Epidémiologie, l'urne de Pólya, partie II

Se mettre par groupe de deux. Traiter la fiche dans l'ordre. Un compte de rendu de TP devra être rendu avec les réponses écrites aux questions, les recherches documentaires et les listings des programmes. Pour l'algorithme de la partie 1 et les programmes de la partie 2, il faudra les faire valider par le professeur avant de les imprimer.
L'activité et la participation pendant la séance seront des éléments important de l'évaluation.

1 Simulation de l'urne de Pólya avec un algorithme

L'urne de Pólya constitue un modèle basique de propagation d'une épidémie. Nous avons déjà simulation l'évolution d'une urne de Pólya avec un tableur, nous allons réaliser une simulation avec un algorithme, écrit avec le logiciel Algobox.

Les hommes utilisent des algorithmes depuis les temps anciens pour calculer (algorithme d'Euclide pour le calcul de PGCD), construire, cultiver, cuisiner ...

Définition 1

Un **algorithme** est une procédure de calcul bien définie qui prend en **entrée** une valeur ou un ensemble de valeurs (on parle d'**instance** du problème) et qui retourne en **sortie** une valeur ou un ensemble de valeurs.

Pour être exécuté par un ordinateur, un algorithme doit être traduit dans un langage de programmation. Algobox est un mini-langage de programmation dont la syntaxe est très proche de celle d'un algorithme écrit en Français.

1. Choisir un personnage parmi les suivants, puis faire une recherche documentaire pour retracer en une dizaine de lignes son apport à l'histoire des algorithmes et de l'informatique :

- Al Kwarizmi
- Charles Babbage
- Ada Lovelace
- Alan Turing

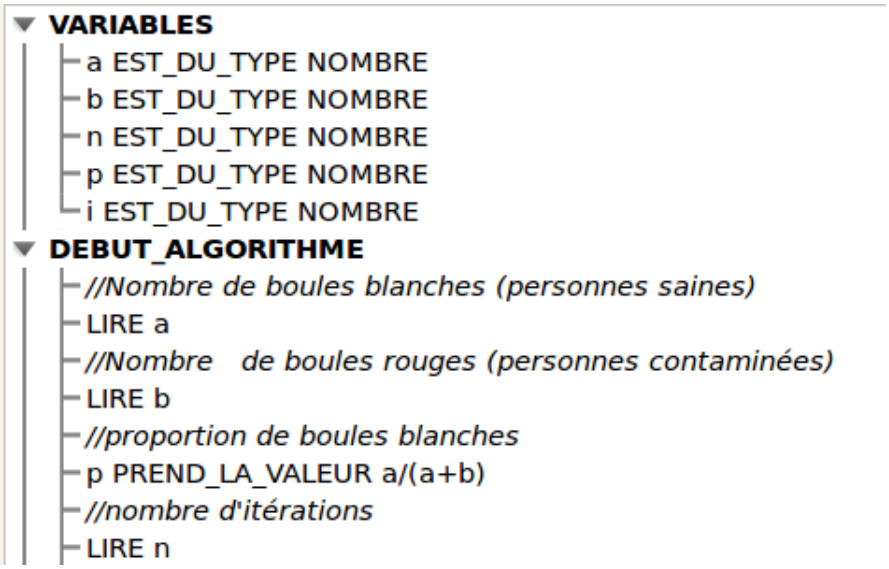
2. Ouvrir le logiciel Algobox et enregistrer un nouveau fichier `Polya.alg`. Nous écrirons l'algorithme au fil des questions, les modèles de syntaxe sont donnés dans des captures d'écrans.

Pour commencer la rédaction de notre algorithme, nous allons d'abord déclarer les variables utilisées.

Définition 2

- **Variable :**
  Une **variable** désigne un emplacement de la mémoire de l'ordinateur. Elle est identifiée par un **nom** (identificateur) et contient une **valeur** (ou pas). C'est une boîte portant une étiquette (son nom) et contenant une valeur.
  Comme il existe des boîtes différentes pour ranger des chaussures, des outils ou un texte (une enveloppe), les variables peuvent être de différents **types** : nombre entier, nombre à virgule, chaîne de caractères, booléen (valeur logique Vrai ou Faux), tableau ....
- **Affectation de variable :**
  Lorsqu'on range une valeur dans une variable, on dit qu'on **affecte** cette valeur à la variable. Il faut que le type de la valeur soit le même que celui de la variable.Par exemple, pour affecter la valeur 2 à la variable A de type entier on écrira l'instruction : *A prend la valeur 2* .
- **Interaction avec l'utilisateur**
  - **Entrée des données :** Au moment de l'exécution de l'algorithme, celui-ci s'arrête à cette instruction et attend l'entrée d'une valeur par l'utilisateur pour qu'elle soit affectée à une variable. Par exemple, pour saisir une valeur et l'affecter à la variable A on écrira l'instruction : *Saisir A*.
  - **Sortie des données :** On distingue l'affichage de la valeur d'une variable par l'instruction *Afficher A* de l'affichage d'un message (noté entre guillemets) comme *Afficher« Valeur de A : »*.

Cinq variables `a,b,n,p` et `i` sont déclarées. Dans Algobox, les lignes commençant par // sont des commentaires, ce ne sont pas des instructions de l'algorithme. Les rôles de `a,b,n,p` sont commentés ci-dessous et la variable `i` servira de compteur de boucle.

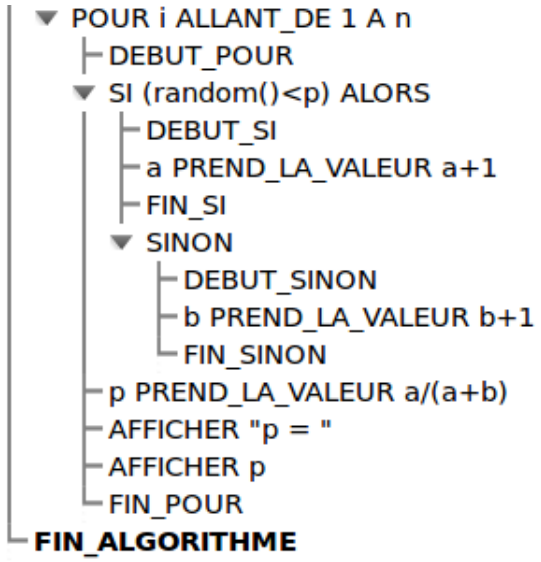


3. Pour réaliser `n` itérations (ou évolutions) de l'urne de Polya, il faut répéter `n` fois la même simulation : on tire une boule dans l'urne, si elle est blanche on la remet avec une boule blanche, sinon, on la remet avec une boule rouge.

Définition 3

On a donc besoin d'imbriquer deux structures de contrôle dans notre algorithme :

- une **boucle** `Pour` qui répète `n` fois le même bloc d'instructions, le décompte étant assuré par l'incrémentation de la variable compteur `i` de sa valeur initiale 1 jusqu'à sa valeur finale `n`.
- ce bloc répété étant constitué d'une **structure conditionnelle** :
  `Si` *boule tirée blanche*
  `Alors` *rajouter 2 boule blanches*
  `Sinon` *rajouter 2 boules rouges*



4. Réaliser plusieurs simulations de la composition de l'urne au bout de $n = 50$ itérations en fixant une composition initiale de l'urne $(a\,;\,b) = (1\,;\,1)$. On pourra faire d'autres tests avec des compositions initiales différentes telles que $(a\,;\,b) = (2\,;\,1)$.

Peut-on faire les mêmes observations que pour les simulations avec le tableur ?

2 Présentation du langage Python et simulation de l'urne de Polya

Python est un langage de programmation utilisé dans le monde professionnel et dans le monde éducatif :
Site officiel : `http://www.python.org/`

2.1 Opérations,affectations, entrées, sorties

Ouvrir le logiciel Idle Python3 (outil de développement de programme en langage Python version 3). On se trouve devant une fenêtre avec un prompt >>> qui attend la saisie d'une commande. La console Python ne sert pas à écrire des programmes mais à tester des commandes ou à faire des calculs. On saisit une instruction après le prompt >>> et on la termine par un retour à la ligne (appui sur « Entrée ») puis l'interpréteur Python l'exécute.

Tout ce qui se situe à droite d'un symbole « dièse » # jusqu'au changement de ligne, est ignoré par l'interpréteur. On peut ainsi insérer des **commentaires**.

En Python, on ne déclare pas le type des variables. Le type d'une variable est celui de la valeur qui lui est affecté. L'affectation de l'entier 2 à la variable `a` se note `a = 2`.

Tester dans la console la série d'instructions suivantes :

- Quelques calculs

```
>>> 3*5+7-(10+3)
>>> 2**3 # pour l'exponentiation (
    et non pas 2^3)
>>> 2,5/3
>>> 2.5/3
>>> 5/2
>>> 5//2
>>> 5%2
```

- Affectations de variables, opérations sur les variables, type d'une variable :

```
>>> a = 5 #affectation de variable
>>> a = a+1
>>> print(a) #affichage, instruction de sortie
>>> print(type(a))
>>> print("Hello World") #les chaines de caractères entre guillemets
>>> b = input('Entrez un entier')
>>> print(b,type(b))
>>> print(a+b)
>>> b = int(b) #on transforme b en entier
>>> print(a+b)
>>> c,d = "Hello ","World"
>>> print(c+d)
>>> e = 3.5
>>> print(type(e))
```

Si une variable contient :

- un nombre entier, elle est de type `int`
- un nombre à virgule (en fait le séparateur décimal est le point), elle est de type `float`
- une chaîne de caractères, elle est de type `str`

2.2 Edition d'un programme, instructions conditionnelles

Pour écrire le code source d'un programme, ouvrir l'éditeur de texte en cliquant sur `File/New Window`.

1. Saisir le programme suivant puis le tester en cliquant sur `Run/Module` :

```
# \n symbolise un retour à la ligne
prix= float(input("Entrez le prix initial : \n"))
taux = float(input("Entrez le taux de variation : \n"))
prix = prix*(1+taux/100)
print("le prix après augmentation est de : ",prix,"euros")
```

2. Ecrire de même un programme qui demande le prix final et le taux de variation t et qui calcule le prix initial avant la variation de t %.

3. Ecrire un programme qui demande à l'utilisateur sa date de naissance et qui calcule son âge.

4. Le programme suivant demande à l'utilisateur de saisir un entier n puis affiche en sortie n si n est positif ou nul ou $-n$ si n est négatif.

L'instruction `if` commande un bloc d'instructions conditionnelles : si le test après le `if` est vrai, le bloc qui le suit est exécuté sinon c'est le bloc suivant le `else`. En python, un bloc d'instructions, qui est un déroutement du flot d'instructions principal, est délimité par son indentation supérieure.

```
n = int(input("Entrez un entier : \n"))
if n>=0:
    print("Résultat :",n)
else:
    print("Résultat :",-n)
```

a. Ecrire un programme qui demande à l'utilisateur de saisir deux entiers a et b et qui affiche leur écart absolu. Par exemple l'écart absolu de $a = -3$ et $b = -2$ est 1, celui de $a = -5$ et $b = 7$ est 12.

b. Ecrire un programme qui demande à l'utilisateur de saisir trois entiers a,b et c et qui retourne le plus grand de ces trois entiers.

5. Beaucoup de fonctions sont préprogrammées en Python mais sont stockées dans des modules qu'il faut importer avec la commande import. Nous utiliserons les modules `math` et `random`.

```
from math import*
from random import*

print("Racine carré de 2 : ",sqrt(2))
print("Nombre aléatoire entre 0 et 1 :",random())
print("Entier aléatoire entre 1 et 6 :",randint(1,6))
```

Ecrire un programme qui calcule la somme de deux dés à 6 faces après leur lancer.

2.3 Boucle

Programmation en Python d'une boucle Pour :

```
Pour i allant de 1 à n faire
  Bloc d'instructions
FinPour
```

```
for i in range(1,n+1):
   Bloc d'instructions
```

On peut aussi écrire `for i in range(n)`, dans ce cas l'indice `i` varie de 0 à n exclu par incrémentations de 1.

Exemple du calcul de la somme des n premiers entiers consécutifs $1+2+3+\cdots+n$:

```
1 n = int(input("Entrez un entier : \n"))
2 s = 0
3 for i in range(1,n+1):
4    s = s+i
5 print("Somme =",s)
```

1. Ecrire un programme qui demande à l'utilisateur de saisir son nombre n de notes puis qui demande les n notes et calcule la moyenne.

2. Ecrire un programme qui demande à l'utilisateur de saisir un entier n et qui affiche les résultats obtenus pour la simulation de n lancers d'un dé à six faces. Modifier le programme pour qu'il compte les 6 apparus.

2.4 Simulation de l'urne de Pólya avec Python

Traduire en Python l'algorithme étudié en section 1 qui simule l'évolution d'une urne de Polya au bout de `n` itérations.

```
1  from random import*
2
3  a = int(input("Entrez le nombre de boules blanches : \n"))
4  ......
5  ......
6  ......
7  for i in range(1,n+1):
8      if random().....:
9          ........
10     else:
11         ........
12     p = a/(a+b)
13     print('Itération :',i,'p =',p)
```