séance 08 : Python : un langage de programmation..

G. Khaznadar

08 novembre 2007

Table des matières

1 But de la séance

Les systèmes de mesure physique commandés par ordinateur sont très puissants, ils permettent de faire des milliers de mesures rapidement. Il est intéressant de savoir programmer un ordinateur pour interagir au plus vite avec des grandeurs physiques. Dans cette première séance, les objectifs sont :

– comprendre ce que veut dire programmer

– apprendre quelques mots de base d'un langage de programmation
– savoir réaliser une opération répétitive simplement
– savoir afficher un texte formaté par programme.

2 Un ouvrage de référence

Les séquences qui suivent doivent beaucoup au livre de Gérard Swinnen, « Apprendre à programmer en Python ». Ce livre est diffusé sous la licence GFDL (GNU Free Documentation licence), sa version originale est accessible à http ://www.ulg.ac.be/ ... et il est publié aux éditions O'Reilly (ISBN 2-84177-299-3).

3 La démarche du programmeur

Copine de geek : un site sur la vie sociale des analystes-programmeurs.

3.1 Comment apprendre penser et à réfléchir comme un analyste-programmeur ?

Ce mode de pensée combine des démarches intellectuelles complexes, similaires à celles qu'accomplissent les mathématiciens, les ingénieurs et les scientifiques.

Comme le mathématicien, l'analyste-programmeur utilise des langages formels pour décrire des raisonnements (ou algorithmes). Comme l'ingénieur, il conçoit des dispositifs, il assemble des composants pour réaliser des mécanismes et il évalue leurs performances. Comme le scientifique, il observe le comportement de systèmes complexes, il ébauche des hypothèses explicatives, il teste des prédictions.

L'activité essentielle d'un analyste-programmeur est la résolution de problèmes.

Il s'agit là d'une compétence de haut niveau, qui implique des capacités et des connaissances diverses : être capable de (re)formuler un problème de plusieurs manières différentes, être capable d'imaginer des solutions innovantes et efficaces, être capable d'exprimer ces solutions de manière claire et complète.

La programmation d'un ordinateur consiste en effet à « expliquer » en détail à une machine ce qu'elle doit faire, en sachant d'emblée qu'elle ne peut pas véritablement « comprendre » un langage humain.

3.2 Exercice 1

il murmurait à l'oreille des ordinateurs ...

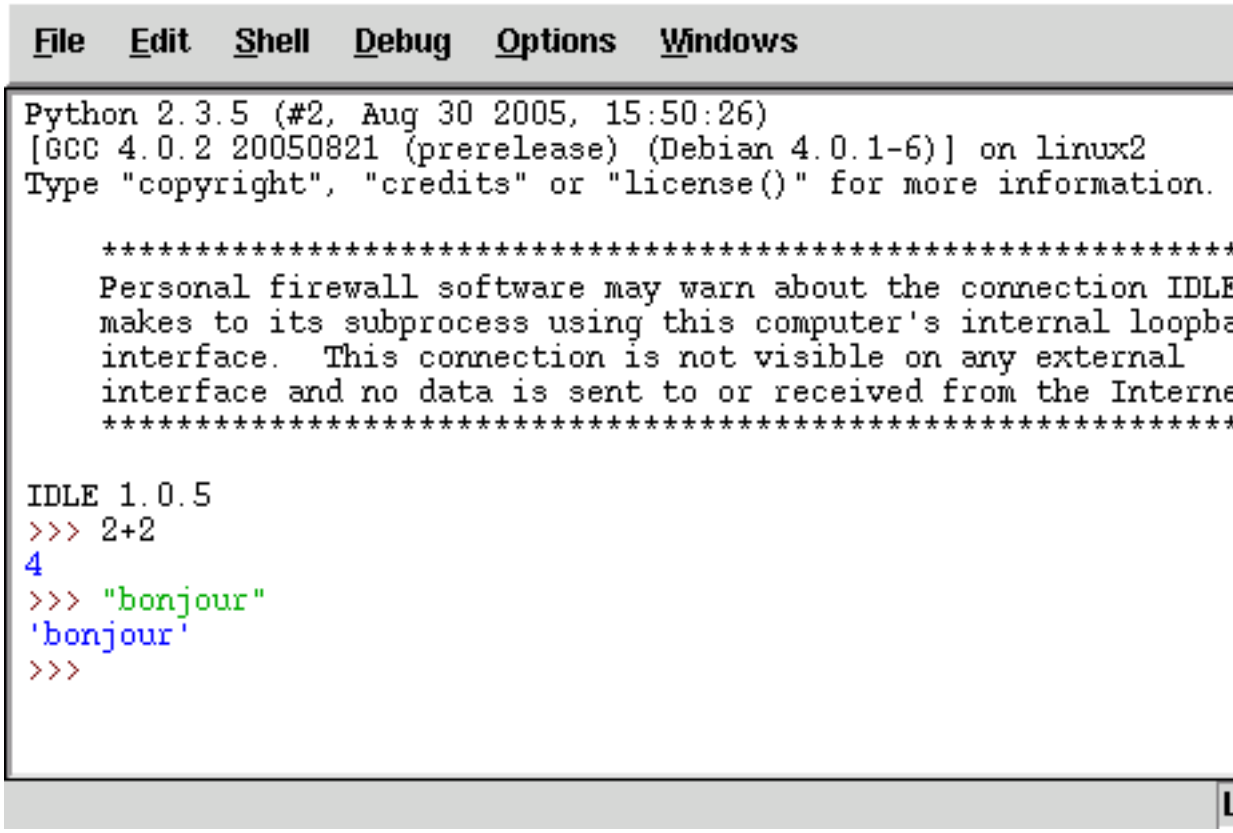
Un élève volontaire va au tableau, et jouera « le rôle de l'ordinateur ». On désigne dans la classe un « analyste-programmeur », qui n'aura le droit d'utiliser qu'un langage très réduit :

- prends une craie, pose une craie, appuie la craie sur le tableau, lève la craie du tableau,
- trace un trait (dans telle direction/sens, de telle longueur)
- recommence l'opération précédente.

Un dessin très simple est donné à l'analyste-programmeur, qui va « commander » l'ordinateur pendant deux minutes. Puis on fait un bilan de la situation, on compare le dessin original et le travail de l'ordinateur, on discute du langage utilisé.

3.3 Premier contact avec un environnement de programmation Python

Lancer la commande « idle ». Une fenêtre s'ouvre, qui permet d'entrer des commandes au clavier et d'obtenir des réponses.



Taper `2+2` puis Entrée (ou une autre opération arithmétique). Que se passe-t-il ?

Taper `"bonjour"` puis Entrée (une autre phrase convient aussi, entre guillemets simples ou doubles). Que se passe-t-il ?

Taper `bonjour` puis Entrée, mais sans guillemets. Que se passe-t-il ?

un message nous avertit que le nom 'bonjour' n'est pas défini.

Avec Python, comme dans beaucoup de langages informatiques, il y a une distinction nette qui est faite entre des littéraux, c'est à dire des choses qui ont un sens toujours le même (c'est le cas du chiffre 2 et de la phrase "bonjour" entre guillemets), et des noms, qui représentent autre chose. Le mot bonjour sans guillemets est considéré comme un nom, mais au démarrage de Python il ne possède aucune valeur prédéfinie.

Taper `bonjour=2+2` puis Entrée. Que se passe-t-il ?

Taper `bonjour` puis Entrée. Que se passe-t-il ?

Cette fois-ci, il n'y a plus de message d'erreur.

Le signe égale de l'instruction tapée précédemment n'a pas du tout le sens mathématique d'égalité. bonjour=2+2 signifie : le nom bonjour représentera

la valeur de 2+2. On dit qu'on fait ainsi une affectation de la valeur 4 au nom (ou à la variable) bonjour.

3.4 Exercice 2

Affecter 2+2 à la variable a, puis affecter 5 à la variable b, et enfin affecter a*b à la variable c. Taper c puis Entrée. Que se passe-t-il ?

3.5 Les mots réservés

Un certain nombre de mots sont réservés et interprétés de façon particulière dans l'environnement python. Vous ne pouvez pas utiliser comme noms de variables les 29 « mots réservés » ci-dessous (ils sont utilisés par le langage lui-même) :

and	assert	break	class	continue	def
del	elif	else	except	exec	finally
for	from	global	if	import	in
is	lambda	not	or	pass	print
raise	return	try	while	yield	

3.6 Les fonctions

Tapez `print "bonjour",2+2==4` puis Entrée (attention au double signe `==`).Que se passe-t-il ?

le mot print n'apparaît pas dans le résultat, mais on voit bonjour True.

« True » signifie « vrai » : en effet, il est vrai que 2+2 est égal à 4. Le double signe `==` a la signification de l'égalité mathématique (essayez ce que donnerait `2+2==5`).

D'autre part, le mot `print` est le nom d'une fonction, qui veut dire « imprimer ».

Quand le programme idle est lancé, celui-ci ajoute une fonction print partout où cela est nécessaire, c'est à dire pour chaque ligne validée par la touche Entrée qui est censée renvoyer une valeur imprimable. Cependant quand vous écrirez un programme autonome en langage Python, vous devrez marquer la fonction print vous-même là où elle est nécessaire pour afficher un résultat.

Tapez `print range(10)` puis Entrée. Que se passe-t-il ?

Le mot `range` désigne une fonction qui doit être suivie d'un paramètre entre parenthèses. Cette fonction renvoie comme valeur une liste, par exemple ici nous obtenons : `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]` qui est la liste des 10 premiers nombres entiers naturels. La fonction `print` fait afficher cette valeur.

Nous connaissons don maintenant deux fonctions : `print` et `range`.

3.7 Exercice 3

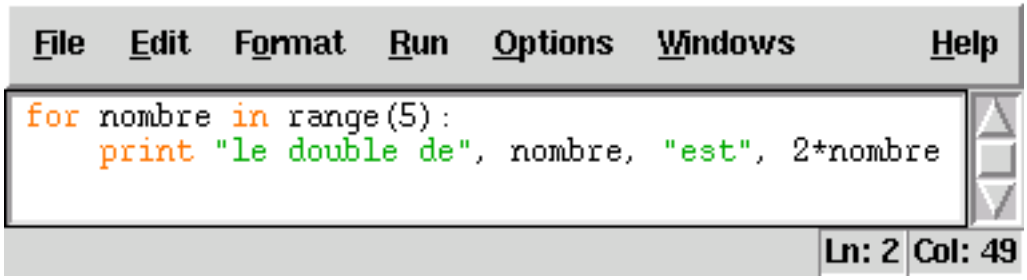
Faites afficher la liste des 2873 premiers nombres entiers naturels.

4 Une première structure de contrôle : la répétition « for »

Tout l'intérêt de la programmation est de pouvoir faire faire par un ordinateur des choses répétitives et fastidieuses, qui sont bien plus simples à laisser faire à une machine qu'à faire soi-même. Voyez combien de temps il vous aurait fallu pour écrire une liste de milliers de nombres, sans aucune erreur !

Nous allons apprendre une première structure de contrôle qui facilite l'écriture de programmes puissants : la répétition de type « for ». Il s'agit d'une portion de programme qui s'écrit généralement sur plusieurs lignes, donc nous allons ouvrir une nouvelle fenêtre, afin de changer de mode d'interaction avec Python, et autoriser l'écriture de programmes longs.

L'ouverture d'une nouvelle fenêtre se fait à l'aide du menu `File -¿ New window` (ou aussi à l'aide du raccourci clavier `Ctrl+N`). Dans cette nouvelle fenêtre, vous inscrirez les lignes suivantes, en faisant attention aux ponctuations et au décalage vers la droite de la deuxième ligne :



Arrangez-vous pour disposer d'un répertoire `html/python1/` puis enregistrez ces lignes dans un fichier (par le menu `File-¿Save` ou le raccourci clavier `Ctrl+S`) de ce répertoire. Par exemple, ce fichier pourrait être `html/python1/test1.py` (notez le suffixe `.py` qui distingue les programmes écrits en langage Python). Vous remarquerez que la coloration du texte apparaît aussitôt qu'on a enregistré le programme sous un nom de fichier avec le suffixe `.py`

Ensuite déclenchez l'exécution de ce petit programme : par le menu `Run -¿ Run Module` ou par le raccourci clavier `F5`. Que se passe-t-il dans la fenêtre initiale de l'application idle ?

4.1 Exercice 4

Créez un nouveau programme, que vous rangerez par exemple dans le fichier `html/python1/test2.py`, qui permet de « copier 200 lignes » d'une punition imaginaire. Par exemple, faites copier 200 fois la ligne « `Ordonne, maître, et je t'obéirai.` ».

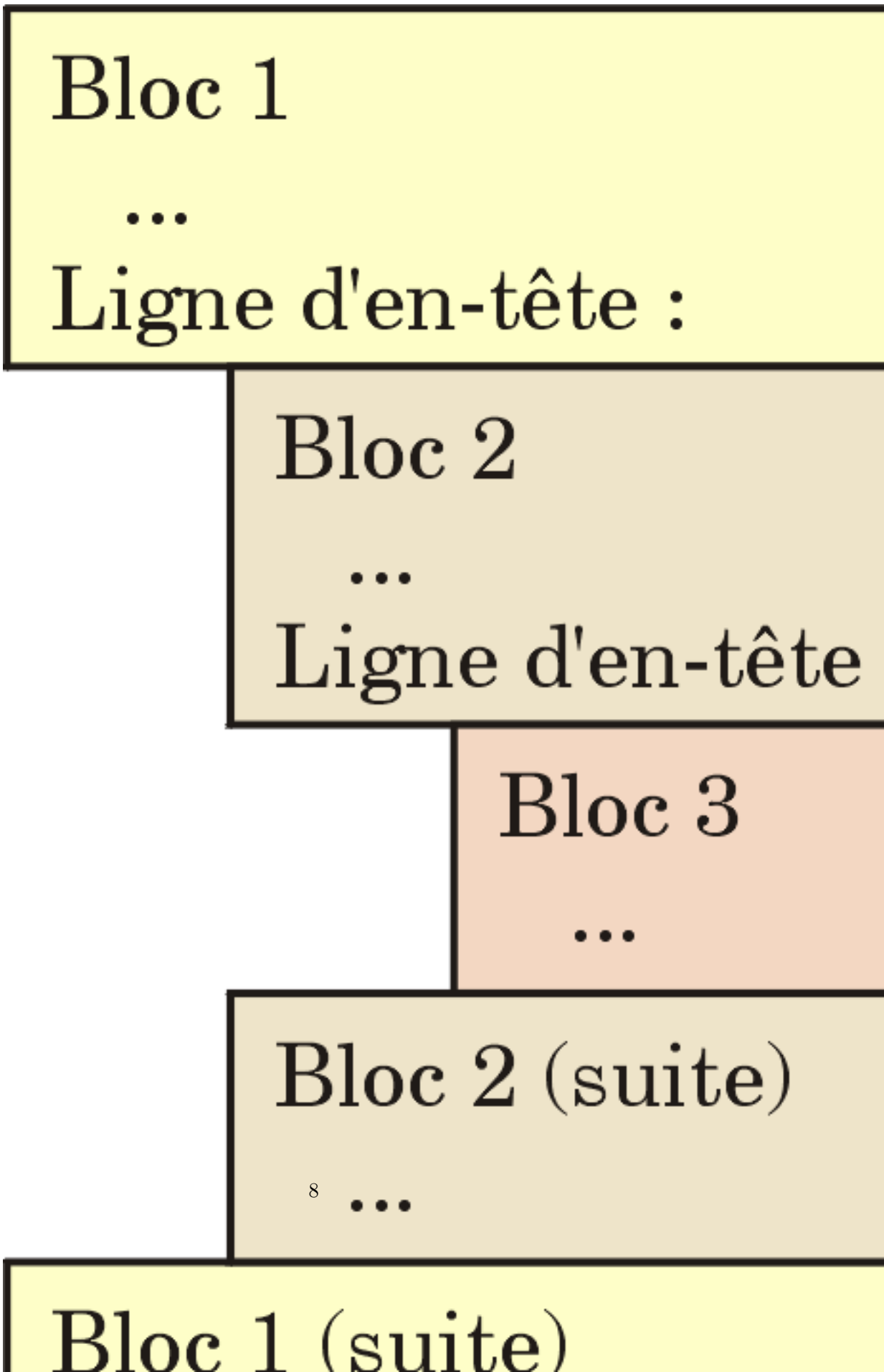
Indication : utiliser la structure de contrôle « for » et la fonction range.

NB : le logiciel Idle vous proposera l'insertion d'une ligne de code, faites en sorte que le programme soit édité automatiquement : cela permet de préciser l'encodage utilisé pour les caractères accentués ou diacritiques (comme éèçàù, etc.)

4.2 Les blocs

Les structures de contrôle peuvent être imbriquées : dans une répétition « for », il peut y avoir une ou plusieurs répétitions « for » imbriquées. À chaque niveau d'imbrication, Python oblige à indenter (décaler vers la droite) le texte du programme à chaque bloc d'instructions.

Nous aurons de nombreuses occasions d'approfondir le concept de « bloc d'instructions ».


Bloc 1
...
Ligne d'en-tête :
Bloc 2
...
Ligne d'en-tête
Bloc 3
...
Bloc 2 (suite)
...
Bloc 1 (suite)

Le schéma ci-dessus en résume le principe.

- Les blocs d'instructions sont toujours associés à une ligne d'en-tête contenant une instruction bien spécifique (for, if, elif, else, while, def, ...) se terminant par un double point « : ».
- Les blocs sont délimités par l'indentation : toutes les lignes d'un même bloc doivent être indentées exactement de la même manière (c'est-à-dire décalées vers la droite d'un même nombre d'espaces). Le nombre d'espaces à utiliser pour l'indentation est quelconque, mais la plupart des programmeurs utilisent des multiples de 4.
- Notez que le code du bloc le plus externe (bloc 1) ne peut pas lui-même être écarté de la marge de gauche (Il n'est imbriqué dans rien).

4.3 Une table de multiplications

Voici quelques lignes de programme pour faire une en-tête de table :

```
ligne="   |"
for a in range(10):
    ligne=ligne+"%3d" %(a)
print ligne
```

Le résultat est le suivant :

```
  |  0  1  2  3  4  5  6  7  8  9
```

Commentaires :

on donne une valeur initiale à la variable ligne : trois espaces et une barre verticale. Puis dix fois de suite, on rallonge la valeur de la variable ligne (l'opérateur + signifie rallonger ou concaténer, dans le contexte des phrases), en y inscrivant la valeur de la variable a, formatée par le symbole `%3d` qui signifie 3 positions, pour une représentation décimale.

Ensuite, il suffit de souligner, puis de réaliser la table de multiplication :

```
ligne="   |"
for a in range(10):
    ligne=ligne+"%3d" %(a)
print ligne

print "---+------------------------------"
for a in range(10):
    ligne="%3d|" %(a)
    for b in range(10):
        ligne=ligne+"%3d" %(a*b)
    print ligne
```

Le résultat est alors le suivant :

```
   -  0  1  2  3  4  5  6  7  8  9
---_------------------------------
  0-  0  0  0  0  0  0  0  0  0  0
  1-  0  1  2  3  4  5  6  7  8  9
  2-  0  2  4  6  8 10 12 14 16 18
  3-  0  3  6  9 12 15 18 21 24 27
  4-  0  4  8 12 16 20 24 28 32 36
  5-  0  5 10 15 20 25 30 35 40 45
  6-  0  6 12 18 24 30 36 42 48 54
  7-  0  7 14 21 28 35 42 49 56 63
  8-  0  8 16 24 32 40 48 56 64 72
  9-  0  9 18 27 36 45 54 63 72 81
```

Il s'agit bel et bien d'une table de multiplication.

4.4 Exercice 5

- reprenez le programme précédent, par copie/collage dans une nouvelle fenêtre Idle.
- transformez-le pour qu'il crée une table d'addition.
- transformez le pour qu'il fasse une table de multiplication, mais avec des nombres de 1 à 10 au lieu des nombres de 0 à 9.
- faites-en autant pour des tables d'addition, puis de soustraction.

4.5 Exercice 6, application à la physique : réfraction d'un rayon lumineux

Nous utiliserons la loi de Descartes de la réfraction, pour résoudre le problème suivant : l'ordinateur demande 3 nombres, l'indice du premier milieu, l'indice du deuxième milieu. Puis l'ordinateur calcule une table des angles d'incidence et de réfraction, de 5 en 5 degrés pour l'angle d'incidence.

Indications :

Si vous utilisez ces premières lignes de programme, la variable n1 et la variable n2 contiendront les valeurs des indices que vous répondrez au début du déroulement du programme. Remarquez bien la ligne « from math import * ». Sans cette ligne, le programme ne peut pas utiliser de fonction mathématique comme sin ($sinus$) ni asin ($sinus^{-1}$) ni la constante pi pour la conversion degré -¿ radian ou radian -¿ degré.

File Edit Format Run Options Windows

```
# -*- coding: iso-8859-15 -*-
##### calcul d'une série de réfractions ##############
from math import *

n1=input("Donnez le premier indice de réfraction : ")
n2=input("Donnez deuxième indice de réfraction : ")
a=5

print "n1 =",n1, "n2 =",n2, "angles tous les", a, "degrés."
```

La formule qui permet de mettre dans i_2 (angle de réfraction) la bonne valeur calculée d'après la loi de Descartes, est très exactement : `i2=180.0/pi*asin(n1/n2*sin(pi/18`

Voici le déroulement du programme (c'est une copie d'écran partielle).

```
Donnez le premier indice de réfraction : 1
Donnez deuxième indice de réfraction : 1.33
n1 = 1 n2 = 1.33 angles tous les 5 degrés.
 0 0.0
 5 3.75732117063
10 7.50210052675
15 11.2214035589
20 14.9014946501
25 18.5273978178
30 22.0824131945
35 25.5475798981
40 28.9010845397
45 32.117631278
```

Il est évident que pour obtenir un programme très court, il **faut** utiliser la commande « for ».

5 barème pour la séance 08

Faites un sous-titre, et les copies d'écran nécessaires, ajoutez au moins une ligne de commentaire pour chaque exercice. Dans le cas de l'exercice 1, qui était un sketch joué en classe, faire une description en une ou deux phrases qui expliquent quelles difficultés sont apparues.

– 3 points pour l'exercice 1
– 3 points pour l'exercice 2
– 3 points pour l'exercice 3
– 3 points pour l'exercice 4
– 3 points pour l'exercice 5
– 3 points pour l'exercice 6
– 2 points pour le message et le lien depuis la page personnelle.