

Estimer le plus petit (grand) flottant


Exercice - niveau basique

Le plus petit flottant

En corollaire de la discussion sur la précision des flottants, il faut savoir que le système de codage en mémoire impose aussi une limite. Les réels très petits, ou très grands, ne peuvent plus être représentés de cette manière.

C'est notamment très gênant si vous implémentez un logiciel probabiliste, comme des graphes de markov, où les probabilités d'occurrence de séquences très longues tendent très rapidement vers des valeurs extrêmement petites.

Le but de cet exercice est d'estimer la valeur du plus petit flottant qui peut être représenté comme un flottant. Pour vous aider, voici deux valeurs:

 `10**-320`

`10**-330`

Comme on le voit, 10^{-320} est correctement imprimé, alors que 10^{-330} est, de manière erronée, rapporté comme étant nul.

À ce stade du cours, pour estimer le plus petit flottant, procédez simplement par approximations successives.

Note Sans utiliser de boucle, la précision que vous pourrez obtenir n'est que fonction de votre patience; ne dépassez pas 4 à 5 itérations successives :)

Il est par contre pertinent d'utiliser une approche rationnelle pour déterminer l'itération suivante (par opposition à une approche "au petit bonheur"). Pour ceux qui ne connaissent pas, nous vous recommandons de vous documenter sur l'algorithme de **dichotomie**.

 `10**-325`

Voici quelques cellules de code vides; vous pouvez créer autant de cellules que nécessaire, le plus simple étant de taper `Alt+Enter`, ou d'utiliser le menu *"Insert -> Insert Cell Below"*

 `<>`

`<>`

```
<>
```

```
<>
```

Le plus grand flottant

La même limitation s'applique sur les grands nombres. Toutefois cela est un peu moins évident, car comme toujours il faut faire attention aux types

```
10**450
```

Ce qui se passe très bien car on obtient un `long`, alors que si on précise bien qu'on veut un flottant on obtient une erreur

```
10**450.0
```

On peut d'ailleurs remarquer que le comportement ici n'est pas extrêmement cohérent, car avec les petits nombres python nous a silencieusement transformé 10^{-330} en `0`, alors que pour les grands nombres il lève une exception (nous verrons les exceptions plus tard, mais vous pouvez dès maintenant remarquer que le comportement est différent dans les deux cas).

Quoi qu'il en soit, la limite pour les grands nombres se situe entre les deux valeurs 10^{300} et 10^{310} . On vous demande à nouveau d'estimer comme ci-dessus une valeur approchée du plus grand nombre qu'il est possible de représenter comme un flottant.

```
<>
```

```
<>
```

```
<>
```

```
<>
```

Complément - niveau avancé

En fait, on peut accéder à ces valeurs minimales et maximales pour les flottants comme ceci

```
import sys
print sys.float_info
```

Et notamment, comme expliqué ici (https://docs.python.org/2/library/sys.html#sys.float_info)

```
> print "Flottant minimum", sys.float_info.min  
| print "Flottant maximum", sys.float_info.max
```

Sauf que: vous devez avoir trouvé un maximum voisin de cette valeur, mais le minimum observé expérimentalement ne correspond pas bien à cette valeur.

Pour ceux que cela intéresse, l'explication à cette apparente contradiction réside dans l'utilisation de nombres dénormaux (<http://en.wikipedia.org/wiki/Denormal%5Fnumber>) .