Licences Mathématiques et Informatique Semestre 2 — 10 avril 2024

Université d'Artois
Faculté des Sciences Jean Perrin

ALGO2 – Session 1

Examen – 1 heure 30 – Sans Document

Le barème est donné à titre indicatif.

Exercice 1 – 3 points – Spécifiez puis écrivez une fonction `un_mot_un_nombre` qui prend en paramètre une pile de chaînes de caractères. Elle modifie le contenu de cette pile qui doit, au sortir de la fonction, contenir les mêmes éléments, mais en alternant un mot et une chaîne représentant un nombre. S'il y a plus de mots ou plus de nombres, le *'trop-plein'* se retrouvera au sommet de la pile.

Exemple : considérons la pile `pileA`. Le sommet de cette pile est l'élément le plus à droite.

`pileA`	"bonjour"	"46"	"123"	"salut"	"coucou"	"37"	"18"	"13"	"25"	"12"	"hello"

Après l'appel `un_mot_un_nombre(pileA)`, l'état de cette pile devrait être :

`pileA`	"bonjour"	"46"	"salut"	"123"	"coucou"	"37"	"hello"	"18"	"13"	"25"	"12"

Vous considérerez que la classe `Pile` est déjà écrite. Vous ne pouvez utiliser que les méthodes de la classe `Pile` que nous avons étudiées en cours. Vous pouvez utiliser d'autres variables locales de type simple ou créer de nouvelles structures `Pile` ou `File`. Les autres structures de données de groupes d'éléments sont interdites (listes, tuples, dictionnaires, ...).

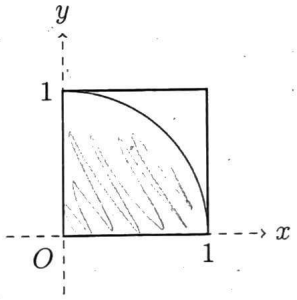
Aide : La méthode `isnumeric` retourne vrai si une chaîne de caractères représente un nombre :

```
>>> ch = "123"
>>> ch.isnumeric()
True
>>> ch = "hello"
>>> ch.isnumeric()
False
```

Exercice 2 – 3 points – Approximation de Pi

Vous allez écrire une fonction qui calcule une approximation de π par une méthode statistique qui vous est décrite.

L'idée est la suivante. Considérons un carré de côté 1 : son aire est donc de $1^2 = 1$. Considérons maintenant le quart de cercle de rayon 1 inscrit dans ce carré : son aire est $1/4$ de l'aire du cercle complet de rayon 1, et donc l'aire du quart de cercle est $\pi/4$.



Si on tire aléatoirement un couple de coordonnées x, y dans l'intervalle $[0; 1]$, la probabilité qu'il corresponde à un point à l'intérieur du quart de cercle est *aire du quart de cercle / aire du carré* soit $\pi/4$.

Si maintenant on tire aléatoirement un grand nombre de points dont les coordonnées sont dans l'intervalle $[0; 1]$, on peut regarder pour chaque point s'il est dans le quart de cercle ou non (sa distance au point $(0, 0)$ est inférieure à 1). Il suffit ensuite de calculer $4\times$*(nombre de points dans le quart de cercle / nombre de points tirés)* pour obtenir une approximation de π.

On appelle méthode de Monte-Carlo la famille des algorithmes basés sur un grand nombre d'expériences aléatoires pour calculer une approximation (et donc qui utilisent des techniques probabilistes).

Spécifier puis écrire une fonction `approx_pi(n)` dont le paramètre `n` est un entier qui indique un nombre de points à choisir aléatoirement, et qui affiche, à chaque tir, la nouvelle approximation de π calculée.

Exemple d'affichage :

```
>>> approx_pi(1500000)
...     ## vous n'avez pas tous les affichages ...
Tir : 1499997 - Approximation de Pi : 3.1418649503965677
Tir : 1499998 - Approximation de Pi : 3.1418655224873633
Tir : 1499999 - Approximation de Pi : 3.1418660945773964
Tir : 1500000 - Approximation de Pi : 3.1418666666666666
```

Aides :

1. la fonction `random()` du module `random` retourne un flottant dans l'intervalle $[0; 1]$
2. la fonction `sqrt(x)` du module `math` retourne la racine carrée de `x` (pour $x \geq 0$).

Exercice 3 – 14 points – Une variante du Sudoku

On s'intéresse ici à une variante du jeu de Sudoku : le joueur doit remplir une grille carrée de dimension `dim`×`dim` de tous les nombres de 1 à `dim` de telle manière que chaque ligne et chaque colonne contiennent tous ces nombres.

Pour les questions suivantes, lorsque vous en avez besoin, vous pouvez toujours utiliser une fonction définie dans une question précédente, même si vous n'avez pas su l'écrire.

```
   0 1 2 3
  +-+-+-+-+
0 |1|2|3|4|
  +-+-+-+-+
1 |2|4|1|3|
  +-+-+-+-+
2 |3|1|4|2|
  +-+-+-+-+
3 |4|3|2|1|
  +-+-+-+-+
```

Un exemple d'une grille complétée

Q 1 . – 1.5 points – Lorsqu'un jeu de Sudoku démarre, il ne contient que quelques cases initialisées. Le jeu consiste à compléter les cases vides.

Écrivez le constructeur de la classe `Sudoku`. Il prend en paramètre la dimension de la grille et une liste de triplets `(lig, col, valeur)`. Le constructeur initialise une matrice carrée d'entiers de la dimension fournie en paramètre. Toutes les cases sont initialisées à 0 (qui sera la marque d'une case vide), à l'exclusion des cases indiquées par les triplets `(lig, col, valeur)` : la case aux coordonnées `(lig,col)` contiendra la valeur `val`.

Les affichages suivants sont donnés pour illustrer le propos. Il ne vous est pas demandé d'écrire la méthode `__str__`.

```
>>> jeu = Sudoku(4,[(0,0,1),(0,2,3),
                    (2,0,3),(2,1,1),
                    (2,2,4)])
```

```
>>> print(jeu)
   0 1 2 3
  +-+-+-+-+
0 |1| |3| |
  +-+-+-+-+
1 | | | | |
  +-+-+-+-+
2 |3|1|4| |
  +-+-+-+-+
3 | | | | |
  +-+-+-+-+
```

Q 2 . – 2.5 points – Écrivez une méthode `est_dans_colonne(self,col:int,val :int)` qui indique si `val` est une valeur présente dans la colonne d'indice `col`.

Écrivez une méthode `pose_valeur(self, lig :int, col :int, val :int)` qui pose la valeur `val` aux coordonnées `(lig,col)` si cela ne viole pas les règles du Sudoku. La méthode retourne vrai si la valeur a pu être posée (on peut toujours poser un `0`).