Bilan du DS de maths discrètes

8 Frimaire an 224 de la Révolution

Quelques éléments chiffrés d'abord : j'ai proposé des défis optionnels à rendre sur madoc. 26 sur 114 m'en ont rendus au moins 1 dont 17 dans le groupe 4 et un seul dans le groupe 2 : c'est pitoyable.
Je pourrais m'arrêter sur ces simples données qui parlent d'elles-mêmes.
La moyenne globale est de 8.5 mais de grosses disparités : Gr1 : 8, Gr2 :6.5, Gr3 :9, Gr4 : 10.6 sachant qu'il y a 5 étudiants qui ont 20 et 20 est la note normale ! Ce n'est pas une blague : cela veut dire qu'on a travaillé et cherché à comprendre, point n'est besoin d'être un génie, ni même de faire le DS en entier.
Certains ont travaillé mais n'ont pas encore trouvé le bon rythme et rattrappé leurs lacunes : cela va venir.
Pour une grande majorité ($114 - 26 = 88$), il n'y a pas de travail et donc vous allez passer à côté de grands pans de la programmation.
Maintenant le DS qui a été si peu réussi...

1. (a) Soit $B = \{x, y\}$. Déterminez :

$$A_1 = B \otimes B^2 \qquad A_2 = B^2 \otimes B \qquad A_3 = \left(B^2 \setminus \{(y,y)\}\right) \otimes B \otimes \{12\} \qquad A_4 = \mathcal{P}\left(B^2 \setminus \{(y,y)\}\right)$$

   On rappelle que $\otimes$ symbolise le produit cartésien, $\setminus$ la différence des ensembles et $\mathcal{P}(E)$ l'ensemble des parties de E.

   (b) Est-ce que le produit cartésien est associatif ? Justifiez votre réponse.

   (c) Est-ce que le produit cartésien est commutatif ? Justifiez votre réponse.

   S'il y a une notion basique, c'est celle de produit cartésien, travaillée également avec M. Attiogbé et qui est fondamentale par exemple en bases de données.

   Il faut avoir un œil d'informaticien : quels sont les types de mes objets ? $B \otimes B^2$ est un ensemble de couples. Dans ce contexte, B^2 représente $B \otimes B$ (et dans le contexte de l'exercie 3, g^2 désignera $g \circ g$).

   Un ensemble se note avec des $\{\}$ alors que les couples se notent avec des $()$ ou des $\langle\rangle$. Un compilateur ou un interpréteur doit recevoir les bonnes notations !

   $\left(B^2 \setminus \{(y,y)\}\right) \otimes B \otimes \{12\}$ a deux symboles $\otimes$ et il n'y a pas de parenthèses : on aura donc un ensemble de triplets donc par exemple $((x,y),x,12)$ et non pas $(((x,y),x),12)$: si vous ne faites pas la différence, vous aurez du mal par la suite en Info.

   Quant aux termes *associatif* et *commutatif*, il est déplorable de mélanger encore ces notions sachant de plus que

   VOUS AVIEZ DROIT À TOUT DOCUMENT MANUSCRIT !!!

   C'est encore plus fou de penser que certains ont rempli leur devoir de fadaises avec pourtant cette opportunité. Mais pour utiliser des notes, il faut en avoir prises...

   Beaucoup affirment : c'est associtaif ! C'est commutatif ! sans démonstration...Dans votre futur métier, on peut espérer que vous chercherez à développer des applications sans bug, et elles ne seront pas sans bug juste parce que vous l'affirmez...

2. Donner la représentation sagittale (des patates avec des flèches) d'une fonction totale de l'ensemble fini E dans l'ensemble fini F.

(a) non injective, non surjective ;
(b) non injective, surjective ;
(c) injective, non surjective ;
(d) bijective.

Encore une notion travaillée également avec M. Attiogbé. Combien ne savent même pas ce qu'est une fonction alors que c'est l'élément de base de la programmation.

Beaucoup ont fait partir deux flèches d'un même élément du domaine. La programmation a le plus souvent besoin de fonctions pures : un même argument en entrée donnera toujours la même sortie. Il faut savoir également traquer les impures mais pour ça il faut savoir quoi chercher.

3. Soit g la fonction définie par $g(t) = \frac{t+1}{t-1}$.
   (a) Quel est le domaine de définition $\mathcal{D}_g$ de g ?
   (b) Déterminez $g^2 = g \circ g$ puis $g^3 = g \circ g \circ g$.
   (c) Déduisez-en g^n pour tout entier naturel n en justifiant votre réponse.

Une poignée a mené à bien les calculs du a). Pour la plupart des touristes de l'informatique, je vois un carré : c'est l'itération de la multiplication comme en sixième. Il est temps de vous rendre compte que vous avez quitté votre village et qu'en tant qu'informaticien vous allez créer des mondes : l'un des principes clés à comprendre est que les notations que vous allez utiliser sont en nombre limité et donc qu'elles vont dépendre du contexte dans lequel elles seront employées (dans quelle classe ? Pour quelle objet ?...)

C'est sûr que petit vous avez vu l'exposant pour l'itération de la multiplication des réels mais maintenant, vous avez découvert qu'il y avait une multitude de lois et que cet exposant peut désigner l'itération de n'importe laquelle de ces lois....

Ensuite, plus prosaïquement, il y a tous ceux qui ne savent pas additionner deux fractions.

4. Déterminez la valeur de vérité (en justifiant votre réponse) de la proposition suivante :

$$n \in \mathbb{N} \Longrightarrow n^2 - n + 41 \text{ est premier}$$

Alors là, il y en a peut-être 5 qui ont répondu correctement à cette question. Pour beaucoup, effectuer une démonstration c'est écrire des choses absurdes mais en les relayant par des *donc*, des *car*, des $\Rightarrow$ et ça suffit pour transformer des bêtises en « démonstration ».

Par exemple 41 est premier, $n^2 - n$ est pair « donc » $n^2 - n + 41$ est premier ! ?

Vous allez développer des applications et chercher à démontrer que votre application programme ce qu'il faut ou à défaut définir une batterie de tests qui permettra d'être un peu rassuré.

La moindre des choses ici est d'au moins tester vos affirmations loufoques : un pair plus un premier, c'est premier ? $2 + 5$, ouais, $4 + 5$, ah ben non.

Tout nombre impair est premier ? ? ! ! (certains l'ont affirmé...)

Bref, cinq ont quand même remarqué que $41^2 - 41 + 41$ ça fait 41^2 qui n'est évidemment pas premier et donc on a un bug.

5. (a) Démontrez par RÉCURRENCE que $\sum_{i=1}^{i=n} i = \frac{n(n+1)}{2}$.
   (b) Proposez deux fonctions Python `plouche_lent` et `plouche_rapide` qui prennent un entier n et une fonction f en arguments et renvoient $\sum_{i=0}^{i=n} f(i!)$ (sans bien sûr utiliser la fonction Python **`factorial`**).
   L'une effectuera beaucoup moins de calculs (multiplication ou addition) que l'autre.
   (c) Donnez une estimation du nombre d'additions et de multiplications effectuées pour chacune des deux fonctions proposées en fonction de n.

Cette démonstration par récurrence a été faite de nombreuses fois et pourtant... Toute boucle ou fonction récursive est issue d'un raisonnement par récurrence : il faut maîtriser ce principe de base de la programmation !

Ensuite on passe à la programmation proprement dite avec la construction de fonctions qui calculent $\sum_{i=0}^{i=n} f(i!)$. Alors là, c'est vraiment le grand n'importe quoi ! On pourrait penser que là, des informaticiens

vont pouvoir s'exprimer : écrire une petite bouclette, de la gnognotte pour nous les informaticiens. Et pourtant...Que d'absurdités.

On élimine d'abord ceux qui semblent découvrir Python le jour du DS ! Les années précédentes, les excuses du style « ah oui mais c'est du Haskell, c'est trop dur, on comprend pas ». Bon, cette année on travaille en python, le langage utilisé universellement pour initier les débutants, du collège comme de l'université. Et ça ne change rien ! Car quand on ne fait rien quand on ne programme pas, quand on ne réfléchit pas, Haskell, Python ou C, peu importe : on est incapable d'aligner deux commandes correctes.

Un Σ, c'est une boucle donc il n'y a pas trop de mystère pour l'écrire. C'est la plus simple des boucles. Et là, que d'horreurs, de bêtises, d'absurdités j'ai été obligé de lire. Mais bon sang ! Bougez-vous ! Programmez au moins ! Vous avez choisi l'informatique alors allez-y ! Et ben non...Quand la syntaxe est environ correcte, c'est pour écrire au mieux quelque chose qui ressemble tantôt à la somme des entiers de 1 à n ou au calcul de $n!$ ou bien d'autres choses encore plus fausses.

Ah c'était dur de passer de $\sum_{i=0}^{i=n} f(i!)$ à

```
S = 0
for i in range(n + 1):
    S += f( fact(i) )
return S
```

avec par exemple :

```
def fact(n) :
    if n == 0 :
        return 1
    return n * fact(n - 1)
```

ou au moins deux autres versions, toutes écrites et vues en cours/TD/TP...

Ah mais il fallait comprendre ce qu'était f, ce que c'était que factorielle et ne pas mélanger tous les arguments dans une grande marmite et les ressortir dans n'importe quel ordre.

On peut espérer que vous soyez amenés à programmer des choses plus dures !

Ensuite on se rend compte qu'on fait plein de choses plusieurs fois en calculant plusieurs fois $i!$ donc on passe à

```
fa = 1
S  = fa
for i in range(1, n + 1) :
    fa *= i
    S  += f(fa)
return S
```

6. On définit sur $\{1,2,3\}^2$ la fonction $f : (i,j) \mapsto 2 \times i - j$. Calculez les sommes suivantes :

(a) $\sum_{i=1}^{3}\sum_{j=1}^{3} f(i,j)$ (b) $\sum_{k=1}^{3}\sum_{\ell=1}^{k} f(\ell,k)$ (c) $\sum_{i=1}^{3}\sum_{j\geq i} f(i,j)$ (d) $\sum_{i+j=4} f(i,j)$ (e) $\sum_{1\leq i<j\leq 3} f(i,j)$

Ah, encore des boucles...Trop dur de travailler sur des boucles pour des apprentis informaticiens qui travaillent dessus depuis la seconde...

$\sum_{i=1}^{3}\sum_{j=1}^{3} f(i,j)$ c'est :

```
S = 0
for i in range(1, 4) :
    for j in range(1, 4) :
```

```
            S += 2*i - j
    return S
```

$\sum_{k=1}^{3} \sum_{\ell=1}^{k} f(\ell, k)$ c'est

```
S = 0
for k in range(1, 4) :
    for l in range(1, k +1) :
        S += 2*l - k
return S
```

etc.

Il suffit de remplir un tableau :

i \ j	1	2	3
1	$2\times1-1$	$2\times1-2$	$2\times1-3$
2	$2\times2-1$	$2\times2-2$	$2\times2-3$
3	$2\times3-1$	$2\times3-2$	$2\times3-3$

et de choisir les cases correspondant à la boucle.

7. (a) Rappelez la définition d'un prédicat.

   (b) Proposez une fonction Python com_cond qui prend une liste et un prédicat en arguments et qui renvoie le nombre d'éléments de la liste qui vérifient le prédicat. Vous donnerez également des exemples d'utilisation.

Vous ne pouvez pas dire que nous n'avions pas travaillé cette notion de filtre et de prédicat en amphi...et pourtant, ceux qui ont écrit la très difficile fonction :

```
def com_cond(xs, pred) :
    cpt = 1
    for x in xs :
        if pred(x) :
            cpt += 1
    return cpt
```

se comptent sur les doigts d'une main.

Certains ont proposé :

```
return len([x for x in xs if pred(x)])
```

Pourquoi pas.

Bref, n'importe quel étudiant curieux, désirant devenir informaticien, a suffisamment travaillé en deux mois pour écrire ces petites bouclettes. Deux mois pour ça ! Et en plus avec toutes vos notes à disposition !

Réagissez les enfants !

Heureusement, certains ont déjà commencé à grandir. On peut espérer que vos tristes prestations ne soient que de mauvais souvenirs et que tout le monde atteindra le petit niveau de 20.

Je ne vais pas continuer à passer des nuits et des week-ends à préparer des cours pour ensuite passer des nuits et des week-ends à corriger des centaines de copies vides de gens qui ne font rien !

Je ne vais pas continuer à passer des nuits et des week-ends à préparer des cours pour ensuite passer des nuits et des week-ends à corriger des centaines de copies vides de gens qui ne font rien !

Au boulot ! ! ! !