TP1 : Conversion et Python

Nous allons introduire ici le cours et les TP d'algorithmique en BTS SIO1.
L'objectif de ces 4 TPs d'introduction est de brosser certaines des possibilités que nous avons avec Python et comment mettre en œuvre notre réflexion afin de répondre à une problématique.

Objectif de ce TP :
- Créer un programme qui permet de gérer la conversion de nombre vers la base 10.
- Prendre contact avec les principales instructions de base en Python

Remarque :
Je recommande la lecture du document base en python avant de le faire, surtout si vos bases sont réduites…

Nous avons vu dans le cours, que la conversion vers la base 10 n'est pas un calcul complexe en soi.
Nous allons utiliser la formule suivante pour un nombre qui s'écrit $(a_n a_{n-1} \dots a_1 a_0)_b$:

$$(a_n a_{n-1} \dots a_1 a_0)_b = a_n \times b^n + a_{n-1} \times b^{n-1} + \dots + a_1 \times b^1 + a_0 \times b^0$$

Exemple :

$$(1101)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (13)_2$$

Identification des besoins de notre programme :
- Le nombre à convertir ;
- Sa base.

À partir de là, vous allez choisir selon votre niveau, le déroulé de TP que vous voulez…

Table des matières

Niveau expert :

Réaliser une fonction `conversions_vers_10` qui prendra comme arguments le nombre et la base de départ et retournera l'écriture de ce nombre en base 10.

Niveau Newbie :

Mise en place :

Vous allez ici être guidé. Il y aura des phases de recherche et si vous ne trouvez pas, les réponses se situeront en bas de page.
Des « erreurs » seront aussi faites, et nous les corrigerons. Cela permettra de voir certains messages d'erreurs.

Nous allons tout d'abord créer un script qui demandera à l'utilisateur le nombre puis la base et ensuite il convertira notre nombre et ensuite nous le transformerons en fonction.

Il faut bien comprendre le déroulé de notre programme.
Pour les demandes à l'utilisateur, nous allons utiliser `input`.

```
N = input("Quel est le nombre à convertir? ")
k = input("Quelle est la base de départ? ")
```

Nous avons donc à ce stade, dans la variable `N` le nombre à convertir et dans la variable k la base dans laquelle il est écrit.

Nous allons devoir ensuite récupérer chacun des chiffres de notre nombre pour le multiplier par la bonne puissance de la base de départ.

Quand nous voulons le faire à la calculatrice, pour le nombre $(1101)_2$, nous rentrons le 1 de gauche, nous le multiplions par 2^3, puis nous appuyons sur +, nous saisissons le second 1 que nous multiplions par 2^2 et ainsi de suite.
Cette étape de compréhension est essentielle car elle va nous guider pour notre script.

Pour récupérer les différents chiffres, le fait que notre nombre soit une chaîne de caractère sera un plus… Et c'est le cas, car la saisie d'un `input` est, par défaut, une chaîne de caractères.
Pour récupérer le premier chiffre (celui de gauche), il faudra faire `N[0]`, pour le second `N[1]`, …

Nous allons ainsi, balayer notre chiffre, faire la récupération puis faire notre somme.
Pour ce faire, une boucle for sera parfaite.
Nous avons besoin de savoir combien de boucles nous allons devoir faire…

Questions :[1]

- Combien de boucles devront nous faire pour $(1101)_2$?
- Comment récupérer cette information après nos 2 `input`, car bien entendu cela dépendra du nombre ?

Nous savons donc maintenant combien de fois nous allons boucler.
Nous allons poser que la conversion sera stockée dans une variable M.

Remarque :
Dans un `for i in range(0, 10):`, `i` prendra les valeurs 0, 1, 2, …, 9, donc 10 valeurs…

Notre code devient :

```
N = input("Quel est le nombre à convertir? ")
k = input("Quelle est la base de départ? ")
l = len(N)   #Nbre de chiffres de notre nombre

for i in range(0, l):
    M = M + N[i]

print(M)
```

J'ai rajouté un print(M) à la fin afin d'avoir un retour du calcul.
Pour l'instant, notre programme effectue la somme des chiffres du nombre.
Donc pour 1101, il devra retourner 3.

Débogage :
C'est une phase importante car elle apparait souvent, et nous allons voir ici les problèmes de notre code.
Je ne parlerai bien sûr ici que des bugs en partant du principe que vous n'avez pas fait d'oubli d'indentation ou de : par exemple .😉

À l'exécution du code vous devez avoir ce message d'erreur :

```
Traceback (most recent call last):
  File "<string>", line 6, in <module>
NameError: name 'M' is not defined
```

Cela veut dire qu'il ne connait pas `M` à la ligne 6.
Normal, pour la première boucle le programme doit faire :
`M = M + N[i]`
Or, nous n'avons pas initialiser la variable `M`.
Nous allons donc l'initialiser avant la boucle avec un `M = 0`.

---

[1] Réponses :
- 4 car il y a 4 chiffres.
- `len(N)` renvoie la longueur de la longueur de la chaîne de caractères, donc le nbre de chiffres.

Le code devient :

```
N = input("Quel est le nombre à convertir? ")
k = input("Quelle est la base de départ? ")
l = len(N)   #Nbre de chiffres de notre nombre

M = 0   #Initialisation de M
for i in range(0, l):
    M = M + N[i]

print(M)
```

Testez ce nouveau code…
Et oui, encore un problème… Mais ça fait partie du travail de programmeur. 😅

Vous devez avoir ce retour :

```
Traceback (most recent call last):
  File "<string>", line 7, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Ligne 7, il nous dit que nous demandons une addition entre un entier (`int`) et une chaîne de caractères (`str`).
Or, Python peut ajouter 2 entiers, ou 2 chaînes de caractères mais pas un entier et une chaîne de caractères. Et nous voulons que ça soit des entiers car nous faisons une addition mathématique et non une concaténation.
Regardons la ligne 7 : `M = M + N[i].`
M est bien un entier car nous l'avons initialisé comme ça. Par contre `N[i]`, est pris d'une chaîne de caractère et c'est donc un `str`.
Nous allons donc convertir `N[i]` en entier avec un `int(N[i])`.

Le code devient :

```
N = input("Quel est le nombre à convertir? ")
k = input("Quelle est la base de départ? ")
l = len(N)   #Nbre de chiffres de notre nombre

M = 0   #Initialisation de M
for i in range(0, l):
    M = M + int(N[i])

print(M)
```

Testez ce nouveau code…
Victoire!!! 🥳

Nous avons donc désormais un code qui additionne les chiffres du nombre de départ…

$$(1101)_2 = 1 + 1 + 0 + 1 = 3$$

Nous voulons qu'il fasse :

$$(1101)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

Donc il devra multiplier la bonne puissance de 2 (ou de k dans notre programme) par le chiffre récupéré puis faire la somme.

Réfléchissons un peu :
Pour notre nombre de 4 chiffres (ou l dans le cas général), le premier chiffre récupéré (celui de gauche) devra être multiplié par 2^3, le second chiffre par 2^2, le troisième par 2^1 et le dernier par 2^0.

Donc la ligne `M = M + int(N[i])` va devenir quelque chose comme ça :

```
M = M + int(N[i])*k** ?
```

`k** ?` est notre puissance de k, mais avec la formule de la puissance à trouver…

Question :[2]
Quelle formule doit-on mettre pour obtenir la bonne puissance ?

Suite à la page suivante…

[2] `k**(l-i-1)`

Notre code devient :

```
N = input("Quel est le nombre à convertir? ")
k = input("Quelle est la base de départ? ")
l = len(N)   #Nbre de chiffres de notre nombre

M = 0   #Initialisation de M
for i in range(0, l):
    M = M + int(N[i]) * k**(l-i-1)

print(M)
```

Testons-le et déboguons-le (si besoin…)

Et mince… Message d'erreur :

```
Traceback (most recent call last):
  File "<string>", line 7, in <module>
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
```

Bon, il nous dit que nous voulons une puissance avec un élément qui n'est pas supporté…
Et oui… Notre `k`, est le retour d'un input, donc c'est… une chaîne de caractères par défaut et non un entier et le reste de notre opération est avec des entiers, donc, problème.

Nous savons comment faire : un petit coup d'`int()` au niveau de l'input du `k`, et ça sera bon.

Le code devient :

```
N = input("Quel est le nombre à convertir? ")
k = int(input("Quelle est la base de départ? "))
l = len(N)   #Nbre de chiffres de notre nombre

M = 0   #Initialisation de M
for i in range(0, l):
    M = M + int(N[i]) * k**(l-i-1)

print(M)
```

Testez le code…

Il fonctionne !!!

Bon maintenant, testons un nombre en base 16 comme A1…

```
Traceback (most recent call last):
  File "<string>", line 8, in <module>
ValueError: invalid literal for int() with base 10: 'A'
```

Le problème des lettres

Notre programme ne sait pas gérer les lettres en tant que chiffres.
Nous allons nous pencher dessus…
Il faut donc que quand notre programme prend un 'A', il le transforme en 10, un 'B' un 11, et ainsi de suite.

On pourrait dans notre programme inséré un `if` de ce type :

```
if N[i] == 'A':
    a = 10
elif N[i] == 'B':
    a = 1
...
else:
    a = N[i]
```

Avec a qui serait une variable qui stockerait la valeur du chiffre.

On va être honnête ce n'est pas optimisé du tout…

Nous allons plutôt utiliser la fonction `ord()`.

Testez les lignes suivantes dans la console :

```
ord('A')
ord('B')
```

Vous avez dû remarquer que `ord('A')` retourne la valeur de 65, `ord('B')` la valeur 66.
En fait, `ord()` renvoie le code UNICODE du caractère.

Nous, nous ne voulons pas une valeur de 65 pour A mais de 10, mais une soustraction par … et ça sera bon.

Par contre, il faut que notre programme détecte s'il s'agit d'une lettre et dans ce cas utiliser le `ord()`, ou d'un chiffre dans ce cas, on sait faire…

Pour faire cela, nous allons utiliser : `.isalpha()`.

Testez les lignes suivantes :

```
'A'.isalpha()
'10'.isalpha()
```

Vous avez un retour `True` pour `'A'` et `False` pour `'10'`.
On a un détecteur de lettres !!!

Je vais vous proposer un code à compléter avec ces nouvelles données…

Code à compléter :[3]

```
N = input("Quel est le nombre à convertir? ")
k = int(input("Quelle est la base de départ? "))
l = len(N)
M = 0   #Initialisation de M

for i in range(0, l):
        if N[i].isalpha() == True:
            …
            …
        else:
            M = M + int(N[i]) * k**(l-i-1)

print(M)
```

Votre code est désormais fonctionnel…
Nous allons le transformer en fonction, ça sera court.

Transformation en fonction

Nous aurions pu le faire dès le début, mais il était plus simple dans un premier temps de le faire comme ça.

Pour créer une fonction en Python c'est assez simple :

```
def fonction(argument1, argument2, ...):
    instructions
    return ...
```

Pourquoi un `return` et non un `print`…
Parce qu'avec un return nous pourrons utiliser le résultat de notre programme alors qu'avec un print on ne fait que l'afficher sans utilisation possible…

Pour nos arguments, il s'agira du nombre à convertir et de la base de départ.
nous retirerons les `input()` car leur fonction est pris en charge par les arguments.

Code à compléter :

```
def conversion_vers_10(N, k):
    …
    return M
```

Réponse page d'après…

[3] `n = ord(N[i]) - 55`
`M = M + n * k**(l-i-1)`

```
def conversion_vers_10(N, k):
    N = str(N)   #permet que l'utilisateur rentre son nombre sans guillement
    l = len(N)
    M = 0   #Initialisation de M

    for i in range(0, l):
        if N[i].isalpha() == True:
            n = ord(N[i]) - 55
            M = M + n * k**(l-i-1)
        else:
            M = M + int(N[i]) * k**(l-i-1)

    return M
```

Dans la console, rentrez :

```
conversion_vers_10(A1,16)
```

Votre fonction … fonctionne.

Félicitations pour le suivi, et dans le prochain TP, nous verrons comment faire une conversion en partant de la base 10 !!!