Quelques algorithmes pour explorer les langages

a) Équations

Algorithme 1 : Premier degré

Écrire un algorithme qui affiche l'ensemble des solutions sur $\mathbb{R}$ de l'équation du premier degré $ax + b = 0$.

```
print "resolution de l'equation ax+b=0"

a=input("entrer la valeur de a : ")
b=input("entrer la valeur de b : ")

if (a==0):
        if (b==0):
                print "tout nombre x est solution"
        else:
                print "aucun nombre x n'est solution"
else:
        x=-b*1./a
        print "la solution x a pour valeur approchee :", x
```

Algorithme 2 : Second degré

Programmer l'algorithme suivant :

Résolution dans $\mathbb{R}$ de l'équation du second degré $ax^2 + bx + c = 0$.
On appelle le discriminant de cette équation le nombre $\Delta = b^2 - 4ac$:

- Si $\Delta > 0$, l'équation admet deux solutions réelles distinctes :

$$x_1 = \frac{-b - \sqrt{\Delta}}{2a} \text{ et } x_2 = \frac{-b + \sqrt{\Delta}}{2a}$$

- Si $\Delta = 0$, l'équation admet une solution réelle double : $x_0 = -\frac{b}{2a}$
- Si $\Delta < 0$, l'équation n'admet pas de solution réelle.

```
from math import *

print "resolution de l'equation ax^2+bx+c=0"

a=input("entrer la valeur de a : ")
b=input("entrer la valeur de b : ")
c=input("entrer la valeur de c : ")

Delta=b*b-4*a*c

print "Delta :", Delta

if (Delta>0):
        x1=(-b-sqrt(Delta))*1./(2*a)
        x2=(-b+sqrt(Delta))*1./(2*a)
        print "les solutions x1 et x2 ont pour valeurs approchees :", x1, "et :", x2
else:
        if (Delta==0):
                x0=-b*1./(2*a)
                print "la solution x a pour valeur approchee :", x0
        else:
                print "il n'y a pas de solution reelle"
```

b) Fonctions

Algorithme 3 : Balayage d'une fonction

On considère la fonction f définie sur $\mathbb{R}$ par $f(x) = x^2 - 5x + 7$
Écrire un algorithme permettant d'afficher les valeurs de $f(x)$ pour x allant de 0 à 10 par valeurs entières.

```
from liste import *
from graphique import *
```

```
fenetreGraphique()

nouveauRepere(xmin=0,xmax=10,ymin=0,ymax=60)

for x in entiers(0,10):
        y=x*x-5*x+7
        print "f(",x,")=",y
        dessinePoint(x, y)

afficheGraphique()
```

Algorithme 4 : Dichotomie

On considère la fonction f définie sur $\mathbb{R}$ par $f(x) = x^3 - 4x^2 + 8x - 7$

Déterminer, par dichotomie, une valeur approchée de la solution de l'équation $f(x) = 0$ sur l'intervalle $[1; 2]$, avec une précision d'au moins 10^{-2}.

```
def f(x):
        return x*x*x-4*x*x+8*x-7

a=1
b=2

while (b-a>0.01):
        c=(a+b)/2.
        if (f(a)*f(c)>0):
                a=c
        else:
                b=c

print "la valeur cherchee est comprise entre", a, "et", b
```

Algorithme 5 : Méthode d'Euler

f est une fonction dérivable sur $\mathbb{R}$ telle que : $\begin{cases} f(0) = 1 \\ \text{pour tout } a \in \mathbb{R},\ f'(a) = 4 - 2a \end{cases}$

En utilisant l'approximation affine $f(a+h) \approx f(a) + h \times f'(a)$, écrire un algorithme calculant successivement des valeurs approchées de $f(0,5)$, $f(1)$, $f(1,5)$, ..., $f(4)$.

Modifier l'algorithme pour utiliser un pas de $0,1$.

Modifier l'algorithme pour utiliser un pas quelconque (entré par l'utilisateur).

```
from graphique import *

fenetreGraphique()

nouveauRepere(xmin=0,xmax=4,ymin=0,ymax=10)

f0=1

def f1(a):
        return 4-2*a

x=0
fx=f0
dessinePoint(0, f0)

h=0.5
while(x<4):
        fx=fx+h*f1(x)
        x=x+h
        dessinePoint(x, fx)
        print "valeur approchee de f(",x,") :",fx

afficheGraphique()
```

c) Suites

Algorithme 6 : Grains de riz sur un échiquier

On remplit un échiquier (64 cases) de grains de riz en posant :

- sur la première case : 1 grain de riz ;
- sur la deuxième case : le double, soit 2 grains de riz ;
- sur la troisième case : le double, soit 4 grains de riz ;
- et ainsi de suite, jusqu'à ce que les 64 cases soient remplies.

Combien de grains de riz sont nécessaires ?

```
from liste import *

cases=liste()
cases[1]=1
for n in entiers(2,64):
        cases[n]=cases[n-1]*2

print "echiquier :", cases

somme=0
for n in entiers(1,64):
        somme=somme+cases[n]

print "somme :", somme
```

Algorithme 7 : Babylone

Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie par $\begin{cases} u_0 = 2 \\ u_{n+1} = \dfrac{1}{2}\left(u_n + \dfrac{2}{u_n}\right) \end{cases}$

Écrire un algorithme, puis le modifier, pour calculer (en valeur approchée) :

a) les dix premiers termes de la suite (u_n)

b) les premiers termes de la suite (u_n), jusqu'à obtenir 6 décimales correctes de $\sqrt{2}$

```
from liste import *
from math import *
import random

u=liste()
n=0
u[0]=2.
precision=0.000001
while (u[n]>sqrt(2)+precision):
        u[n+1]=(u[n]+2/u[n])/2
        print "u[", n+1, "]=",u[n+1]
        n=n+1

print "approximation au rang n=", n, ", u[",n,"]=", u[n]
print "racine de 2 : ", sqrt(2)
```

d) Probabilités, statistiques

Algorithme 8 : Tirage aléatoire

On lance deux dés à 6 faces, et on additionne les valeurs lues sur les deux dés.
Quel résultat a-t-on le plus de chance d'obtenir ?

```
from liste import *
from graphique import *
from random import *

nbSimulations=1000
simulations=liste()

for n in entiers(1,nbSimulations):
        de1=choice(entiers(1,6))
        de2=choice(entiers(1,6))
        simulations[n]=de1+de2
```

```
print "simulations :", simulations

effectifs=liste()
for n in entiers(0,12):
        effectifs[n]=0

for n in entiers(1,nbSimulations):
        effectifs[simulations[n]]= effectifs[simulations[n]]+1

print "effectifs :", effectifs

fenetreGraphique()
dessineDiagrammeBatons(effectifs)
afficheGraphique()
```

Algorithme 9 : Statistiques

(Source : *L'induction statistique au lycée*, Philippe Dutarte, IREM Paris-Nord, éditions Didier 2005.)

L'université de Montréal a mené, en 2002, une étude sur l'influence des pesticides sur la proportion de garçons et de filles à la naissance. Cette étude a été menée dans la ville d'Ufa (fédération de Russie) auprès de personnes ayant été exposées à des pesticides contenant de la dioxine : on a observé chez ces personnes la naissance de 91 garçons et 136 filles.

Écrire un algorithme permettant de simuler 227 naissances aléatoires de garçons et de filles. Remarque : statistiquement, la probabilité pour un nouveau-né d'être un garçon est $p \approx 0,512$.

Modifier l'algorithme pour produire 1000 simulations. La proportion observée dans la ville d'Ufa vous semble-t-elle due au hasard ?

```
from liste import *
from graphique import *
from random import *

nbSimulations=1000
simulations=liste()

for n in entiers(1,nbSimulations):
        simulations[n]=0
        for i in entiers(1,227): #227 naissances
                naissance=random()
                if (naissance<0.512):
                        simulations[n]=simulations[n]+1

print "simulations :", simulations

effectifs=liste()
for n in entiers(0,227):
        effectifs[n]=0

for n in entiers(1,nbSimulations):
        effectifs[simulations[n]]= effectifs[simulations[n]]+1

print "effectifs :", effectifs

cumul=0
for n in entiers(0,91):
        cumul=cumul+effectifs[n]

print "nombre de simulations donnant entre 0 et 91 garcons :", cumul

fenetreGraphique()
dessineDiagrammeBatons(effectifs)
afficheGraphique()
```