Travaux dirigés : Algorithmes programmations, statistiques,simulations

Exercice 1

1. Soit un réel x de l'intervalle [0 ; 1] et deux nombres entiers a et b tels que $a < b$.
   On a $0 \leqslant x \leqslant 1$.
   (a) Dans quel intervalle est le nombre $x(b-a)$.
   (b) Dans quel intervalle est le nombre $x(b-a)+a$

2. On donne l'algorithme suivant et une traduction en langage Python :

$x \leftarrow$ nombre réel au hasard de [0 ; 1]
$i = 0$
$c \leftarrow$ choisir un entier c
Tant que c différent de n faire :
    $i \leftarrow i+1$
    Si c<n faire :
        $c \leftarrow$ choisir un entier c
    Sinon faire :
        $c \leftarrow$ choisir un entier c
Fin Tant que

```python
from math import*
from random import*

def cherche(a,b):
    n=int(random()*(b-a))+a
    i=0
    c=int(input('entrer un nombre :'))
    while c!=n :
        i=i+1
        if c<n:
            c=int(input('plus grand :'))
        else :
            c=int(input('plus petit :'))
```

python7exercice1.py

(a) Ouvrir le fichier Pythonexercice1.py qui se trouve sur le réseau du lycée.
(b) Exécuter cet algorithme en utilisant la commande $cherche(1,20)$ en donnant les étapes dans un tableau comme celui-ci, le nombre d'étape est variable suivant les valeurs de c choisies :

étape i	nombre c choisi par l'utilisateur
0	...
...	...
...	...

(c) Que fait cet algorithme ? Que permet de compter la variable i ?
(d) Existe-t-il une stratégie pour faire cet algorithme ? Décrire cette stratégie par un ou plusieurs exemples pour la commande $cherche(1,20)$ en complétant un tableau comme celui-ci :

étape i	a	b	c
0	1	20	...
1	...	...	...
...	...	...	...

(e) Écrire un algorithme pour cette stratégie et la traduire par un programme en Python.
(f) Existe-t-il une autre stratégie pour faire l'algorithme associé à la commande $cherche(a,b)$? Si oui, écrire un algorithme pour cette stratégie et la traduire par un programme en Python.

Exercice 2 : comparaison des deux algorithmes

1. on donne l'algorithme et son programme suivants :

   L_{dicho} = []

   L_{hasard} = []

   Pour i variant de 1 à n :

   Ajouter à L_{dicho} l'élément ChercherDichotomie(a,b)

   Ajouter à L_{hasard} l'élément ChercherHasard(a,b)

   Fin Pour

   Calculer $moyenne(L_{dicho})$, $EcartType(L_{dicho})$

   Calculer $moyenne(L_{hasard})$, $EcartType(L_{dicho})$

```
def comparaison(a,b,n):
    L_dicho=[]
    L_hasard=[]
    for i in range(0,n):
        L_dicho.append(OrdinateurChercheDichotomie(a,b))
        L_hasard.append(OrdinateurChercheHasard(a,b))
    return [moyenne(L_dicho),ecarttype(L_dicho),moyenne(L_hasard),ecarttype(L_hasard)]
```

python7exercice1.py

2. Recopier les fonctions moyenne et écart-type crée lors du dernier TP, les coller dans le fichier puis saisir le programme ci-dessous.

3. Comparer les critères statistiques des deux séries à l'aide de la commande $comparaion(1,20,1\,000)$. Quelle série est la plus efficace ?

```
'''
dichotomie
recherche d'un nomrbe au hasard sur un intervalle [a;b]
a<b, a et b sont des entiers
'''

from math import*
from random import*
import matplotlib
import matplotlib.pyplot as plt


def cherche(a,b):
    n=int(random()*(b-a))+a
    c=int(input('entrer un nombre :'))
    while c!=n :
        if c<n:
            c=int(input('plus grand :'))
        else :
            c=int(input('plus petit :'))
    #if c==n:
        #print('n=',n,' vous avez trouvé le nombre n !')


def OrdinateurChercheDichotomie(a,b):
    n=int(random()*(b-a))+a
    i=0
    c=int((a+b)/2)
    while c!=n :
        if c<n:
            a=c+1
        else :
            b=c-1
        i=i+1
        c=int((a+b)/2)
        #print(n,a,b,c)
    #if c==n:
        #print('n=',n,' vous avez trouvé le nombre n en ',i,' coups')
    return i

def OrdinateurChercheHasard(a,b):
    n=int(random()*(b-a))+a
    i=0
    c=int(random()*(b-a))+a
    while c!=n :
        if c<n:
            a=c+1
        else :
            b=c-1
        i=i+1
        c=int(random()*(b-a))+a
        #print(n,a,b,c)
    #if c==n:
        #print('n=',n,' vous avez trouvé le nombre n en ',i,' coups')
    return i


def moyenne(L):
    s=0
    for i in range(0,len(L)):
        s=s+L[i]
    return s/len(L)

def variance(L):
    s=0
    m=moyenne(L)
```

```
67      for i in range(0,len(L)):
68          s=s+(L[i]-m)**2
69      return s/len(L)
70
71  def ecarttype(L):
72      return sqrt(variance(L))
73
74  def comparaison(a,b,n):
75      L_dicho=[]
76      L_hasard=[]
77      for i in range(n):
78          L_dicho.append(OrdinateurChercheDichotomie(a,b))
79          L_hasard.append(OrdinateurChercheHasard(a,b))
80      plt.subplot(311)
81      bins = [x + 0.5 for x in range(0, max(L_hasard))]
82      plt.hist([L_dicho, L_hasard], bins = bins, color = ['blue', 'red'],edgecolor = 'grey', hatch = '/', label = [
        'dichotomie', 'hasard'],histtype = 'bar') # bar est le defaut
83      plt.ylabel('effectifs')
84      plt.xlabel('nombre de coups')
85      plt.title('comparaison des deux stratégies')
86      plt.legend()
87      plt.subplot(313)
88      plt.boxplot([L_dicho, L_hasard],vert=False)
89      plt.xlim(0, max(L_hasard))
90      plt.title('comparaison de deux séries')
91
92      plt.show()
93      return  [moyenne(L_dicho),ecarttype(L_dicho),moyenne(L_hasard),ecarttype(L_hasard)]
```

python7exercice1correction.py