

# ITI 1520 Automne 2019 - Devoir 4

Date de remise: **le 17 Novembre, avant 22h00**

## Instructions :

**Vous devez faire ce travail en groupe de deux étudiant(e)s. Indiquez dans les commentaires les noms des deux étudiant(e)s. Un(e) des deux seulement doit soumettre le devoir dans le campus virtuel, mais il est important d'indiquer le nom du (de la) deuxième étudiant(e) dans les commentaires pour qu'il (elle) prend lui (elle) aussi la note.**

Répondez à la question 1 dans le fichier D4Q1.py, et à la question 2 dans le fichier D4Q2.py. Pour l'exercice 3, il faut compléter les fichiers jeuSudoku.py et D4Q3.py (ils sont fournis). Mettez tous les fichiers dans un répertoire compressé d4\_numEtudiant.zip pour soumission dans le campus virtuel.

L'objectif de ce devoir c'est de pratiquer les notions que nous avons vues jusqu'ici, particulièrement les dictionnaires et les listes 2D. Ajouter des **commentaires** dans chaque programme pour expliquer le but du programme, la fonctionnalité de chaque fonction et le type des paramètres et du résultat, et des commentaires pour expliquer votre programme. Des points vous seront enlevés si les commentaires sont déficients.

## **Question 1 (6 points)**

Pour cet exercice, refaire la question 4 de Devoir1 en utilisant des dictionnaires. Vous faites la gestion d'un magasin qui vend des articles dans le tableau ci-dessous au prix et quantités suivants:

Produits	Prix	Quantité
bureau	75.9\$	9
chaise	50.9\$	25
imprimante	32.5\$	46
scanneur	28.0\$	17

Utiliser deux variables globales dans le programme principal, un pour l'inventaire (les quantités disponibles), et un pour les prix:

```
q = {"bureau":9, "chaise":25, "imprimante":46, "scanneur":17}
```

```
p = {"bureau":75.9, "chaise":50.9, "imprimante":32.5, "scanneur":28.0}
```

- a) Écrire une fonction Python *calculPrix(article, quantite)*, qui prend en entrée le nom de l'article et la quantité demandée. L'article est de type *str* (une chaîne de caractères) et la quantité est de type *int* (un nombre entier). Cette fonction retourne le prix de type *float* (un nombre réel). Pour cette fonction, on suppose que l'article et la quantité choisie sont déjà valides. Ajouter le contrat de type dans les commentaires docstring pour cette fonction : (str, int) -> float. Tester la fonction.

**Exemple:**

```
>>> calculPrix("bureau", 1)
75.9
```

# ITI 1520 Automne 2019 - Devoir 4

- b) Écrire une fonction Python *calculTotal()* qui prend en entrée 6 paramètres  
*calculTotal(article1, quantite1, article2, quantite2, article3, quantite3)*

Elle appelle la fonction *calculPrix()* autant de fois que nécessaire pour obtenir les prix des 3 articles selon les quantités. *calculTotal()* s'occupe ensuite du calcul du prix total de la commande incluant les trois articles. La fonction *calculTotal()* n'est pas autorisée à utiliser les prix directement. Les prix ne peuvent pas être utilisés que par la fonction *calculPrix()*.

La fonction *calculTotal()* retourne le prix total de la commande.

Pour cette fonction, on suppose que l'article et la quantité choisie sont déjà valides. Ajouter le contrat de type dans les commentaires docstring pour cette fonction. Tester la fonction.

**Exemple:**

```
>>> calculTotal("chaise",2,"bureau",1,"scanneur",1)
205.7
```

- c) Écrire la fonction *validerCommande(article1, quantite1, article2, quantite2, article3, quantite3)* qui prend en entrée 6 variables (les nom des articles avec les quantités associés). Elle retourne une variable de type *bool* (booléen) qui sera *True* si la commande est valide et *False* sinon. Une commande est valide si tous les articles existent dans le magasin et toutes les quantités sont disponibles. Si la commande est valide pour tous les trois articles, actualisez les quantités disponibles dans l'inventaire (le dictionnaire q).  
Ajoutez le contrat de type dans les commentaires docstring pour cette fonction. Tester la fonction.

**Exemple:**

```
>>> validerCommande("bureau",15,"imprimante",2,"scanneur",3)
False
>>> validerCommande("bureau",1,"imprimante",2,"scanneur",3)
True
>>> validerCommande("bureau",1,"souris",2,"scanneur",3)
False
```

- d) Écrire le programme principal qui demande au client d'entrer la liste de trois articles avec les quantités associés, et utilise les sous-algorithmes déjà définis pour retourner le prix total de la commande. Après l'entrée de chaque article, le programme appelle la fonction *validerCommande()* afin de vérifier la disponibilité des articles demandés. Si la commande contient des articles qui ne sont pas valides, la commande sera annulée (dans ce cas pas besoin de faire le calcul de prix total). Après la vérification de tous les articles, le programme appelle les fonctions implémentées en a) et b) afin de calculer et afficher le prix total de la commande du client.  
Des messages appropriés doivent être affichés (voir les exemples d'exécution ci-dessous). Affichez aussi les quantités encore disponibles dans l'inventaire (le dictionnaire q).

**Exemple d'exécution 1:**

```
Entrez le premier article: chaise
Entrez la quantité de votre 1ere article: 1
Entrez le deuxième article: bureau
```

# ITI 1520 Automne 2019 - Devoir 4

Entrez la quantité de votre 2eme article: 1  
Entrez le troisième article: trombone  
Entrez la quantité de votre 3eme article: 3

Votre commande est annulée. SVP, vérifier les articles ou les quantités.

Les quantités disponibles après l'achat sont:  
{'scanneur': 17, 'chaise': 25, 'bureau': 9, 'imprimante': 46}

## Exemple d'exécution 2:

Entrez le premier article: chaise  
Entrez la quantité de votre 1ere article: 4  
Entrez le deuxième article: bureau  
Entrez la quantité de votre 2eme article: 18  
Entrez le troisième article: scanneur  
Entrez la quantité de votre 3eme article: 3

Votre commande est annulée. SVP, vérifier les articles ou les quantités.

Les quantités disponibles après l'achat sont:  
{'scanneur': 17, 'imprimante': 46, 'bureau': 9, 'chaise': 25}

## Exemple d'exécution 3:

Entrez le premier article: chaise  
Entrez la quantité de votre 1ere article: 2  
Entrez le deuxième article: bureau  
Entrez la quantité de votre 2eme article: 1  
Entrez le troisième article: scanneur  
Entrez la quantité de votre 3eme article: 1

Le prix total de votre commande est : 205.7\$. Merci et à la prochaine.

Les quantités disponibles après l'achat sont:  
{'imprimante': 46, 'chaise': 23, 'bureau': 8, 'scanneur': 16}

## Question 2 (2 points)

Écrire une fonction en python modifierMat(matrice) qui permet de modifier seulement les nombres paires par leurs racine carrée. Cette fonction prend en paramètre une matrice. Une matrice est une liste à deux dimensions. Tester la fonction modifierMat.

### Exemple :

```
>>> matrice = [[5, 3, 8], [7, 4, 6], [1, 9, 2], [8, 7, 1], [3, 2, 9], [4, 6, 5]]
>>> modifierMat(matrice)
>>> matrice
[[5, 3, 2.8284271247461903], [7, 2.0, 2.449489742783178], [1, 9, 1.4142135623730951], [2.8284271247461903, 7, 1], [3, 1.4142135623730951, 9], [2.0, 2.449489742783178, 5]]
```

# ITI 1520 Automne 2019 - Devoir 4

## Question 3 (12 points)

On va programmer le jeu Sudoku (<https://fr.wikipedia.org/wiki/Sudoku>). Vous devez donc compléter le programme pour ce jeu. Une partie de programme vous est donnée dans deux fichiers: *D4Q3.py* et *jeuSudoku.py* (que vous devez compléter). Ne modifier pas les prototypes des fonctions. Vous devez implémenter les fonctions pour ce jeu. Un exemple d'exécution du jeu est donné à la fin de ce document.

Le programme principal contrôle le jeu (il est déjà donné).

Au début un menu de 3 choix est affiché à l'utilisateur. Ensuite, il demande à l'utilisateur d'entrée son choix. On suppose que l'utilisateur va entrer exactement le bon choix (un choix entre 1 et 3) donc la validité du choix n'est pas vérifiée.

Le choix 1 : il réinitialise la grille de Sudoku par la matrice donnée dans le programme principal (dans le fichier D4Q3.py). Par la suite, le jeu commence en suivant les mêmes étapes que le choix 2.

Le choix 2 : le jeu commence. Il affiche la grille à l'écran et puis demande à l'utilisateur d'entrer la case à jouer (il doit entrer le numéro de la ligne, de la colonne et le nombre qu'il veut ajouter). Si la case à jouer est correcte (une case valide), alors il faut afficher le message Bravo !!. Sinon, le message Echec sera affiché. Si la case est déjà remplie ou si la grille de Sudoku est pleine, il affiche le message approprié à l'utilisateur.

Le choix 3 : Le jeu s'arrête. Le message Au revoir ! est affiché.

Afin de bien contrôler le jeu, il faut implémenter les fonctions suivantes :

*AfficherGrille()* : permet d'afficher une grille Sudoku sous format d'une matrice 9x9.

*VérifierLigne()* : permet de vérifier si le nombre à ajouter existe déjà sur la ligne.

*VérifierCol()* : permet de vérifier si le nombre à ajouter existe déjà sur la colonne.

*VérifierSousGrille()* : permet de vérifier si le nombre à ajouter existe déjà sur la sous grille où l'ajout sera fait.

*VérifierCase()* : vérifier si une case est valide c.-à-d. le nombre à ajouter n'existe pas ni sur la ligne, ni sur la colonne, ni sur la sous-grille.

*VérifierGagner()* : vérifier si la grille est complète c.-à-d. il n'y a aucun case nul. Dans ce cas, la fonction retourne True.

*Jouer()* : ajouter une case dans la grille de Sudoku seulement si la case est valide.

Notez que:

- 1) Après chaque choix de menu, on réaffiche la grille après modification (s'il en y a) et le menu avec les options et redemande à l'utilisateur d'entrer de nouveau un entre choix jusqu'à que l'utilisateur choisis l'option 3 ou la grille est complétée.
- 2) La case à ajouter est demandé à l'utilisateur avec `input()`. On ne vérifie pas la validité de ces entrées (on suppose que l'utilisateur il va entrer toujours un nombre entre 1 et 9 et un numéro de ligne et numéro de colonne entre 0 et 8).
- 3) Les fonctions *verifierLigne()*, *verifierCol()*, *verifierSousGrille()* retournent *True* si le nombre n'existe pas sur la ligne ou la colonne ou la sous-grille.

# ITI 1520 Automne 2019 - Devoir 4

## Annexe – exemple jeu

Menu: 1- Commencer un nouveau jeu.  
2- Continuer le jeu.  
3- Quitter le jeu.

SVP entrez votre choix (1, 2 ou 3): 1

5	3	8	6	9	1	0	4	7
7	4	6	5	3	2	8	1	9
1	9	2	7	8	4	3	5	6
8	7	1	2	6	3	4	9	5
3	2	9	4	5	7	1	6	8
4	6	5	9	1	8	7	2	3
6	1	4	3	7	9	5	8	2
9	8	3	1	2	5	6	7	4
2	5	0	8	4	6	9	3	1

Entrez votre choix de case ligne: 0  
Entrez votre choix de case col: 6  
Entrez votre choix de case num: 2  
Bravo!!

5	3	8	6	9	1	<b>2</b>	4	7
7	4	6	5	3	2	8	1	9
1	9	2	7	8	4	3	5	6
8	7	1	2	6	3	4	9	5
3	2	9	4	5	7	1	6	8
4	6	5	9	1	8	7	2	3
6	1	4	3	7	9	5	8	2
9	8	3	1	2	5	6	7	4
2	5	0	8	4	6	9	3	1

Menu: 1- Commencer un nouveau jeu.  
2- Continuer le jeu.  
3- Quitter le jeu.

SVP entrez votre choix (1, 2 ou 3): 2

5	3	8	6	9	1	2	4	7
7	4	6	5	3	2	8	1	9
1	9	2	7	8	4	3	5	6
8	7	1	2	6	3	4	9	5
3	2	9	4	5	7	1	6	8
4	6	5	9	1	8	7	2	3
6	1	4	3	7	9	5	8	2
9	8	3	1	2	5	6	7	4
2	5	0	8	4	6	9	3	1

Entrez votre choix de case ligne: 1  
Entrez votre choix de case col: 2  
Entrez votre choix de case num: 7  
Case déjà remplie !!

Menu: 1- Commencer un nouveau jeu.  
2- Continuer le jeu.  
3- Quitter le jeu.

SVP entrez votre choix (1, 2 ou 3): 2

# ITI 1520 Automne 2019 - Devoir 4

5	3	8	6	9	1	2	4	7
7	4	6	5	3	2	8	1	9
1	9	2	7	8	4	3	5	6
8	7	1	2	6	3	4	9	5
3	2	9	4	5	7	1	6	8
4	6	5	9	1	8	7	2	3
6	1	4	3	7	9	5	8	2
9	8	3	1	2	5	6	7	4
2	5	0	8	4	6	9	3	1

Entrez votre choix de case row: 8

Entrez votre choix de case col: 2

Entrez votre choix de case num: 5

Echec!!

Menu: 1- Commencer un nouveau jeu.

2- Continuer le jeu.

3- Quitter le jeu.

SVP entrez votre choix (1, 2 ou 3): 2

5	3	8	6	9	1	2	4	7
7	4	6	5	3	2	8	1	9
1	9	2	7	8	4	3	5	6
8	7	1	2	6	3	4	9	5
3	2	9	4	5	7	1	6	8
4	6	5	9	1	8	7	2	3
6	1	4	3	7	9	5	8	2
9	8	3	1	2	5	6	7	4
2	5	0	8	4	6	9	3	1

Entrez votre choix de case ligne: 8

Entrez votre choix de case col: 2

Entrez votre choix de case num: 7

Bravo!!

5	3	8	6	9	1	2	4	7
7	4	6	5	3	2	8	1	9
1	9	2	7	8	4	3	5	6
8	7	1	2	6	3	4	9	5
3	2	9	4	5	7	1	6	8
4	6	5	9	1	8	7	2	3
6	1	4	3	7	9	5	8	2
9	8	3	1	2	5	6	7	4
2	5	<b>7</b>	8	4	6	9	3	1

Bravo!! Vous avez gagné

