Boucles et tests

Les 9, 10 et 16 octobre 2013
http://www.mp933.fr/ - stephane@gonnord.org

Buts du TP

- Continuer à dompter l'environnement.
- Écrire encore et encore des boucles simples et des tests.

Dans la section « Pour ceux qui s'ennuient », ceux connaissant déjà Python gnagnagna...

Exercice 1 *Créer (au bon endroit) un dossier associé à ce TP. Dans ce dossier, placer une copie du fichier* `cadeau-tp-boucles.py` *fourni dans le dossier partagé de la classe (ou sur le web).*

Lancer Spyder, sauvegarder immédiatement le fichier du jour au bon endroit. Écrire une commande absurde, de type `print(5*3)` *dans l'éditeur ; sauvegarder et exécuter.*

1 Cinq boucles

Exercice 2 *Calculer* $\sum_{k=831}^{944} k^{10}$*, si possible sans regarder le corrigé du tp précédent... mais en le consultant tout de même si la difficulté vous semble insurmontable !*

Le résultat sera copié collé de l'interpréteur vers le fichier `.py`, et commenté. À ce moment du TP, votre feuille de travail dans l'éditeur doit contenir quelque chose comme :

```
# -*- coding: utf-8 -*-
"""
Created on ...

@author: ...
"""

#
# Exo 1 : Fait !
#

#
# Exo 2 :
#

somme = 0
...
...

# >>> somme
# 367241911503651005721610202208 25L

#
# Exo 3 :
#
```

Je vous conseille vivement de ne pas attendre six mois pour commencer à travailler correctement : prenez dès maintenant de bonnes habitudes en TP.

On continue par des boucles basiques pour calculer a^b et $n!$

Exercice 3 *Calculer* 3^{841} *en appliquant l'algorithme basique suivant :*

$res \leftarrow 1$
pour i *de* 1 *à* 841 **faire**
  $res \leftarrow res \times 3$
Résultat *: res*

Comparer avec le résultat de `3**841`

Exercice 4 *Calculer* 100! *en appliquant l'algorithme suivant :*

$res \leftarrow 1$
pour i *de* 2 *à* 100 **faire**
  $res \leftarrow res \times i$
Résultat *: res*

Comparer avec le résultat fourni par la fonction `factorial` *de la bibliothèque* `math` *:*

```
>>> import math
>>> math.factorial(...)
```

ou encore :

```
>>> from math import factorial
>>> factorial(...)
```

ou enfin :

```
>>> from math import *
>>> factorial(...)
```

Dans l'exercice suivant, on va calculer la somme des chiffres d'un gros entier. Si $\varphi(n)$ désigne la somme des chiffres de n (dans son écriture décimale...), on a par exemple $\varphi(841) = 13$. Pour calculer $\varphi(1234567654398)$, on peut prendre une variable `somme` dans laquelle on va sommer les décimales, en les faisant parallèlement disparaître du nombre initial. Par exemple, $n = 1234567654398$ et $somme = 0$ au départ. Après une étape, $n = 123456765439$ et $somme = 8$; après deux étapes, $n = 12345676543$ et $somme = 17$... et après 13 étapes, $n = 0$ et $s = 63$: la somme vaut 63. L'idée est, à chaque étape, de faire passer la dernière décimale de n dans la somme, *puis* de la faire disparaître de n.

Exercice 5 Project Euler, problème numéro 20

Calculer la somme des décimales de 100! *de la façon suivante :*

$somme \leftarrow 0$
$n \leftarrow 100!$
tant que $n > 0$ **faire**
  $somme \leftarrow somme + (n\%10)$
  $n \leftarrow n//10$
Résultat *: somme*

Ceux qui sont à la bourre zappent l'exercice suivant.

Exercice 6 Suite de Fibonacci : premier épisode.

La suite de Fibonacci est définie par $f_0 = 0$, $f_1 = 1$ *et pour tout* $n \in \mathbb{N}$, $f_{n+2} = f_n + f_{n+1}$.

- *Calculer* f_n *à la main, pour* $n \leqslant 10$.
- *Écrire un algorithme permettant de calculer* f_{100}.
- *Programmer cet algorithme en Python.*
- *Que vaut finalement* f_{100} *? Et* f_{1000} *?*

Pour ceux qui sèchent, un algorithme est proposé en dernière partie de TP.

2 Autour des nombres premiers

Exercice 7 *Importer la fonction* `est_premier` *du fichier* `cadeau_tp_boucles.py` *et exécuter :*

```
for n in range(20):
    if est_premier(n):
        print(n)
```

(on aura écrit cette boucle dans l'éditeur[1] *avant de l'exécuter).*

EXERCICE 8 Un peu de complexité
Lire le code de la fonction `est_premier` *: combien réalise-t-elle d'« opérations élémentaires » lorsqu'elle est exécutée avec en entrée un entier pair ? Et un entier premier ?*

EXERCICE 9 Complexité à la louche
Sachant que «à la louche, la proportion d'entiers $\leqslant N$ *qui sont premiers est de l'ordre de* $\frac{1}{\ln N}$ *», évaluer le nombre d'opérations élémentaires nécessaires pour tester la primalité des entiers* $\leqslant N$*.*

Pour $N = 10^6$*, le calcul va-t-il prendre un temps de l'ordre du pouillème de seconde, de la minute, ou de la journée ?*

EXERCICE 10 *Combien il y a-t-il d'entiers plus petits que* 100 *qui sont premiers ? Même chose pour les entiers majorés par* 10^4 *puis* 10^6*.*

cpt $\leftarrow 0$
pour *n allant de* 1 *à ...* **faire**
  si *n est premier* **alors**
    cpt $\leftarrow$ *cpt* $+ 1$
Résultat *: cpt*

EXERCICE 11 *Combien existe-t-il de* $n \leqslant 10^6$ *tels que* n *et* $n+2$ *sont premiers ?*

EXERCICE 12 *Trouver le plus petit entier* n *supérieur à* 10^{10} *tel que* n *et* $n+2$ *sont premiers.*

EXERCICE 13 Seulement pour ceux qui ont de l'avance !
Compter précisément le nombre de divisions euclidiennes effectuées pour tester la primalité des entiers majorés par 10^6*.*

3 Observons une suite d'entiers

On s'intéresse ici à la suite définie par son premier terme $u_0 = 42$ puis la relation de récurrence $u_{n+1} = 15091u_n \mod 64007$ pour tout $n \in \mathbb{N}$.

EXERCICE 14 *Que vaut* u_1 *? Et* u_{10} *? Et* u_{10^6} *?*

EXERCICE 15 *Compter le nombre de* $n \leqslant 10^7$ *vérifiant les conditions suivantes :*

1. u_n *est pair ;*
2. u_n *est premier ;*
3. $u_n \mod 3 = 1$ *;*
4. $u_n \mod 3 = 1$ *et* u_n *est premier ;*
5. u_n *est pair et* u_n *est premier ;*
6. n *est pair et* u_n *est premier.*

1. Dans le fichier de script `tp3.py`, ou quelque chose comme ça.

4 Autour de la multiplication et l'exponentiation modulaire

Exercice 16 *Sans calculatrice : quelle est la dernière décimale de* 17×923 *?*
Quelle est la dernière décimale de $123345678987654 \times 836548971236$ *?*

Bien... donc finalement : *pour connaître* $ab \mod 10$*, on fait le produit de* $a \mod 10$ *par* $b \mod 10$*, et on regarde ce produit modulo* 10.

On montrerait sans problème que ce résultat reste valable modulo n'importe quel entier. De même, pour calculer $a^b \mod c$, on peut faire b multiplications par a et réduire modulo c à chaque étape.

$res \leftarrow 1$
pour i *de* 1 *à* b **faire**
  $res \leftarrow res \times a \mod c$
Résultat : res

Exercice 17 *Expliquer l'intérêt de cette façon de procéder par rapport à la version «on calcule* a^b*, puis on réduit le résultat modulo* c*».*
Calculer ainsi $123456^{654321} \mod 1234567$*.*

Exercice 18 Project Euler : problem 48
The series, $1^1 + 2^2 + 3^3 + \cdots + 10^{10} = 10405071317$*. Find the last ten digits of the series,*

$$1^1 + 2^2 + 3^3 + \cdots + 1000^{1000}.$$

5 Pour ceux qui s'ennuient

Exercice 19 *Vérifier le théorème de* Wilson*, pour* $p \leqslant 10^4$ *:*

Un entier p est premier si et seulement si p divise $(p-1)! + 1$.

Exercice 20 *Pour* $n \in [\![3, 30]\!]$*, déterminer le nombre de* $k \in [\![2, n-1]\!]$ *tels que* $k^2 \mod n = -1$ *(ou encore :* $k^2 \equiv -1\ [n]$ *: on parle de racine de* -1 *modulo* n*).*
Quel est le nombre de $n \in [\![3, 1000]\!]$ *tels que l'équation* $k^2 \equiv -1\ [n]$ *possède (au moins) une solution ?*

Exercice 21 *Déterminer les nombres premiers* p *majorés par* 100 *tels qu'existe* $a, b \in \mathbb{N}$ *tels que* $a^2 + b^2 = p$*. Qu'observe-t-on (regarder modulo* 4*) ?*
Beaucoup plus difficile : déterminer les entiers n *majorés par* 50 *tels qu'existe* $a, b \in \mathbb{N}$ *tels que* $a^2 + b^2 = n$*. Qu'observe-t-on ?*

Exercice 22 Très difficile
On définit une suite « de type Fibonacci » en posant $f_0 = 1$*,* $f_1 = f_2 = \cdots = f_{99} = 0$*, et* $f_{n+100} = f_n + f_{n+1}$ *pour tout* $n \in \mathbb{N}$*.*

1. *Que vaut* f_{10^6} *modulo* 1515 *?*
2. *Quelles sont les quatre dernières décimales de* $f_{10^{15}}$ *?*

6 Besoin d'indications ?

– Exercice 6. On calcule les valeurs du couple (f_k, f_{k+1}) pour k allant de 0 à 99. L'idée est que si $(f_k, f_{k+1}) = (a, b)$, alors au rang suivant : $(f_{k+1}, f_{k+2}) = (b, a+b)$, ce qui donne un algorithme assez simple :

  $(a, b) \leftarrow (0, 1)$
  pour k *de* 1 *à* 99 **faire**
    # À l'entrée $(a, b) = (f_{k-1}, f_k)$
    $(a, b) \leftarrow (b, a+b)$ # Et à la sortie $(a, b) = (f_k, f_{k+1})$
  Résultat : b

  On trouvera $f_{100} = 354224848179261915075$ et $f_{1000} = 434665...849228875$.

– Exercice 14. On a $u_1 = 57750$, $u_{10} = 52866$ et $u_{10^6} = 14919$.
– Exercice 15. On calcule les termes de proche en proche, en mettant à jour 6 compteurs :

$(c_1, c_2, ..., c_6) \leftarrow (0, 0, ..., 0)$
$x \leftarrow 42$ # x va représenter le terme courant u_n
pour n *de* 0 *à* 10^6 **faire**
  # On traite ici u_n
  si $x \mod 2 = 0$ **alors**
    $c_1 \leftarrow c_1 + 1$
  si ... **alors**
    ...
  si ... **alors**
    $c_6 \leftarrow c_6 + 1$
  $x \leftarrow 15091x \mod 64007$ # Calcul du terme suivant
Résultat : $(c_1, ..., c_6)$

– Exercice 18. Il s'agit de calculer cette somme (mais aussi chaque terme) modulo 10^{10}.
– Exercice 19. On calcule $(p-1)!$ directement modulo p.
– Exercice 22. Pour $f_{10^{15}}$, un calcul de proche en proche est exclu. Si on note $F_n = \begin{pmatrix} f_n \\ f_{n+1} \\ ... \\ f_{n+99} \end{pmatrix}$, alors $F_{n+1} = AF_n$ avec A une matrice pas trop compliquée... Il s'agit alors de calculer par *exponentiation rapide* une puissance de matrice... en faisant des réductions modulo 10^4 à chaque étape !