Séance 4: Encodage et décodage

Université Paris-Diderot

Objectifs:

- Comprendre l'encodage des nombres entiers en binaire et en hexadécimale
- Manipulation de chaînes de caractères

Le but de ce TP est tout d'abord de comprendre l'encodage en binaire des entiers naturels et relatifs et les différentes formes d'encodage. Afin de rester dans un cadre simple, les encodages seront donnés sous forme de chaînes de caractères.

Nous verrons aussi comment encoder et décoder une chaîne de caractères à l'aide d'un codage transformant une chaîne de caractères en une autre chaîne.

Pour ce TP, il vous est demandé de faire un fichier par section (c'est à dire un fichier pour la section 1 et un pour la section 2). Il vous est également demandé de tester chaque fonction que vous programmerez.

Rappel : Tous les fichiers que vous sauvegarderez ou créerez pour ce TP devront être mis dans le répertoire `TP4` sous-répertoire du répertoire `IP1-Python` que vous avez créé au premier TP .

1 Encodage/Décodage de textes

Exercice 1 (Le chiffrement de César, ⋆⋆)
On souhaite utiliser une méthode connue pour encoder des chaînes de caractères qui s'appelle le chiffrement de César. Le chiffrement de César correspond à un décalage de lettre. On donne la lettre qui correspond à 'A' et on peut en déduire toutes les autres. Par exemple, si on dit qu'à 'A' correspond 'C' alors à 'B' on associe 'D', à 'C' on associe 'E', . . . , à 'Y' on associe 'A' et à 'Z' on associe 'B'. Ainsi dans l'encodage où à 'A' correspond 'C', le mot 'BONJOUR' s'encode 'DQPLQWT'. On supposera dans cet exercice, que les espaces ne sont pas codés et que toutes les chaînes de caractères sont faites uniquement d'espaces et de lettres majuscules.

1. *Écrire une fonction* `cesarLetter` *qui prend en paramètre une chaîne de caractères contenant le caractère correspondant à 'A' et une chaîne de caractères contenant un autre caractère et renvoie la chaîne de caractères codant ce second caractère. Si le second caractère est un espace, alors la fonction renvoie la chaîne* `" "`*. On supposera que les paramètres auront toujours le bon format (ce n'est pas la peine de le vérifier).*

   Contrat:

   - `cesarLetter("D","B")` *donne "E"*
   - `cesarLetter("D","C")` *donne "F"*
   - `cesarLetter("Z","C")` *donne "B"*

   Indice : *Commencer par créer une chaîne de caractères* `"ABCDEFGH...XYZ"` *avec toutes les lettres, ensuite comptez la distance dans cette chaîne du premier paramètre par rapport à 'A' (par exemple*

pour "D" il s'agit de 3) ensuite cherchez le deuxième paramètre dans cette chaîne et le résultat se trouve à la distance calculée précédemment. Attention : quand on arrive à la fin de l'alphabet, il faut compter modulo 26. Ainsi si le premier paramètre est `"Z"` *la distance par rapport à 'A' est 25, et le code pour* `"C"` *se trouve à la position 2+25 modulo 26 ce qui vaut 1 et qui est la position de 'B' dans la chaîne (2 étant la position de 'C').*

2. *Écrire une fonction* `cesar` *qui prend en paramètre une chaîne de caractères contenant le caractère correspondant à 'A' et une chaîne de caractères* `s` *(ne contenant que des lettres majuscules et des espaces) et renvoie la chaîne de caractères correspondant au chiffrement de César de* `s`.

   Contrat:

   — `cesar("D","BONJOUR")` *donne "ERQMRXU"*

3. *Écrire une fonction* `deCesarLetter` *qui prend en paramètre une chaîne de caractères contenant le caractère correspondant à 'A' et une chaîne de caractères correspondant au codage d'un caractère et renvoie le caractère original associé à ce second caractère. Si le second caractère est un espace, alors la fonction renvoie le caractère espace. On supposera que les paramètres auront toujours le bon format (ce n'est pas la peine de le vérifier).*

   Contrat:

   — `deCesarLetter("D","E")` *donne "B"*
   — `deCesarLetter("D","F")` *donne "C"*
   — `deCesarLetter("Z","B")` *donne "C"*

   Indice : *Utiliser une technique similaire à celle indiquée à la question 1.*

4. *Écrire une fonction* `deCesar` *qui prend en paramètre une chaîne de caractères contenant le caractère correspondant à 'A' et une chaîne de caractères* `s` *(ne contenant que des lettres majuscules et des espaces) correspondant à un message chiffré en César et renvoie la chaîne de caractères originale.*

   Contrat:

   — `deCesar("D", "ERQMRXU")` *donne "BONJOUR"*

5. *Dans l'encodage où à 'A' on associe 'D', qu'a voulu dire Jules César avec ce message secret 'DOHD MDFWD HVW' ?*

□

2 Encodage des entiers en binaire

Exercice 2 (Encodage des entiers naturels non-signés, ⋆)
Dans cet exercice. on s'intéresse à l'encodage en binaire des entiers naturels sur un octet. On rappelle qu'un octet contient 8 bits valant 0 ou 1. Les octets seront représentés par une chaîne de caractères de longueur 8 où le bit de poids faible sera situé le plus à droite. Ainsi une chaîne de caractères `"`$a_0a_1a_2a_3a_4a_5a_6a_7$`"` *composée uniquement de 0 et de 1 correspondra à l'entier :* $a_0*2^7+a_1*2^6+a_2*2^5+a_3*2^4+a_4*2^3+a_5*2^2+a_6*2+a_7$. *Par exemple, l'encodage de l'entier 5 en binaire sera donné par la chaîne* `"00000101"`. *Avec cet encodage on peut donc coder les entiers de* 0 *à* 255.

1. *Écrire une fonction* `isBinaryEncoding` *qui prend en paramètre une chaîne de caractères et teste si elle correspond à un octet, c'est à dire si sa taille est 8 et si elle ne contient que des* `"0"` *ou des* `"1"`. *Dans le cas positif elle renverra* `True` *et sinon* `False`.
2. *Écrire une fonction* `powerTwo` *qui prend un argument un entier* `n` *(supposé positif) et renvoie* 2^n.
3. *Écrire une fonction* `decode` *qui prend en paramètre une chaîne de caractères, et si cette chaîne correspond à un octet renvoie la valeur entière correspondante et renvoie -1 sinon.*

   Contrat:

   — `decode("11111110")` *donne* 254
   — `decode("10001000")` *donne* 136

4. *Écrire une procédure* `encodeAndPrint` *qui prend en paramètre un entier (supposé positif) et affiche son encodage en binaire à l'envers et va à la ligne. Si l'entier est négatif ou supérieur ou égal à 256, cette procédure va simplement à la ligne sans rien afficher.*

   Contrat:

   — `encodeAndPrint(254)` *affiche* `01111111`
   — `encodeAndPrint(136)` *affiche* `00010001`

   Indice : *Si l'on cherche à connaître l'encodage en octet* `"`$a_0a_1a_2a_3a_4a_5a_6a_7$`"` *d'un entier* `n`*, il est utile de remarquer que* a_7 *vaut* `n % 2`, a_6 *vaut* `(n // 2) % 2`, a_5 *vaut* `((n // 2))//2 % 2` *etc.*

5. *Écrire une procédure* `encode` *qui prend en paramètre un entier (supposé positif) et renvoie une chaîne de caractères contenant un octet correspondant à son encodage en binaire. Si l'entier est négatif ou supérieur ou égal à 256, cette procédure renverra la chaîne vide* `""`*.*

   Contrat:

   — `encode(253)` *donne* `"11111101"`
   — `encode(15)` *donne* `"00001111"`

6. *Selon vous, à quoi doit être égal* `decode(encode(x))`*. Vérifier que cela est vrai pour différentes valeurs de* `x` *prises entre 0 et 255. Vous pouvez par exemple tester dix valeurs différentes prises au hasard en utilisant la fonction* `randrange` *vue au TP3.*

□

Exercice 3 (Complément à un, ⋆⋆)

Un autre codage des entiers en binaire, similaire à celui proposé à l'exercice précédent, est obtenu en changeant les 0 en 1 et les 1 en 0. Il s'agit de l'encodage connu sous le nom de complément à un. Ainsi dans l'encodage complément à un, l'octet correspondant à 253 s'écrira `"00000010"`*.*

1. *Écrire une fonction* `oneComplement` *qui prend en paramètre une chaîne de caractères et retourne une chaîne de caractères identique dans laquelle chaque 0 est remplacé par un 1 et chaque 1 par un 0. Remarque : ici les chaînes ne comportent pas nécessairement que des 0 ou des 1 et ne sont pas nécessairement de longueur 8.*

   Contrat:

   — `oneComplement("010")` *donne* `"101"`
   — `oneComplement("bob1081")` *donne* `"bob0180"`

2. *En utilisant les fonctions de l'exercice 1 et la fonction précédente, donnez une fonction* `encodeOneC` *qui prend en paramètre un entier et renvoie une chaîne de caractères représentant un octet contenant l'encodage binaire en complément à un de l'entier. Donnez également une fonction* `decodeOneC` *qui fait l'opération inverse, c'est-à-dire donne l'entier correspondant à un octet encodé en complément à un.*

   Contrat:

   — `encodeOneC(253)` *donne* `"00000010"`
   — `encodeOneC(15)` *donne* `"11110000"`
   — `decodeOneC("11101101")` *donne* `18`

□

Exercice 4 (Encodage des entiers relatifs, ⋆)

Afin de coder des entiers relatifs, une technique consiste à utiliser le premier bit de l'octet comme bit de signe. Si ce bit vaut 1 alors l'entier est négatif et sinon il est positif. Avec cette technique sur un octet, on peut coder les nombres allant de 127 à -127. Avec le codage par chaîne de caractères de longueur 8 pour représenter des octets, l'entier 11 sera codé par la chaîne `"00001011"` *et l'entier -11 par la chaîne* `"10001011"`*.*

1. *Écrire une fonction* `isNegative` *qui prend en paramètre une chaîne de caractères correspondant à un octet et renvoie* `True` *si la chaîne est l'encodage d'un entier négatif et* `False` *sinon. On supposera dans cette question que la chaîne donnée en paramètre correspond toujours à un octet.*

2. *Écrire une fonction* `decodeNeg` *qui prend en paramètre une chaîne de caractères et qui renvoie l'entier relatif correspondant si cette chaîne correspond à l'encodage d'un entier relatif et renvoie -128 sinon.*

   Contrat:

   — `decodeNeg("00000110")` *donne 6*
   — `decodeNeg("11000001")` *donne -65*

3. *Écrire une fonction* `encodeNeg` *qui prend en paramètre un entier relatif et qui renvoie la chaîne de caractères correspondant à son encodage binaire. Si l'entier donné en paramètre n'est pas compris entre -127 et 127, la fonction renverra la chaîne vide* `""`*.*

   Contrat:

   — `encodeNeg(-17)` *donne "10010001"*
   — `encodeNeg(9)` *donne "00001001"*

4. *Écrire une procédure qui ne prend pas de paramètre et qui affiche* `Tout va bien` *si pour les entiers de 0 à 127, les fonctions* `encode` *(de l'exercice 1) et* `encodeNeg` *renvoient les mêmes valeurs et qui affiche* `Il y a un souci` *si pour au moins un entier, ces fonctions ne retournent pas la même valeur. (*Attention : *il ne faut pas afficher 128 fois* `Tout va bien` *mais juste une fois).*

□