Modifier un message entrant en Python

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 24 novembre 2023

https://www.bortzmeyer.org/repair-email.html

Un peu de programmation aujourd'hui. Supposons qu'on reçoive des messages qui ont été modifiés en cours de route et qu'on veut remettre dans leur état initial. Comment faire cela en Python ?

Comme exemple, on va supposer que les messages contenant le mot « chiffrer » ont été modifiés pour mettre « crypter » et qu'on veut remettre le terme correct <https://www.bortzmeyer.org/cryptage-n-existe-pas.html>. On va écrire un programme qui reçoit le message sur son entrée standard et met la version corrigée sur la sortie standard. D'abord, la **mauvaise** méthode, qui ne tient pas compte de la complexité du courrier électronique :

```
import re
import sys

botched_line = re.compile("^(.*?)crypter(.*?)$")

for line in sys.stdin:
    match = botched_line.search(line[:-1])
    if match:
        # re.sub() serait peut-être meilleur ?
        newcontent = match.group(1) + "chiffrer" + match.group(2) + "\n"
    else:
        newcontent = line
    print(newcontent, end="")
```

On lit l'entrée standard, on se sert d'une expression rationnelle (avec le module re <https://docs.python.org/3/library/re.html>) pour trouver les lignes pertinentes, et on les modifie (au passage, le point d'interrogation à la fin des groupes entre parenthèses est pour rendre l'expression non gourmande). Cette méthode n'est **pas** bonne car elle oublie :

— Qu'un message peut être composé de plusieurs parties,
— et que même les parties faites de texte peuvent être encodées, par exemple en "*quoted-printable*".

Il faut donc faire mieux.

Il va falloir passer au module email <https://docs.python.org/3/library/email.html>. Il fournit tout ce qu'il faut pour analyser proprement un message, même complexe :

```
import sys
import re
import email
import email.message
import email.policy
import email.contentmanager

botched_line = re.compile("^(.*?)crypter(.*?)$")

msg = email.message_from_file(sys.stdin, _class=email.message.EmailMessage,
                              policy=email.policy.default)
for part in msg.walk():
    if part.get_content_type() == "text/plain":
        newcontent = ""
        for line in part.get_content().splitlines():
            match = botched_line.search(line)
            if match:
                # re.sub() serait peut-être meilleur ?
                newcontent += match.group(1) + "chiffrer" + match.group(2) + "\n"
            else:
                newcontent += line + "\n"
        email.contentmanager.raw_data_manager.set_content(part, newcontent)
print(msg.as_string(unixfrom=True))
```

Ce code mérite quelques explications :

— `email.message_from_file` lit un fichier (ici, l'entrée standard) et rend un objet Python de type message. Attention, par défaut, c'est un ancien type, et les opérations suivantes donneront des messages d'erreur incompréhensibles (comme « *"AttributeError : 'Compat32' object has no attribute 'content_manager'"* » ou « *"AttributeError : 'Message' object has no attribute 'get_content'. Did you mean : 'get_content_type' ?"* »). Les paramètres `_class` et `policy` sont là pour produire des messages suivant les types Python modernes.
— `walk()` va parcourir les différentes parties MIME du message, récursivement.
— `get_content_type()` renvoie le type MIME de la partie et nous ne nous intéressons qu'aux textes bruts, les autres parties sont laissées telles quelles.
— `get_content()` donne accès aux données (des lignes de texte) que `splitlines()` va découper.
— Si l'expression rationnelle correspond au motif donné, on ajoute la version modifiée, sinon on ne touche à rien.
— `set_content()` remplace l'ancien contenu par le nouveau.
— Et enfin, `as_string` transforme l'objet Python en texte. Notre message a été transformé.

Ce code peut s'utiliser, par exemple, depuis procmail, avec cette configuration :

```
:0fw
| $HOME/bin/repair-email.py
```

Évidemment, il peut être prudent de sauvegarder le message avant cette manipulation, au cas où. En procmail :

```
:0c:
Mail/unmodified
```