Des mini-projets pour la prochaine journée de formation

Mini-projet 1 : La distribution de la somme des faces de 2 dés

L'activité portant sur la distribution de la somme de deux dés est largement connue des professeurs de mathématique ; de façon classique, elle constitue une bonne activité à proposer sur tableur. Voir cette activité proposée sous la bannière « algorithmique » pourrait sembler faire double usage sur ce que chacun d'entre nous a déjà développé dans ses classes ; il n'en est rien :

- la mise en œuvre d'algorithmes pour simuler cette expérience met en œuvre de façon efficace les 3 champs : instructions élémentaires, boucles et itérateurs, et instructions conditionnelle ;
- le développement de ces algorithmes sur ordinateur donnera des simulations produisant des échantillons de grande taille.

I L'algorithmique enseignée

On propose ici trois algorithmes :

- L'algorithme 1 simule une seule fois l'expérience.
- L'algorithme 2 simule N fois l'expérience.
- L'algorithme 3 simule N fois l'expérience et renvoie à la fin la fréquence du 7.

1. Les trois algorithmes en détails.

a) L'algorithme 1.

On crée deux variables D1 et D2 affectées de « la » valeur int(6*random)+1.

La variable S donnera ensuite la somme des 2 dès, ainsi elle est le résultat du calcul D1+D2. On fait afficher la variable S.

b) L'algorithme 2.

On donne au départ de l'algorithme la variable N.
On crée une variable k (compteur de boucles) qui prend au départ la valeur 0.
Tant que k est inférieur à (N+1), on effectue la boucle suivante :
Algorithme 1 et on ajoute 1 à k

c) L'algorithme 3.

On donne au départ de l'algorithme la variable N.
On crée une variable k (compteur de boucles) qui prend au départ la valeur 0.
On crée une variable n (compteur de réalisation) qui prend au départ la valeur 0.
Tant que k est inférieur à (N+1), on effectue la boucle suivante :
Algorithme 1 et [si S = 7, on ajoute 1 à N] et on ajoute 1 à k
A la fin, on fait afficher n/N.

2. Liste des instructions élémentaires, boucles et itérations, instructions conditionnelles.

Dans ce paragraphe, nous listons les différentes instructions que nous allons présenter aux élèves. L'objectif de ce paragraphe est d'aider le professeur à enseigner l'algorithmique.

a) Les instructions élémentaires.

Rappels : il s'agit des affectations, calcul, entrée, sortie. Les deux variables D1 et D2 sont affectées de « la » valeur int(6*random)+1
La variable S donne ensuite la somme des 2 dès, ainsi elle est le résultat du calcul D1+D2. Pour les algorithmes 2 et 3, l'utilisateur fixe en entrée la taille N de l'échantillon désirée. Dans l'algorithme 1, on fait afficher à la sortie la valeur de la variable S.

b) Boucle et itérateur. L'algorithme 2 produit un échantillon de taille N, pour cela on parcourt N fois le même groupe d'instructions, c'est une boucle : lancer les 2 dès et ajouter 1 au compteur de boucles. Un calcul qui est placé dans un algorithme afin d'être refait « éventuellement » plusieurs fois est appelé calcul itératif (ou parle aussi d'itération). On va créer des compteurs : - un compteur de boucles k : ainsi dans les algorithmes 2 et 3, un

compteur k sera mis en place. Il partira de la valeur 0, et sera incrémenté de 1 à chaque parcours de la boucle (tant que sa valeur maximale ne sera pas dépassée) ; - un compteur de réalisation n : ainsi dans l'algorithme 3, on attend à la fin de l'algorithme, le nombre d'occurrences du 7 sur l'échantillon ; pour cela on mettra en place un compteur n qui vaudra au départ 0 et qui sera incrémenté de 1 à chaque fois qu'il aura succès.

c) Instructions conditionnelles. Tant que … La condition de fin d'expérience sera un test de fin de boucle. Pour être plus précis, lorsque l'on souhaite produire uniquement une série de N expériences, on introduit un compteur de boucles k qui au départ vaut 0, on met une boucle dans laquelle le compteur est incrémenté à chaque fois que la boucle est exécutée. La boucle devra être exécutée tant que le compteur est strictement inférieur à N+1. L'instruction qui consiste à parcourir la boucle sera donc conditionnée par le fait que le compteur ne dépasse pas strictement N, nous avons donc une instruction soumise à condition, que l'on appelle une instruction conditionnelle. Si …alors… L'algorithme 3 a pour intention de faire ressortir le nombre de fois où l'on obtient la valeur 7. Pour ce faire, nous plaçons à l'intérieur de la boucle, une instruction conditionnelle portant sur le fait que S vaille 7 ou non, avec incrémentation dans l'affirmative d'un compteur de réalisation n de 1 (le compteur étant initialisé à 0).

II) Développement des algorithmes 1, 2 et 3 avec Python.

Le générateur de nombre pseudo aléatoire sur Python. Il a deux façons de produire des nombres aléatoires avec Python : avec randint ou avec random.

```
>>> from random import randint
>>> from random import random
>>> random()
0.39721444426193631
>>> randint(1,6)
2
```

Pour rester cohérent avec un usage éventuel des calculatrices, nous choisissons d'utiliser la fonction random.

```
>>> int(6*random())+1 2
```

III) Prolongements : Distribution de la somme des faces

Ce programme donne la distribution de la somme des faces. Il utilise des listes et pour limiter les difficultés de compréhension, nous montrons en préambule comment construire une liste simple (celle des valeurs possibles de l'expérience.

1. Préambule : Construction de la liste des valeurs. On construira dans ce programme une liste nommée valeurs qui réunit les valeurs possibles de l'expérience :
   Valeurs = [2 , 3, 4 ,5, 6, 7,8, 9 ,10,11,12] Pour ce faire, on automatise le remplissage de cette liste :

Séquence	Explications

valeurs =[]
i=2
while i < 13 :
  valeurs.append(i)
  i=i+1
	Au départ, la liste est vide.
	i prend la valeur 2 au départ
	tant que i est < 13,
	on ajoute i à la liste Valeurs
	on incrémente i de 1.

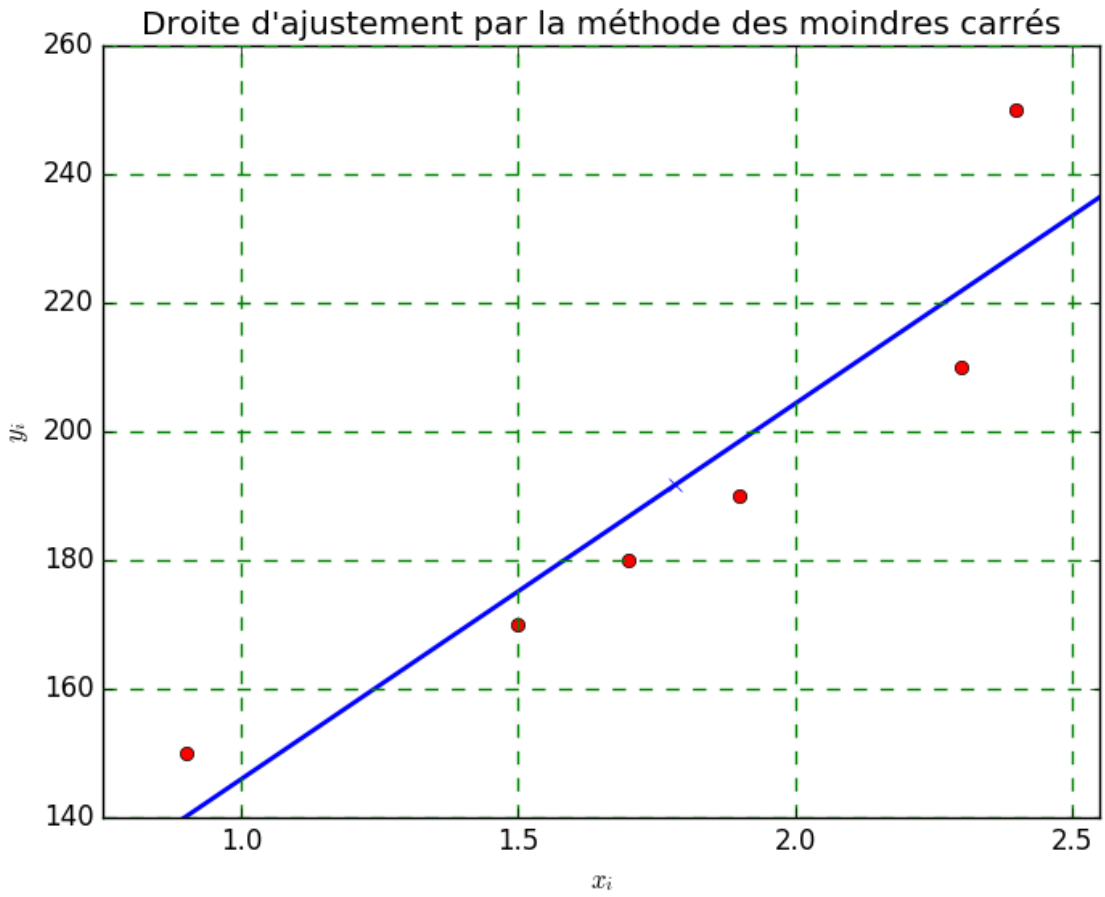
2. Construction de la liste des distributions.
   A vous de jouer : Construire la liste des résultats de l'expérience qui donnera pour notre échantillon la distribution de la somme des faces, par exemple pour 1000 tirages.

Mini-projet 2 : Ajustement affine par la méthode des moindres carrés

Objectif : Créer un programme Python qui utilise la librairie Matplotlib pour tracer le nuage de points associé à une série statistique à deux variables, son point moyen et sa droite d'ajustement affine obtenue par la méthode des moindres carrés.

```
import matplotlib.pyplot as plt
############### LE NUAGE DE POINTS
# Saisir la liste des X et la liste des Y
X=[1.9, 2.4, 1.5, 0.9, 2.3, 1.7]
Y=[190, 250, 170, 150, 210, 180]
….
```

Le graphique attendu :



NB : On rappelle que la droite d'ajustement affine obtenue par la méthode des moindres carrés a pour équation $y=ax+b$ où $a=\frac{cov(x,y)}{V(x)}$ et b peut se calculer en prenant en compte que le point moyen du nuage appartient à cette droite.
Vous trouverez ce qu'il vous est nécessaire de connaître sur Matplotlib ici par exemple (http://apprendre-python.com/page-creer-graphiques-scientifiques-python-apprendre).

Mini-projet 3 : combinaison d'opérations sur les chiffres

Présentation globale

On a droit aux opérations +, - ,*, /.
On se donne une succession de 8 opérations, que l'on appelle « Combinaison ».
On effectue ici ces opérations de gauche à droite sur les entiers de 1 à 9 rangés par ordre croissant.
Exemple : opérations : +,-,+,-,+,-,+,*
Suite d'opérations : 1+2=3 ; 3-3=0 ; 0+4=4 ;4-5=-1 ;-1+6=5 ; 5-7=-2;-2+8=6;6*9=54.
Résultat : 54

- On se demande si 2018 est un résultat possible.
- Comme le prof ne va pas tarder à poser la question, on se demande directement quels sont tous les résultats possibles.

Conventions de programmation communes :
Les opérations sont représentées par le caractère associé et les combinaisons sont stockées dans des listes de chaines de caractères de longueur 1. La combinaison de l'exemple est stockée sous la forme : ['+','-','+','-','+','-','+','*'].

Partie A : Décomposition en sous-tâches :

1. Proposer une fonction EvalCombinaison(C) qui évalue le résultat d'une combinaison C entrée en paramètre. La tester sur différentes combinaisons.
2. Proposer une fonction genereCombinaisons() qui crée l'ensemble des combinaisons possibles et les renvoie sous forme d'une liste de combinaisons. Vérifier qu'il y en a le bon nombre.
3. Proposer une fonction qui, à partir de la liste précédente, renvoie la liste de tous les résultats.
4. 2018 est-il un résultat possible ? Classer ces résultats par ordre croissant (python propose des fonctions pour cela…).

Partie B : Une nouvelle opération

Aux opérations de la partie A, on ajoute l'opération 'a' telle que 3a4 vaut le nombre 34.
'a' est prioritaire sur les autres opérations :
Le résultat de la combinaison ['a','a','a','+','a','a','a','a'] vaut donc: 1234+56789 = 58023
Modifier EvalComb afin de prendre en compte cette nouveauté.
Répondre aux mêmes questions qu'en A.

Partie C : Des priorités plus réalistes

On applique les règles de priorités opératoires usuelles pour +,-,*,/ ; a est prioritaire sur tout.
Par exemple : ['a','-','a','*','a','+','*','a'] vaut : 12-34*56+7*89=-1269.
Répondre aux mêmes questions qu'en A.

Partie D : Une paire de parenthèses supplémentaire

On peut ajouter à la combinaison une paire de parenthèses qui priorisent le calcul selon les règles usuelles. Par exemple ['(','a','-','a',')','*','a','+','*','a'] vaut : (12-34)*56+7*89=-609.
On se pose les mêmes questions qu'au A.

Partie E : Des paires de parenthèses

On ajoute autant de parenthèses que l'on veut (on reste cohérent pour que cela ait un sens mathématique) et on se pose les mêmes questions qu'au A.