Sujet n°1

Algorithmique & Structudes de données

Premiers algorithmes itératifs

`Temps de réalisation: 3h`

Le TP n°1 a pour objectif de :

- résoudre quelques *problèmes simples* en développant des *algorithmes*.

Échelle de progression :

● « débutant· e » ● « confirmé· e » ● « avancé· e » ● « expert· e »

Ex.	01	02	03	04	05	06	07	08	09	10	11
	●	●	●	●	●	●	●	●	●	●	●

Pour acquérir un niveau de compétence donné, il faut achever **tous** les exercices de ce niveau et des niveaux inférieurs. L'objectif de chacun· e est d'atteindre au moins le niveau *confirmé·e* (●) à l'issue du TP.

La programmation est une activité délicate et incertaine : pensez dès le début à tester votre code régulièrement, en procédant par petites touches successives, tel un peintre devant sa toile...

Tous les programmes de ce TP seront créés dans le paquet `tp1`.

Table des matières

1 Cahier d'exercices

Dans cette partie, tous les algorithmes qui requièrent une saisie de nombre(s) entier(s) devront réutiliser la fonction `saisir_entier` du module `tp0.util`.

Exercice 1: Disques et cylindres

QUESTION 1. Écrire un programme qui demande à l'utilisateur le rayon d'un disque et affiche en retour son périmètre et sa surface. Le calcul du périmètre et de la surface doit être fait dans des fonctions idoines.

QUESTION 2. Écrire un autre programme qui demande à l'utilisateur le rayon d'un cylindre ainsi que sa hauteur et affiche sa surface et son volume. Outre la définition de deux nouvelles fonctions pour les calculs de la surface et du volume, vous devrez réutiliser les fonctions créées à la question précédente.

Exercice 2: Second degré

Écrire une fonction qui résout une équation du second degré, de signature :

```
def eqn_second_degre(a: float, b: float, c: float) -> None
```

Le résultat sera affiché sur la sortie standard. Le déterminant sera calculé dans une fonction dédiée. Le programme de test affichera les racines des équations suivantes :

$$-x^2 + 5x + 14 = 0 \qquad 3x^2 + 5x + 7 = 0 \qquad x^2 - 6x + 9 = 0 \qquad x^2 - \frac{7}{6}x + \frac{1}{3} = 0$$

Exercice 3: Tu devines mon nombre

Écrire un programme qui génère un nombre au hasard entre 0 et 100 (voir fonction `random.randint`) et le fait deviner à un utilisateur. À chaque réponse de l'utilisateur, le programme indiquera si le nombre proposé est le bon ou s'il est inférieur ou supérieur au nombre à trouver.

Le programme doit itérer tant que la réponse n'a pas été trouvée.

Exercice 4: Je devine ton nombre

Écrire un programme qui demande à un utilisateur de choisir un nombre qu'il doit garder secret, puis essayer de deviner ce nombre. À chaque proposition du programme l'utilisateur doit indiquer s'il s'agit du bon nombre ou s'il est inférieur ou supérieur au nombre à trouver.

Le programme doit itérer tant que la réponse n'a pas été trouvée, ou qu'une erreur de l'utilisateur a été détectée.

Exercice 5: Calcul d'agrégats

Demander à l'utilisateur de saisir des nombres entiers, et afficher à la suite de chaque nombre fourni : le minimum, le maximum et la moyenne des nombres déjà saisis. La saisie s'arrête lorsque l'utilisateur entre une ligne vide.

Remarque: *Pour ce programme, la mémorisation systématique des éléments déjà saisis, par exemple dans une liste, n'est pas autorisée.*

2 Problèmes

Exercice 6: Championnat

Écrire un programme élaborant une saison de championnat comportant n équipes, par exemple de football. Le principe consiste à fixer une équipe « pivot », puis à faire tourner les autres équipes d'une journée à l'autre.

Par exemple, avec 6 équipes, et l'équipe n°6 comme pivot :

```
Jour 1: 1-6   Jour 2: 2-6   Jour 3: 3-6    Jour 4: 4-6    Jour 5: 5-6
        2-5           3-1           4-2            5-3            1-4
        3-4           4-5           5-1            1-2            2-3
```

Lorsque l'on est revenu à la permutation originale, toutes les équipes se sont affrontées une fois. Il est important d'équilibrer les matchs à domicile et à l'extérieur pour chaque équipe, ce qui n'est pas vérifié pour l'équipe pivot dans l'exemple ci-dessus. Enfin, Si le nombre d'équipes est impair, il suffit d'ajouter une « équipe fictive » et de considérer que ses rencontres correspondent à des journées de repos pour l'équipe adverse.

Le résultat escompté est la liste des journées de championnat, avec pour chacune, la liste des rencontres. Vous proposerez un affichage similaire à :

```
Journée N°2:
  équipe 3 reçoit équipe 2
  équipe 1 reçoit équipe 5
  équipe 4 au repos
```

Astuce: *Si vous avez du mal à construire l'algorithme par vous-même (essayez d'abord !), voici sa version mathématique.*

Soit n le nombre d'équipes, j le numéro de la journée et i le numéro du match de la journée ; soit n' le nombre d'équipes à prendre en compte pour le calcul. Si n est pair : $n' = n$, sinon $n' = n + 1$. On aura alors $n' - 1$ jours de championnat et $\frac{n'}{2}$ matchs par jour.

Pour déterminer le $i^{ème}$ match d'une journée, pour chaque jour j :

— L'équipe locale est :
- si $i = 1$: n' si n est pair et 0 si n est impair (pas de match).
- sinon $(i > 1)$: $((j + i - 2) \mod (n' - 1)) + 1$

— L'équipe des visiteurs est :
- $((j - i + n' - 1) \mod (n' - 1)) + 1$

Remarque: *La version ci-dessus n'équilibre pas les déplacements de l'équipe pivot. Trouvez un correctif !*

Exercice 7: Calculatrice

Écrire un programme qui demande la saisie au clavier d'une opération sur des nombres entiers naturels et affiche le résultat du calcul.

Les opérateurs à considérer sont $\{+, *, \%, //\}$. Chaque saisie doit se terminer par le signe '='. La calculatrice acceptera par exemple, les suites de caractères suivantes (sans les guillemets) :

```
"344+15="
"22 // 432 ="
"93451*    0="
"12 %3="
```

et refusera les saisies comme celles-ci :

```
"1+2= "
"1+b_="
" "
"3+15"
"2.4+3="
"-4//3="
"4%(-2)="
```

Afficher un message d'erreur le cas échéant.

Remarque: *La fonction* `str.split()`*, voire la construction d'une expression rationnelle, offrent des moyens pratiques pour analyser la chaîne qui représente l'opération à réaliser. Un examen séquentiel des caractères est également possible.*

3 Pour aller plus loin...

Les exercices de cette section sont issus des archives du portail Project Euler. Il s'agit des problèmes n°14, 19, 54 et 787.

Exercice 8: Suite de Collatz

La suite de Collatz, aperçue au TP0, est définie par :

$$C(n) = \begin{cases} n/2 & \text{si } n \text{ est pair,} \\ 3 \cdot n + 1 & \text{sinon.} \end{cases}$$

La séquence obtenue à partir de $n = 13$ est :

$$13 \hookrightarrow 40 \hookrightarrow 20 \hookrightarrow 10 \hookrightarrow 5 \hookrightarrow 16 \hookrightarrow 8 \hookrightarrow 4 \hookrightarrow 2 \hookrightarrow 1$$

Cette séquence contient 10 termes. Bien que cela n'ait pas été prouvé, la conjecture de Syracuse pose le principe d'une convergence à 1, ou plus exactement à la sous-séquence répétée $(4, 2, 1)$, quel que soit le nombre entier n de départ.

Question 1. Donner le (ou les) nombre(s) $n \leq 10\,000$ qui génère(nt) la plus longue suite jusqu'à 1.

Question 2. Même question, pour $n \leq 10^7$.

Exercice 9: Dimanche

La construction du calendrier grégorien est donnée par la comptine suivante, dont il est facile de vérifier la pertinence :

> 1 Jan 1900 was a Monday.
> Thirty days has September,
> April, June and November.
> All the rest have thirty-one,
> Saving February alone,
> Which has twenty-eight, rain or shine.
> And on leap years, twenty-nine.

Une année bissextile (*leap year*) survient tous les 4 ans, lorsqu'elle est divisible par 4 et à l'exception des centaines, sauf si elles sont elles-mêmes divisibles par 400 !

Combien de dimanches sont tombés un premier jour du mois au vingtième siècle (du 1er janvier 1901 au 31 décembre 2000) ?

Exercice 10: Poker

Au jeu de poker, une main est composée de cinq cartes, pouvant former l'une des combinaisons suivantes, dans l'ordre croissant de leur puissance :

1. plus forte carte ;
2. une paire : deux cartes de même rang (ou valeur) ;
3. deux paires ;
4. brelan : trois cartes de même rang ;
5. quinte : cinq cartes consécutives ;
6. couleur : cinq cartes de même enseigne (ou couleur) ;
7. full : brelan et paire ;
8. carré : quatre cartes de même rang ;
9. quinte flush : quinte dont les cinq cartes sont de même enseigne ;
10. quinte flush royale : quinte flush incluant un As.

Pour mémoire, les cartes sont, dans l'ordre : 2, 3, 4, 5, 6, 7, 8, 9, 10 (Ten), Valet (Jack), Reine (Queen), Roi (King), et As (Ace). Les enseignes sont Pique (Spade), Coeur (Heart), Carreau (Diamond), Trèfle (Club).

En cas d'égalité de combinaison entre deux joueurs, c'est la plus haute carte complétant la main qui l'emporte, et ainsi de suite. Voici cinq exemples de mains à deux joueurs :

Main	Joueur 1	Joueur 2	Vainqueur
1	5H 5C 6S 7S KD paire de 5	2C 3S 8S 8D TD paire de 8	Joueur 2
2	5D 8C 9S JS AC plus haute carte As	2C 5C 7D 8S QH plus haute carte Reine	Joueur 1
3	2D 9C AS AH AC brelan	3D 6D 7D TD QD couleur à Carreau	Joueur 2
4	4D 6S 9H QH QC paire de Reines plus haute carte 9	3D 6D 7H QD QS paire de Reines plus haute carte 7	Joueur 1
5	2H 2D 4C 4D 4S full par les 4	3C 3D 3S 9S 9D full par les 3	Joueur 1

Le fichier `poker.txt` contient cent mains aléatoires de 2 joueurs. Sur chaque ligne sont représentées 10 cartes, les cinq premières correspondent à la main du joueur 1, les cinq dernières celle du joueur 2. On suppose que toutes les mains sont valides. En outre, les cartes de chaque main ne sont pas triées. Enfin, il y a systématiquement une main meilleure que l'autre.

Combien de mains le joueur 1 aura-t'il gagné à l'issue de la partie ?

Exercice 11: Le jeu de Bézout

Imaginons un jeu conçu à partir de deux tas de cailloux. Il se joue à deux, au tour par tour. S'il y a a cailloux dans le premier tas et b dans le second, un tour consiste à retirer $c \geq 0$ cailloux du premier tas et $d \geq 0$ du second, de telle sorte que $ad - bc = \pm 1$. Le vainqueur est le premier joueur qui vide l'un des deux tas.

Une condition nécessaire pour jouer est que les nombres a et b soient premiers entre eux.

On dit que l'état (a, b) du jeu est une « position gagnante » si le joueur suivant, considérant qu'il joue de la meilleure des manières, remporte la partie. Les positions (a, b) et (b, a) sont distinctes.

On désigne $H(N)$ comme le nombre de positions gagnantes, avec $\mathsf{pgcd}(a, b) = 1$, $a > 0$, $b > 0$ et $a + b \leq N$.

QUESTION 1. Retrouver $H(4) = 5$ et $H(100) = 2043$.

QUESTION 2. Calculer $H(10^9)$.

En marge: *L'algorithme d'*EUCLIDE *étendu, au cœur de cet exercice, admet beaucoup d'applications comme par exemple en géométrie algorithmique (trouver un point sur une droite dont on donne l'équation), ou en cryptographie (déterminer une paire de clés dans le protocole RSA).*