W3-S4-C2-references-circulaires

December 15, 2014

1 Listes infinies & références circulaires

1.1 Complément - niveau intermédiaire

Nous allons maintenant construire un objet un peu abscons. Cet exemple précis n'a aucune utilité pratique, mais permet de bien comprendre la logique du langage.

Construisons une liste à un seul élément, peu importe quoi:

```
In []: infini_1 = [None]
```

À présent nous allons remplacer le premier et seul élément de la liste par... la liste elle-même

```
In []: infini_1[0] = infini_1
       print infini_1
```

Nous vous conseillons d'évaluer cette séquence sous http://pythontutor.com pour bien visualiser ce qui se passe.

Pour essayer de décrire l'objet liste ainsi obtenu, on pourrait dire qu'il s'agit d'une liste de taille 1 et de profondeur infinie, une sorte de fil infini en quelque sorte.

Naturellement, l'objet obtenu est difficile à imprimer de manière convaincante. Pour faire en sorte que cet objet soit tout de même imprimable, et éviter une boucle infinie, python utilise l'éllipse `...` pour indiquer ce qu'on appelle une référence circulaire. Si on n'y prenait pas garde en effet, il faudrait écrire `[[[[ etc.. ]]]]` avec une infinité de crochets.

Toutes les fonctions de python ne sont pas aussi intelligentes. Bien qu'on puisse comparer cette liste avec elle-même:

```
In []: infini_1 == infini_1
```

il n'en est pas de même si on la compare avec un objet analogue mais pas identique:

```
In []: infini_2 = [0]
       infini_2[0] = infini_2
       print infini_2
       infini_1 == infini_2
```

1.1.1 Généralisation aux références circulaires

On obtient un phénomène équivalent dès lors qu'un élément contenu dans un objet fait référence à l'objet lui-même. Voici par exemple comment on peut construire un dictionnaire qui contient une référence circulaire:

```
In []: collection_de_points = [
           {'x': 10,'y': 20},
           {'x': 30,'y': 50},
       # imaginez plein de points
       ]
```

```
# on rajoute dans chaque dictionnaire une clé 'points'
# qui référence la collection complète
for point in collection_de_points:
    point['points'] = collection_de_points

# la structure possède maintenant des références circulaires
print collection_de_points
```

On voit à nouveau réapparaître les élipses, qui indiquent que pour chaque point, le nouveau champ 'points' est un objet qui a déjà été imprimé.

Cette technique est cette fois très utile et très utilisée dans la pratique, dès lors qu'on a besoin de naviguer de manière arbitraire dans une structure de données compliquée. Dans cet exemple, pas très réaliste naturellement, on pourait à présent accéder depuis un point à tous les autres points de la collection dont il fait partie.

À nouveau il peut être intéressant de voir le comportement de cet exemple avec http://pythontutor.com pour bien comprendre ce qui se passe, si cela ne vous semble pas clair à première vue.

Voici d'ailleurs comment pythontutor présenterait la situation à la fin de ce fragment de code