

Séance 4 : CODAGES ET REPRÉSENTATIONS

L1 – Université Côte d'Azur

Exercice 1 – Représentations binaires

1. Traduire 116 en binaire. On utilisera l'algorithme vu dans le premier cours.
2. En déduire la représentation de -116 sur 8 bits.
3. On considère le nombre 110 0001 sur 8 bits. Quel est son signe? Combien vaut-il? On appliquera là encore l'algorithme vu en cours.
4. Même question avec 1001 1111

Exercice 2 – Représentations en bases quelconques

1. Qu'est-ce le code ASCII? Unicode? et l'encodage UTF-8?
2. Quel est le numéro de la lettre "r" en ASCII.
3. Comment afficher en une boucle `for` le code des chiffres de '0' à '9'?
4. Écrivez une fonction `liste_chiffres()` qui renvoie la chaîne contenant tous les "chiffres" '0...9A...Z'. Ces chiffres-là permettent de représenter les nombres jusqu'à quelle base?
5. Écrivez une fonction `écrire_nombre(n, b)` qui renvoie la représentation d'un nombre `n` dans une base `b`.

```

1 >>> écrire_nombre(17, 2) # 1×24 + 1
2 '10001'
3 >>> écrire_nombre(17, 10) # 1×10 + 7
4 '17'
5 >>> écrire_nombre(17, 16) # 1×16 + 1
6 '11'
```

6. Écrivez une fonction `valeur(ch)` qui renvoie le nombre correspondant au caractère `ch`. Si `ch` n'est pas un caractère valide, on renverra `None`.

```

1 >>> valeur('1')
2 1
3 >>> valeur('7')
4 7
5 >>> valeur('A')
6 10
7 >>> valeur('G')
8 16
```

7. Déduisez-en une fonction `lire_nombre(chaîne, b)` qui à partir d'une représentation en base `b` renvoie la valeur numérique associée.

```

1 >>> lire_nombre("10001", 2)
2 17
3 >>> lire_nombre("17", 10)
4 17
5 >>> lire_nombre("11", 16)
6 17
```

Exercice 3 – Couleur RGB

En html les couleurs peuvent être définies en hexadécimal, par exemple '#FF12E4' où FF, 12 et E4 correspondent aux trois composantes de rouge, de vert et de bleu. Écrire une fonction `couleur_rgb(r,g,b)` qui prend trois entiers en paramètres compris entre 0 et 255 et qui **renvoie** la chaîne de caractère correspondante de la forme '#RRVVBB' où chaque composante est codée sur un nombre à deux chiffres hexadécimaux.

Exercice 4 – Détecter l'ASCII

1. Écrivez la fonction `décoder(chaîne)` qui renvoie le tableau des numéros ASCII de chaque caractère de chaîne.

```
1 >>> decoder("bonjour camarade !")
2 [98, 111, 110, 106, 111, 117, 114, 32, 99, 97, 109, 97, 114, 97, 100, 101, 32, 33]
```

```
1 def decoder(chaîne):
2     L = [0] * len(chaîne)
3     for i in range(len(chaîne)):
4         L[i] = ord(chaîne[i])
5     return L
```

2. Déduisez-en une fonction `est_ascii(ch)` qui indique si la chaîne ne contient que des caractères ASCII.

```
1 >>> est_ascii("bonjour camarade !")
2 True
3 >>> est_ascii("ΞΙΝΟ ΑΑΧΑΝΟ") # Quelle est la dernière lettre de la chaîne ?
4 False
```

```
1 def est_ascii(chaîne):
2     L = decoder(chaîne)
3     for i in range(len(chaîne)):
4         if L[i] > 127:
5             return False
6     return True
```

Pour info, la lettre est un omicron majuscule grec!

3. Écrivez une fonction `recoder(L)` qui renvoie la chaîne ASCII correspondant à la liste L. Tous les caractères non-ASCII seront remplacés par "?".

```
1 >>> français = decoder("bonjour camarade génial !")
2 >>> grec = decoder("ΞΙΝΟ ΑΑΧΑΝΟ")
3 >>> recoder(français)
4 'bonjour camarade g?nial !'
5 >>> recoder(grec)
6 '???? ???????'
```

```
1 def recoder(liste):
2     c = ""
3     for i in range(len(liste)):
4         if liste[i] < 128:
5             c = c + chr(liste[i])
6         else:
7             c = c + "?"
8     return c
```

Exercice 5 – Encoder le volume sonore

1. Dans le code ASCII, comment passer en un seul calcul d'une lettre minuscule à une lettre majuscule ?
2. Écrivez la fonction `HURLER(chaîne)` qui renvoie la chaîne donnée en argument où chaque minuscule est remplacée par une majuscule.

```
1 >>> HURLER("Parle plus fort j'entends rien !")
2 'PARLE PLUS FORT J'ENTENDS RIEN !'
```

```
1 def HURLER(chaîne):
2     r = ""
3     for i in range(len(chaîne)):
4         d = ord(chaîne[i])
5         if ord('a') <= d and d <= ord('z'):
6             d = d - ord('a') + ord('A')
7         r = r + chr(d)
8     return r
```

3. De même, écrivez la fonction `murmurer(chaîne)` qui renvoie la chaîne tout en minuscule.

```
1 >>> murmurer("CHUT... Le Prof nous regarde...")
2 'chut... le prof nous regarde...'
```

```
1 def murmurer(chaîne):
2     r = ""
3     for i in range(len(chaîne)):
4         d = ord(chaîne[i])
5         if ord('A') <= d and d <= ord('Z'):
6             d = d - ord('A') + ord('a')
7         r = r + chr(d)
8     return r
```

4. Pourquoi vos fonctions sont fausses ? En Python, comment faudrait-il faire pour gérer proprement les majuscules et minuscules.

Notre fonction ne gère que l'ASCII et non les caractères accentués. En vrai, il faut utiliser les méthodes `"été".upper()` et `"ÉTÉ".lower()`

5. Écrivez une fonction `title(chaîne)` qui met la première lettre de chaque mot de la phrase en capitale (comme c'est l'usage dans les titres en anglais).

```
1 >>> title("the beauty and the beast")
2 'The Beauty And The Beast'
3 >>> title("the beauty and spider-man (first spin-off)")
4 'The Beauty And Spider-Man (First Spin-Off)'
```

```
1 def title(ch):
2     ancien = False # Le caractère précédent est-il une lettre ?
3     res = ""
4     for i in range(len(ch)):
5         if not ancien: # Si je suis en début de mot
6             res = res + ch[i].upper()
7         else:
8             res = res + ch[i]
9         ancien = est_lettre(ch[i])
10    return res
```

Exercice 6 – Interpréter une chaîne de caractères

On se donne une chaîne de caractères représentant une liste de nombres. Écrivez une fonction `chaîne_vers_liste(ch)` qui renvoie la liste associée.

```
1 >>> chaîne_vers_liste("[12,23, 24, 2 , 1]")
2 [12, 23, 24, 2, 1]
```

```
1 def est_chiffre(c):
2     return ord("0") <= ord(c) and ord(c) <= ord("9")
3
4 def nettoyer(ch):
5     """On ne conserve que les chiffres et les virgules"""
6     rés = ""
7     for i in range(len(ch)):
8         if est_chiffre(ch[i]) or ch[i] == ",":
9             rés += ch[i]
10    return rés
11
12 def nombre_de_virgules(ch):
13    n = 0
14    for i in range(len(ch)):
15        if ch[i] == ",":
16            n = n+1
17    return n
18
19
20 def chaîne_vers_liste(ch):
21    ch = nettoyer(ch)
22    # On gère à part le cas de la liste vide
23    if ch== "":
24        return []
25
26    # Dans le tableau il y a une case de plus que de virgules
27    tab = [0] * (1+nombre_de_virgules(ch))
28    # k est l'indice courant du tableau
29    k = 0
30    # courant est la chaîne contenant les caractères du prochain nombre
31    courant = ""
32
33    for i in range(len(ch)):
34        if ch[i] == ",":
35            tab[k] = lire_nombre(courant, 10) # cf. exercice 2
36            courant = ""
37            k = k+1
38        else:
39            courant = courant + ch[i]
40    tab[k] = lire_nombre(courant, 10) # cf. exercice 2
41    return tab
42
```