Comment intégrer des chaines binaires dans le code source en base64

par Jean-Paul Vidal

Date de publication : 27/10/2010

Dernière mise à jour : 05/11/2010

On appellera ici chaine binaire une chaine de caractères dans laquelle chaque caractère peut avoir une valeur de 0 à 255 (bornes comprises).

Intégrer une chaine binaire dans le code source n'est pas facile, parce que la chaine binaire peut comporter a priori des octets de 0 à 255, mais les caractères intégrables dans le code source, donc affichables, devraient exclure au minimum les caractères de contrôles inférieurs à 32, certains caractères comme les guillemets ou les backslashs et se limiter à des octets inférieurs 128 (c'est à dire rester à 7 bits) pour échapper aux problèmes d'encodages.

On va utiliser ici le module base64.

I - Transformer une chaine binaire en chaine encodée base64

```
import base64

ch64 = base64.b64encode(chbin)
```

Par exemple :

```
chbin = "bonjour à tous!"
ch64 = base64.b64encode(chbin)
print ch64
'Ym9uam91ciDgIHRvdXMh'
```

NB : l'Unicode ne semble pas être supporté avec Python 2.7.

À noter que, en transformant une chaine binaire en chaine encodée en base64, on augmente le nombre d'octets d'environ 33 %.

II - Transformer une chaine encodée base64 en chaine binaire

```
import base64
 chbin = base64.b64decode(ch64)
```

Par exemple :

```
ch64 = 'Ym9uam91ciDgIHRvdXMh'
chbin = base64.b64decode(ch64)
print chbin
"bonjour à tous!"
```

III - Intégration dans le code source

Comme la chaine encodée est monoligne, on va la répartir en plusieurs lignes comme suit :

```
import base64

def encodeb64chaine(chbin, nvar='ch64', nbcar=69, fdl='\n'):
    ch64 = base64.b64encode(chbin)
    ch = nvar + ' = "\\' + fdl
    lg = len(ch64)
    i2 = 0
    while True:
        i1 = i2
        i2 = min(i1+nbcar, lg)
        if i2>i1:
            ch += ch64[i1:i2] + '\\' +  fdl
        else:
            break
    ch += '"'
    return ch
```

Vous pouvez, bien entendu, adapter à l'appel :

- le nom de variable nvar ('ch64' par défaut) ;
- le nombre de caractères dans la ligne nbcar (69 par défaut, pour obtenir 70 avec le backslash final) ;
- la fin de ligne fdl en fonction de votre contexte: Windows, Linux, Mac OSX, et de votre éditeur de texte ('\n' par défaut, comme pour Linux).

Avec cette fonction, une chaine binaire de 500 octets, par exemple, donne un résultat comme ça, qu'on peut intégrer dans un code source par copier-coller :

```
ch64 = "\
LoygvW2rPYGStOXglIo4TePBsmhTxy3Yg3xg+x/ehsYe4n0HiHEHu00CW2uDyzEXGWrfw7\
twiUJcJwHvwrDQKiM97csxnlNmVLtu1WGCQohacme2qSAAF7sc19FA2qaVvaNRE4RxQD0/\
D8keEEjBSLP9Zw0dTfxyfyeqthSXcRo1XVncBIzGsEsFULI93vN238PBhDHslKvYtWaGIs\
/UTFbQd2KGM7MVgd6HQVzWRdVcYFaLTnHdJ+sM/A/MSLW1anOr4QoiIeKXIEaTEdntwN79\
ZxkSt+WCHWfv0n97bD/2rPixSAIuMctyBCqgLXnnbPuXY3ykNW2CAOFhAHTYYr/g0gNISk\
9jSaCf0d6+HogZVdFTmeHW/tM7ngyi+rpZ4kITQyXskm2DJbmLtorQKepMu6/c60pX8nia\
Fw/NEZaE8kGLqIe1+D3H7Txan+h6q0IBCTYWm+XZDPC/RwCKK3knco9Y9KuBOEsOaL+QRm\
2jELle47LwEpYDCRdfy5+IownYsIVDhl/wCpGQ7PM6zwMFF8AwT76Sl/Y5ZI47ljbiZCrO\
C+MZ1WUdSh46sg+dkgRX97EznxbNIYHojps5n/XYubIpHiVNhCrsedF7r9/VlIlee1HsqG\
0CKesIZO7oqol38l/Z3tG68WFxJpXg+1DpbK8=\
"
```

Par la suite de ce code source, on peut retrouver la chaine binaire initiale avec :

```
chbin = base64.b64decode(ch64)
```

Et si c'est une image, on peut maintenant l'afficher !

On peut aussi transformer la chaine encodée en liste de chaine :

```
def encodeb64liste(chbin, nvar='lch64', nbcar=67, fdl='\n'):
    ch64 = base64.b64encode(chbin)
    ch = nvar + ' = [' + fdl
    lg = len(ch64)
    i2 = 0
    while True:
        i1 = i2
        i2 = min(i1+nbcar, lg)
        if i2>i1:
            ch += '"' + ch64[i1:i2] + '",' +  fdl
        else:
            break
    ch += ']'
    return ch
```

Vous pouvez, bien entendu, adapter :

- le nom de variable nvar ('lch64' par défaut) ;
- le nombre de caractères dans la ligne nbcar (67 par défaut, pour obtenir 70 avec les 2 guillemets et la virgule finale) ;
- la fin de ligne fdl en fonction de votre contexte: Windows, Linux, Mac OSX, et de votre éditeur de texte ('\n' par défaut, comme pour Linux).

Ce qui donnera la chaine suivante pour l'intégration dans le code source :

```
lch64 = [
"pbgi5FekGy2QlFTjJNalkn9tEuq/bBb82AjoQ6NTWG2hakr5bS6LCpMxSGoADtNPZeSW9i",
"Zk/HSavcaE8dF6u1HxFBsVB54Yr2dU9W2eNm9QuO9KMehuIBZeNOcKXmVH4vx9QtmBOSKm",
"67ZvDp6GW8jsXNyV5kLM30Bs/gp9Xr5a9g2js8lOKsXa8JGc3Am2xqF1IfWZeWa84kq+K6",
"0GXa6RiKuH7w7ro463PIXeMULYsmPdjaTX2gCAU05Sfr0x+mi9c2i3hUTzZo7AmR079Wak",
"3Tm7p2Ixj8Bs2Qx1k6BxRw2slX7aBPYJ5JXahCfD2+nbs71ZxFFbx6jeFC8HHXLt673yW9",
"FB8R27V+rYjXEnL8pWNXnWblLpSKfTBKPlRvv204koYGviVRYeRHUxqie7pEo9Cw2PenL3",
"JRYYpaqNFR8PDLaxeRZXRhTPmqauwieMOmcUUSgpiVQk8UvdYIz2g5ZE2c32cAO9aEfFOn",
"zHLBLXSZVH/hzAL9ComutbA/RCXOsvGpidDlKo/Ijt9GCrJ95/bawfz7MSrEEFKiV+vUGj",
"Z8JC38oVcFV8BfeZmDdibTKO8hcAFrytqhs7GV70Lbeaiwj IZ2mtrPlR53QMaEBf4OBYv3",
"LXROlcC8hbdyZ8IID2KGwa4eQ/3QtSGKtz9fE=",
]
```

Dans ce code source, on pourra retrouver la chaine binaire initiale comme suit :

```
chbin = base64.b64decode(''.join(lch64))
```