http://irem.univ-reunion.fr/spip.php?article529

Boucles et itérateurs en Python

- Algorithmique et programmation
- Programmation Python en Seconde
-

Date de mise en ligne : jeudi 20 octobre 2011

La chapitre sur les intervalles a servi de prétexte à une introduction des boucles à nombre prédéfini d'exécutions, celles-ci pouvant être stockées dans des variables ! En fait, un *itérateur* est un algorithme, qui permet d'économiser de la mémoire en remplaçant les données par un procédé permettant de les obtenir.

boucles statiques

On peut donc affecter une variable avec une boucle :

boucle=range(1,9)
for compteur in boucle:
 print('La valeur actuelle du compteur est ',compteur) <a href='http://irem.univ-reunion.fr/local/cache-code/ccabf7c9ef663de6faad36b658e06c65.txt' style='font-family: verdana, arial, sans; font-weight: bold; font-style: normal;'>Télécharger

Ce qui permet de manipuler algorithmiquement des intervalles d'entiers avec des choses comme ceci :

print(list(range(8))
print(3 in range(5)) <a href='http://irem.univ-reunion.fr/local/cache-code/59e927db2c9977fb2392d0c15e82f4c9.txt' style='font-family: verdana, arial, sans; font-weight: bold; font-style: normal;'>Télécharger

Le TP a servi essentiellement à rappeler les notions suivantes :

1. Tableau de valeurs d'une fonction (en bas de la première page)
2. Diagramme en bâtons des effectifs
3. Tableur (pour la comparaison)

Le sujet est ici :
 <a href="http://irem.univ-reunion.fr/IMG/pdf/TP3.pdf" title='PDF - 102.6 ko' type="application/pdf">[PDF - 102.6 ko] **intervalles et itérateurs : sujet**

Mais peu d'élèves ont pu faire le TP à cause d'une panne de réseau, et aucun élève ne semble savoir se servir d'un tableur pour simuler le hasard ; en particulier, les fonctions *ALEA.ENTRE.BORNES* et *NB.SI* leur étaient totalement inconnues.

Module Â« tortue Â»

Pour illustrer les boucles, l'exemple archiclassique des polygones réguliers a été choisi, parce qu'il est graphique et

permet d'introduire le module Â« tortue Â» de Python. Le TP a servi aussi de révisions sur les coordonnées (Python sait lire celles de la tortue) et la géométrie du triangle (la réciproque de Pythagore et la trigonométrie étaient nécessaires pour la dernière question). En voici le sujet :
<a href="http://irem.univ-reunion.fr/IMG/pdf/TP4.pdf" title='PDF - 105.3 ko' type="application/pdf">[PDF - 105.3 ko] **découverte du module tortue : sujet**

Malgré la simplicité de l'énoncé, les élèves font preuve d'une extraordinaire imagination pour faire des erreurs totalement imprévues :
<a href="http://irem.univ-reunion.fr/IMG/pdf/TP4corr.pdf" title='PDF - 872.7 ko' type="application/pdf">[PDF - 872.7 ko] **corrigé du TP tortue**

Fonctions

En fait, le module Â« tortue Â» permet très facilement de représenter des fonctions, au point que la seule vraie difficulté est le tracé des axes :
<a href="http://irem.univ-reunion.fr/IMG/pdf/TP5.pdf" title='PDF - 104.6 ko' type="application/pdf">[PDF - 104.6 ko] **représentation graphique d'une fonction avec turtle : sujet**

Certains élèves manquent cruellement de sens de l'orientation, ce qui donne des résultats parfois comiques et pas toujours proches de l'effet escompté...

Écriture de la fonction V

9 élèves ont écrit Â« 5 Â» comme fonction (sans doute parce qu'ils ont vu le Â« 5 Â» de l'axe des abscisses sur le graphique de l'énoncé et qu'ils ne savaient pas quoi mettre d'autre). Cette erreur est exploitable sous la forme d'un rappel : La représentation graphique d'une fonction constante est une droite horizontale.

Voici quelques tentatives :

```
from math import *
def V(x):
 y=forward(4/3000)
 y=y*pi
 y=y*x**3
 return y<div class='code_download'   style='text-align: right;'> <a
href='http://irem.univ-reunion.fr/local/cache-code/6ed7716317c228e7c27e54d5980c0a1f.txt'   style='font-family:
verdana, arial, sans; font-weight: bold; font-style: normal;'>Télécharger
```

(l'idée de décomposer la fonction en plusieurs étapes simples est plutôt bonne algorithmiquement parlant, même s'il était parfaitement possible de n'écrire la fonction qu'en une ligne, ce qu'ont d'ailleurs fait les meilleurs élèves [1])

Variante qui montre mieux les hésitations :

```
from math import *
```

```
def V(x):
 y=froward(4/3000)
 y=pi+y
 return y<div class='code_download'   style='text-align: right;'> <a
href='http://irem.univ-reunion.fr/local/cache-code/38f6dd76b5082e791c950e54eb156b40.txt'   style='font-family:
verdana, arial, sans; font-weight: bold; font-style: normal;'>Télécharger
```

qui pourrait bien être copié sur la voisine :

```
from math import *
def V(x):
 y=forward(4/3000)
 y=pi+y
 y=y*x*3
 return y<div class='code_download'   style='text-align: right;'> <a
href='http://irem.univ-reunion.fr/local/cache-code/d82b225dccde548d7a11f8a7542ecad0.txt'   style='font-family:
verdana, arial, sans; font-weight: bold; font-style: normal;'>Télécharger
```

Graduations

En fait, il y a une erreur dans l'énoncé, un *left* ayant malencontreusement pris la place d'un *right*. Ce qui a constitué un test pour la découverte d'autres méthodes, voire des angles négatifs. Finalement

- 24 élèves ont barré le *left* fautif et mis un *right(90)*
- 5 élèves ont trouvé que *left(270)* avait le même effet qu'une rotation vers la droite (espérons que ces 5 là seront à l'aise en trigonométrie lorsqu'il faudra réinvestir cela)
- 4 élèves ont maintenu un *left(90)* qui dessine un petit carré au lieu de l'axe gradué.

Statistique sur les coups de tampon

Python donne à chaque coup de tampon (*stamp()*) un numéro, qu'IDLE affiche parfois. Ce numéro change au cours du TP, et n'est pas le même chez tous les élèves. Voici le résultat d'un comptage partiel de ces numéros :

numéros	*effectifs*
6	11
7	1
8	4
9	2
10	7
12	1
14	3
17	1

18	1
20	1
21	1
34	1
41	1

Un modèle par loi de Poisson paraît assez bon, sauf que les valeurs paires sont plus fréquentes et que les valeurs faibles (inférieures à 6) sont absentes.

---

[1] Ça fera plaisir aux AAA, les fameux *Ayatollahs Anonymes d'AlgoBox* qui mettent tant d'énergie à déverser leur fiel anonyme en commentaire, sans jamais étayer leur propos d'un seul fait vérifiable...