UNIVERSITE DE TOURS

LICENCE 1 MATHEMATIQUE – PHYSIQUE

Contrôle Continue 1 – Algorithmique avec Python

2023-2024 Semestre 1

Instructions :

- Vous rendrez **un** fichier python intitulé `nom_prenom.py` *(exemple : dupont_marie.py)*.
- Les codes correspondants à chaque question seront mis dans l'ordre de l'exercice et précédés de `# Question X` (où X est le numéro de la question).
- Vous pouvez copiez-collez le contenu du fichier `cc1.py` pour avoir une base de travail. Ce dernier contient quelques lignes de tests déjà écrites ainsi que les données nécessaires pour l'exercice 2.
- Attention : les bouts de code doivent être bien exécutables sous peine de perdre des points.
- Testez vos fonctions avec au moins deux tests. La commande de tests doit apparaitre dans vos codes : il en sera tenu compte dans l'évaluation.

Exercice 1 :

1. Codez une fonction calculant le nombre d'entiers compris entre 1 et `n`, qui sont divisibles par 8 et pas par 6. La fonction demande `n` à l'utilisateur.

2. (a) On considère la fonction f, définie sur $]0;+\infty[$ par $f(x)=\frac{1}{2}\left(x+\frac{2}{x}\right)$
   Définir f en python et testez que f(5) renvoie 2,7.
   (b) Soit la suite suivante : $a_0 > 0,\ \forall n \in \mathbb{N}, a_{n+1} = f(a_n)$
   En créant une fonction, testez l'évolution de cette suite. Vers quoi semble tendre a_n pour $n \to +\infty$?

3. Ecrivez une fonction d'affichage nommée `afficheChaineBin(texte)` qui prend en entrée une chaine de caractère et qui affiche caractère par caractère, sur 1 ligne :
   - Le caractère,
   - Sa représentation décimale puis binaire.

   Reproduisez l'affichage ci-dessous.

   Exemple ici si `texte` est valorisé avec la chaine de caractère : `"Test"`
   ```
   T : 84 - 0b1010100
   e : 101 - 0b1100101
   s : 115 - 0b1110011
   t : 116 - 0b1110100
   ```

4. [Bonus] Ecrivez une fonction `afficheHisto(lst, car)` qui prend en entier une liste de réels positifs ou nuls `lst` ainsi qu'un paramètre optionnel `car` qui est une chaine de caractère. La fonction affiche les valeurs sous la forme d'un histogramme.

   Exemple :
   ```
   lst = [2.5, 5.0, 10.2, 3.4, 6.0]
   afficheHisto(lst)
   ```

   Cela affichera :
   ```
   1 : **
   2 : *****
   3 : **********
   4 : ***
   5 : ******
   ```

   Le caractère utilisé pour représenter les unités est l'astérisque `'*'` par défaut, si le paramètre `car` de la fonction n'est pas valorisé. Le cas contraire, la valeur de `car` sera utilisée.

   Exemple :
   ```
   afficheHisto(lst, "=")
   ```

   Cela affichera :
   ```
   1 : ==
   2 : =====
   3 : ==========
   4 : ===
   ```

```
5 : ======
```

Exercice 2 :

L'observation des relevés de températures permet de mieux comprendre l'évolution du climat et d'anticiper les défis futurs liés au réchauffement climatique. Pour aider dans cette tâche, l'informatique a toute sa place : des supercalculateurs, sorte d'ordinateurs surpuissants sont couramment utilisés pour affiner les prévisions. La courses à cette surpuissance est aussi un moteur de compétition international car les prévisions météorologiques et climatiques brassent un nombre titanesque de données.

A notre échelle, il est possible d'utiliser ce que vous avez appris pour effectuer quelques statistiques élémentaires sur une large série de données. C'est l'objectif de cet exercice.

Attention, les questions sont dépendantes les unes avec les autres.

Voici deux tuples qui vont vous permettre de travailler. Ils se trouvent dans le fichier `cc1.py`.

```python
# Tuple indiquant le nombre de jours dans chaque mois
nbrJoursParMois = (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)

# Tuple contenant les tuples des températures minimales et maximales observées chaque mois.
tParMois = (
    (2, 7.8),
    (4.5, 11.8),
    (3.7, 13.7),
    (4.7, 15.4),
    (8.7, 18.1),
    (14.6, 24.5),
    (15.5, 24.9),
    (14.8, 24.6),
    (14, 24),
    (8.6, 18.3),
    (4.9, 11.3),
    (3.8, 10.3)
)
```

Le tuple `nbrJoursParMois` précise le nombre de jours pour chaque mois d'une année. Exemple : 31 jours pour le mois de janvier (indice 0 du tuple), 28 jours pour le mois de février (indice 1 du tuple), 31 jours pour le mois de mars (indice 3 du tuple), etc.

Le tuple `tParMois` comprend 12 sous-tuples, chacun associé à un mois spécifique. Chaque sous-tuple contient deux valeurs : la température minimale et maximale observées durant ce mois. Ainsi, le premier `(2, 7.8)` à la position 0 dans `tParMois`, indique qu'en janvier, la température minimale observée est de 2°C et la maximale de 7.8°C.

Ces deux tuples sont placés dans l'espace global de votre fichier python.

1. Ecrire une fonction `genereListeTemperature(nbJour, tMin, tMax)` avec 3 paramètres en entrée :
   - `nbJour` : Un entier représentant le nombre de jours pour lesquels générer des températures,
   - `tMin` : Un nombre réel indiquant la température minimale possible,
   - `tMax` : Un nombre réel indiquant la température maximale possible

   La fonction doit retourner une liste de `nbJour` températures aléatoires, chaque température étant un nombre réel choisi aléatoirement entre `tMin` et `tMax`. Chaque température dans la liste doit être arrondie à deux chiffres après la virgule.

   Exemple :

   ```python
   temperatures = genereListeTemperature(30, 10.5, 25.8)
   print(temperatures)
   ```

   Dans cet exemple, `temperatures` sera une liste de 30 températures aléatoires, chacune comprise entre 10.5°C et 25.8°C, et arrondie à deux chiffres après la virgule.

   Remarque : La fonction `randint(min, max)` du module random n'est pas appropriée ici, puisqu'elle retourne un entier. Utilisez à la place la fonction `uniform(min, max)` du module random pour tirer aléatoirement un réel entre `min` et `max`.

2. Ecrire une fonction `genereAnnee()`, sans paramètre, qui retourne une liste complète de températures pour chaque d'une année en utilisant les tuples `nbrJoursParMois` et `tParMois` ainsi que la fonction `genereListeTemperature(nbJour, tMin, tMax)` de la question 1.

   Ainsi, il y aura dans cette liste 31 mesures de températures générées aléatoirement entre 2°C et 7.8°C pour le mois de janvier, 28 mesures de températures générées aléatoirement entre 4.5°C et 11.8°C pour le mois de février, 31 pour mars, etc.

   Dans le corps principal, testez qu'il y ait bien 365 éléments dans la liste en retour, correspondant aux 365 relevés de températures avec le code suivant que vous pourrez placer dans l'espace principal :

```
temperatures_annee = genereAnnee()
# La liste devrait contenir 365 éléments.
assert len(temperatures_annee) == 365
print("Nombre total de jours :", len(temperatures_annee))
print("Températures de l'année :", temperatures_annee)
```

3. Ecrire une fonction `moyennesAnnee()` qui prend en entrée la liste des 365 relevés de températures (la liste en sortie de la question 2). Cette fonction :

   - Réalisera l'affichage suivant :

```
Moyennes :
 Mois 1 (31 jours) : 4.9 °C
 Mois 2 (28 jours) : 8.15 °C
 Mois 3 (31 jours) : 8.7 °C
 Mois 4 (30 jours) : 10.05 °C
 Mois 5 (31 jours) : 13.4 °C
 Mois 6 (30 jours) : 19.55 °C
 Mois 7 (31 jours) : 20.2 °C
 Mois 8 (31 jours) : 19.7 °C
 Mois 9 (30 jours) : 19.0 °C
 Mois 10 (31 jours) : 13.45 °C
 Mois 11 (30 jours) : 8.1 °C
 Mois 12 (31 jours) : 7.05 °C
```

   - Retournera une liste de 12 nombres réels, chaque élément représentant la moyenne des températures d'un mois spécifique, arrondi à deux chiffres après la virgule. Utilisez les informations du tuple `nbrJoursParMois`.

   Exemple de liste retournée :

```
[4.9, 8.15, 8.7, 10.05, 13.4, 19.55, 20.2, 19.7, 19.0, 13.45, 8.1, 7.05]
```

4. (a) Écrivez une fonction `question4()` sans paramètre qui génère deux listes de températures annuelles et leurs moyennes mensuelles. Utilisez les fonctions précédentes.

   (b) Écrivez une fonction `compareAnnees(listeA, listeB)` qui compare les températures mensuelles moyennes de deux années différentes et renvoie un score basé sur la comparaison. Comparez les éléments correspondants des deux listes. Incrémentez le score de +1 si l'élément de `listeA` est supérieur à celui de `listeB`, et de -1 dans le cas contraire. Enfin, retournez le score final après avoir comparé tous les éléments.

   (c) Dans le corps de la fonction `question4()`, comparez les deux listes de moyennes puis, si le score renvoyé par `compareAnnees()` est positif, affichez :

```
Les températures de la liste A sont en moyennes supérieures à celles de la liste B
```

   Sinon affichez :

```
Les températures de la liste A sont en moyennes supérieures à celles de la liste B
```

   N'oubliez pas d'appeler la fonction `question4()` dans le corps principal de votre fichier.