Introduction à la programmation (C5-101115-INFO)

Licence 1 - Année 2022/2023

TP 9 - Dictionnaire et Fonctions

B. Besserer, C.Demko, P. Franco, L. Mascarilla, C. Saint-Jean, E. Zahzah

Exercice 1: *Exo 2 - TP 8 - (45 mins)*

Exercice 2: *Bases Dictionnaire (*bases_dict.py*) - (20 mins)*

1. Créer un dictionnaire *D* à deux clés 'Nom', 'Prénom' dont les valeurs sont vos noms et prénoms.
2. Rajouter la clé 'Age' avec la valeur adéquate.
3. Est ce que votre âge est supérieur à la longueur de la chaine issue de la concaténation de votre 'Nom' et 'Prénom' ?
4. Créer un deuxième dictionnaire *D2* contenant d'autres informations sur vous. Mettez à jour *D* avec *D2* (*update*)
5. La clé 'Age' de la question 2 représente en fait votre âge à l'époque de votre entrée au Lycée.
   Corrigez dans *D* en changeant la clé 'Age' en 'AgeLycée' et mettez la valeur adéquate pour vous.
6. Parcourir et afficher le dictionnaire à l'aide d'un *for*.

Exercice 3: *Petites fonctions (*fonctions.py*) - (45 mins)*

1. Définir la fonction *aire_carre* qui donne l'aire d'un carré dont le coté est passé en paramètre.
2. Définir la fonction *aire_rectangle* qui donne l'aire d'un rectangle.
3. Définir la fonction *aire_carre2* qui utilise *aire_rectangle*.
4. Définir la fonction *aire_rectangle2* qui utilise *aire_carre* lorsque celle est nécessaire :
   — Soit les valeurs des deux cotés sont égales.
   — Soit une seule valeur est donnée (l'autre contient None comme valeur par défaut).
5. Définir la fonction *coords_carre* qui retourne la liste des coordonnées (un tuple) des coins d'un carré[4] ; il est de coté c, son coin inférieur gauche est en $(x0, y0)$.

```
Exemples:
coords_carre(1, (0, 0)) retourne la liste [(0, 0), (1, 0), (1, 1), (0, 1)]
coords_carre(3, (2, 1)) retourne la liste [(2, 1), (5, 1), (5, 4), (2, 4)]
```

4. Sens anti-horaire en partant du bas gauche

Exercice 4: *(Bonus) Suite de l'exercice 2 - TP 8*

1. Terminer l'exercice 2 - TP 8 si cela n'a pas été fait.
2. Si cela n'est pas déjà fait, transformer vos différents algorithmes de test de primalité en fonctions.
3. Test de primalité probabiliste de Fermat :

```
fonction testPrimaliteFermat(N)
     choisir aléatoirement un entier positif a < N
     si a**(N-1) % N == 1   alors
        retourner oui (N est probablement premier)
     sinon
        retourner non (N est composé)
```

   On vous demande d'implémenter cet algorithme sans nécessairement comprendre la partie mathématique.
4. Evaluer en moyenne combien de normes premiers < 1000000 sont "manqués" par cet algorithme.
5. Pour améliorer sa précision, on va tirer k nombres aléatoires a (ligne 1).
   Modifier votre fonction pour rajouter un paramètre *k* qui par défaut vaudra 1.
   La fonction retournera True si $a^{N-1} \equiv 1$ mod N pour les *k* valeurs de a.
6. Dans quelle mesure cette modification a t'elle améliorée la précision du test ?