

* 1er élément de ma liste numéroté
* 2ème élément de ma liste numéroté

Liste de définition :

* Définition 1

Texte de la définition 1

* Définition 2

Texte de la définition 2

Exemple de tableau simple

données commentaires

donnée 1 commentaire 1

donnée 2 commentaire 2

Formulaires :

<form>…</form> : déclaration d’un formulaire

Les attributs obligatoire de cette balise sont :

\*method=« post » ou method=« get », selon le type de transmition que l’on souhaite (avec get, on est limité dans la quantité de données pouvant être envoyées et surtout, celles-ci vont être envoyées en clair).

\*action=… pour préciser l’adresse de destination du formulaire (par exemple, une page php du serveur qui effectuera un traitement et retournera une réponse au client).

<input…/> : champ d’un formulaire

<input.../>: champ d'un formulaire

l'attribut type en html

nom d'attribut : accept, accept-charset, accesskey, action, align, allow, alt, async, autocapitalise, autocomplete, autofocus, autoplay, background.

éléments auxquels s'appliquent ces noms d'attributs:

accept: <form>, <input>: la description: la liste des types acceptés par le serveur, il s'agit généralement de types de fichier;

accept-charset: <form>: description de la liste des jeux de caractères pris en charge

accesskey: attribut universel: description: cet attribut permet de définir un raccourci clavier pour activer un élément ou lui passer le focus;

action: <form>: descritpion: l'url d'un programme qui traite les informations envoyées par le formulaire.

action: <applet>, <caption>, <col>: description: idem

align: <colgroup>, <hr>, <iframe>, <img>, <table>, <tbody>, <td>, <thead>, <tfoot>, <tr>: description: cet attribut définit l'alignement horizontal de l'élément

allow: <iframe>: description: défini les règles des fonctionnalitées pour cette iframe

alt: <applet>, <area>, <img>, <input>: description: un texte alternatif à afficher lorsque l'élément ne peut pas être affichée.

async: <script>:description: cet attrribut indique que le script doit être éxécuté de façon asyncron.

autocapitalise: attribut universel: cet attribut controle la façon dont un champs texte est saisi en majuscules de façon automatique.

autocomplete: <form>, <input>, <textearea>: description: cet attribut indique que le comtrole peut être remplie automatiquement

L’atribut type en html

Formulaires<form> ... </form> : Déclaration d'un formulaire. Les attributs obligatoires de cette balise sont◆ method="post" ou method="get", selon le type de transmission que l'on souhaite (avec get, on est limité dans la quantité de données pouvant être envoyées et surtout, celles-ci vont être envoyées en clair).◆ action=... pour préciser l'adresse de destination du formulaire (souvent une page PHP d'un serveur qui effectuera un traitement et retournera une réponse au client).<input .../> : Champ d'un formulaire (associé ou non à un label). L'attribut type est obligatoire :

◆ type="text" : le champ est une 1one de texte

◆ type="password" : le champ est une 1one de texte où les caractères sont masqués

