le cnam

Mathématiques Appliquées pour le Génie des Procédés et l'Energétique

Paris, automne 2018

Algorithmique et programmation

Notes du cours 05

Amélie Danlos, Marie Debacq, François Dubois

MA-GPE (5)

Algorithmique et programmation.

F. Deloos
7 novembre 2018

Séance entièrement consacrée à un "cours-TP". On suppose que l'auditeur a un accès à une machine disposant du logiciel Python, soit grâce à son propre ordinateur, soit via le site internet du CNAM où Python est installé (Jupyter).

(1) Calculette

Dans la ligne de commande Python, on effectue des opérations arithmétiques fondamentales:

addition	2+2
multiplication	2*2
exponentiation	2**3 (2^3)
division	3/2

Dans ce dernier cas, le résultat peut être 1.5, ou (dans certaines versions de Python) 1 (!). La commande est alors interprétée en

nombres entiers et le logiciel effectue la division euclidienne de 3 par 2. Pour être certain d'une interprétation de la commande avec des "nombres réels", on pourra écrire 3./2.

(2) Variables.

- On écrit, toujours en ligne de commande, les deux instructions suivantes

```
x = 0
print ('x = ', x)
```

La réponse de l'ordinateur est: x=0. Le fait d'introduire une "variable" x et de lui donner la valeur 0 a une interprétation physique en terme de gestion des données par la machine. A gauche du symbole d'égalité, la lettre "x" signifie qu'on réserve une place en mémoire (un "mot")

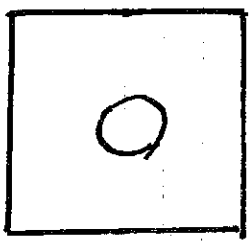

Figure 1. Interprétation physique de l'instruction "x=0".

qui a (pour l'utilisateur !) le nom symbolique "x". Puis à droite du symbole d'égalité, on écrit la donnée (ici le nombre zéro) qui doit être introduite dans la case mémoire (voir la Figure 1).

- Puis vient l'ordre (probablement le plus important de toute l'informatique !) suivant :

$$x = x + 1$$

L'interprétation de cette relation d'un point de vue purement mathématique ne permet pas de définir "x" comme un nombre réel et semble être au contraire contradictoire : "0 = 1". En fait, cette instruction d'égalité est parfaitement codifiée on considère d'abord ce qui est à droite du symbole d'égalité : ici "x+1". La machine interprète cette instruction ainsi : aller à l'adresse en mémoire appelée "x", ajouter 1 à la valeur contenue dans la case nommée "x". Le résultat du calcul est rangé en mémoire à l'adresse écrite à gauche du symbole d'égalité, ici "x". Si on demande d'écrire le contenu de la "variable" x, l'ordinateur répond : $x = 1$.

Si on itère maintenant la même instruction " $x = x+1$ ", le processus décrit à la page précédente se répète : aller chercher les données écrites à droite du symbole d'égalité, effectuer le calcul arithmétique, ranger le résultat obtenu dans la case mémoire indiquée à gauche du symbole d'égalité. Le résultat de cette instruction identique est alors différent : $x = 2$.

(3) Boucle pour le calcul d'intégrales.

- on se propose de calculer

$$\ln 10 = \int_1^{10} \frac{dt}{t}$$

par la formule des rectangles à gauche en utilisant 9 intervalles.
Nous devons générer 9 points : $x = 1, 2, \ldots, 9$.
Puis calculer la valeur de leur inverse. Enfin faire la somme. On peut programmer ces instructions par exemple sous la forme

```
x = 1          # premier point
S = 0          # initialisation
for i in range(0,9):   # itération
    S = S + 1/x        # incrémentation
    x = x + 1          # incrémentation
print('S=', S)         # resultat.
```

on note l'indentation des instructions qui permet à Python de comprendre qu'il faut répéter 9 fois ces deux instructions.

- Si on veut mettre plus de points, on doit complexifier un peu le programme :

```
n = 20      # ou plus ; ou 9 !
dx = (10-1.)/n   # intervalle
x = 1.      # premier point
S = 0
for i in range(0,n):   # boucle
    S = S + dx/x       # incrémentation
    x = x + dx         # incrémentation
print('S=', S)         # résultat.
```

- La boucle "for i..." permet de ne pas avoir à réécrire un grand nombre de fois le même ordre à la machine. C'est une des instructions fondamentales.

- On peut également calculer une approximation par la formule des rectangles à droite avec une minime modification du programme précédent :

```
for i in range(0,n):
    x = x + dx
    S = S + dx/x.
```

on a juste intervertit deux instructions.
La valeur de sc initiale est '1'.
Elle est explicitement utilisée dans le premier calcul. Elle est remplacée par "1+dx" dans le second.

- Une approximation de l'intégrale par la méthode des trapèzes consiste à faire la moyenne des deux résultats précédents (où on utilise la même valeur de n, c'est à dire le même nombre d'intervalles!).

(4) Algorithme de dichotomie.

- Cet algorithme est fondé sur le théorème des valeurs intermédiaires:

> f continue $[a,b] \to \mathbb{R}$, $f(a)\,f(b)<0$.
> alors $\exists \xi \in]a,b[$, $f(\xi)=0$.

Par exemple avec $f(x)=x^2-2$, $a=1$ et $b=2$.
on a $f(1)=-1<0$ et $f(2)=2>0$. On est sûr alors qu'il existe un nombre réel $\xi \in]1,2[$ (c'est $\sqrt{2}$!) tel que $(\xi)^2-2=0$

- L'algorithme de dichotomie consiste à réduire de plus en plus la taille de l'intervalle $[a,b]$. On pose $c=\frac{a+b}{2}$ et on calcule $f(c)$. On suppose pour fixer les idées $f(a)<0<f(b)$.

* Si $f(c) > 0$, alors le théorème des valeurs intermédiaires affirme qu'il existe $\xi \in]a, c[$ tel que $f(\xi) = 0$. On remplace donc l'intervalle $[a, b]$ par l'intervalle $[a, c]$:
$a' = a$; $b' = c$.

* Dans le cas contraire, la racine ξ appartient à $[c, b[$ et on remplace $[a, b]$ par $[c, b]$: $a' = c$; $b' = b$.

* Et on recommence !

• L'algorithme de dichotomie se programme avec l'instruction "if" qui indique un <u>branchement</u>.

$a, b, c = \frac{a+b}{2}$, $f(a)$, $f(b)$, $f(c)$ connus [et notés fa, fb, fc pour les trois derniers].

```
if fc > 0:
    b = c ; fb = fc  # a est inchangé
else:
    a = c ; fa = fc  # b est inchangé
```

Alors l'intervalle $[a, b]$ a été réduit de moitié au cours de ce processus.

• Pour calculer complètement $\sqrt{2}$, on pourra par exemple écrire

```
a=1 ; b=2 ;
fa=a*a-2 ; fb=b*b-2
for i in range(0, 10): #10 itérations
    c=(a+b)/2 ; fc=c*c-2
    if fc>0 :
        b=c ; fb=fc
    else :
        a=c ; fa=fc
print(a,b, fa, fb)
```

On voit les valeurs des variables a et b se rapprocher de 1,414... alors que fa et fb tendent vers 0.

(5) Méthode de Newton

- Elle ne demande qu'une simple boucle.

initialisation $x = x_0$.
boucle $1 \to n$
    calculer $f = f(x)$; $fp = f'(x)$
    incrément $dx = -f/fp$
    modification de x : $x = x + dx$
afficher x.

Le programme précédent n'est pas du langage Python. Il peut être transformé en Python (ou tout autre langage de programmation) sans difficulté.

Par exemple pour calculer $\sqrt{2}$ avec une très grande précision, on pourra écrire

```
x = 1.
for i in range(0, 6):
    f = x * x - 2
    fp = 2 * x
    dx = - f / fp
    x = x + dx
    print ('x = ', x)
```

Et on voit les décimales de $\sqrt{2}$ se figer. On constate même que le nombre de décimales exactes double à chaque itération !

(6) La machine calcule toujours faux !

En effet, si on ajoute à 1 un nombre de l'ordre de 10^{-17}, le résultat est encore égal à 1 ! Le calcul électronique néglige 10^{-17} devant 1. Ce sont les classiques <u>erreurs d'arrondis</u>. On doit toujours garder en tête cette caractéristique technique des ordinateurs. L'art de l'ingénieur consiste à générer des résultats avec un certain nombre de <u>décimales exactes</u> alors que chaque calcul est entaché d'une erreur !

Paris, 9 novembre 2018

Dubois.