TD 1 : Tests unitaires & Spécification de fonction

Info1.Algo1

2022-2023 Semestre Impair

Exercice 1 : conditions booléennes

Dans chacun des cas suivants, écrire l'expression booléenne permettant de vérifier la condition décrite pour l'entier `n` :
1) `n` est un entier pair supérieur ou égal à `10`.
2) `n` est un entier non nul divisible par `3`.
3) `n` appartient à l'intervalle `]1;5[`.

Exercices : Tests unitaires

Consignes : Pour chacun des exercices suivants :
- Dresser une liste des jeux de tests : cas particuliers, extrêmes, généraux.
- Écrire la fonction de test associée.

Attention ! On ne demande pas d'écrire la fonction testée

Exercice 2 : produit des éléments

Écrire les **jeux de tests** et la **fonction de test** associés à la fonction `produit_elements` qui accepte en paramètre une liste d'entiers et retourne le produit de ses éléments.
Indication : Le cas de la liste vide est-il pertinent à prendre en compte ? Comment feriez-vous s'il s'agissait de calculer la somme des éléments ?

Exercice 3 : nombre de chiffres

Écrire les **jeux de tests** et la **fonction de test** associés à la fonction `nombre_chiffres` qui accepte en paramètre un entier `n` et retourne le nombre de chiffres (en base 10) de `n`.

Exercice 4 :

Écrire les **jeux de tests** et la **fonction de test** associés à la fonction `nb_valeurs_inferieures` qui accepte en paramètres :
- `liste` : une liste d'entiers
- `valeur` : une valeur entière

La fonction détermine et retourne le nombre de valeurs de `liste` qui sont inférieures ou égales à `valeur`.
Exemple : si `liste` vaut `[-3,2,8,1,7,11,3,4]` et `valeur` vaut `3`, les valeurs inférieures ou égales à `3` sont dans l'ordre d'apparition dans la liste `-3`, `2`, `1` et `3`, soit `4` valeurs. La fonction doit donc retourner `4`.

Exercice 5 :

Écrire les **jeux de tests** et la **fonction de test** associés à la fonction `indices_valeur` qui accepte en paramètres :

- `liste` : une liste d'entiers
- `valeur` : une valeur entière

La fonction détermine et retourne la liste des indices dans `liste` où l'on peut trouver `valeur`.
Exemple : si `liste` vaut `[11,23,67,23,23,83,23,10]` et `valeur` vaut 23, la fonction doit retourner `[1,3,4,6]`.

Exercice 6 :

Écrire les **jeux de tests** et la **fonction de test** associés à la fonction `insertion_ordonnee` qui accepte en paramètres :

- `liste` : une liste d'entiers triée par ordre croissant.
- `valeur` : une valeur entière

La fonction insère valeur dans `liste` de telle façon que la liste reste triée par ordre croissant, et retourne la liste ainsi complétée.
Exemple : si `liste` vaut `[-3,2,7,11]` et `valeur` vaut 9, l'insertion doit avoir lieu à l'indice 3 (l'élément 11 est ainsi décalé vers la droite), et la liste retournée est `[-3,2,7,9,11]`.

Exercices : Spécification de fonction

Attentes : pour les **fonctions auxiliaires** (qui servent justement aux tests de propriétés) aucune pré-condition / post-condition n'est demandée.

Exercice 7 : lecture d'énoncé

Écrire la sous-forme de booléens Python la **pré-condition** et la **post-condition** correspondant aux problèmes suivants :
1) Calculer la puissance a^b des deux entiers positifs a et b.
2) On souhaite écrire la fonction `log2_entier` qui accepte en paramètre un entier `n` strictement positif et retourne l'unique entier `k` postif tel que `2**k<=n<2**(k+1)`
Indication : Préciser toutes les notations utilisées.

Exercice 8 : écriture de fonctions auxiliaires

1) Écrire la fonction auxiliaire `est_croissant` qui accepte en paramètre une liste d'entiers `tab`, vérifie que cette liste est triée par ordre croissant et retourne le booléen correspondant.
2) Écrire la fonction auxiliaire `est_membre` qui accepte en paramètre une liste d'entiers `tab` et un entier `m`, et retourne le booléen indiquant si `m` est un élément de `tab`.
3) Écrire la fonction auxiliaire `est_majorant` qui accepte en paramètre une liste d'entiers `tab` et un entier `m`, et retourne le booléen indiquant si `m` est supérieur ou égal à tous les éléments de `tab`.

Exercice 9 : comprendre une spécification

On donne la spécification de fonction suivante :
Type de l'entrée : `n` est un entier.
Type de la sortie : `a` est un entier.

Pré-condition : `n>=0`
Post-condition : `a>=0 and a**2<=n<(a+1)**2`.
On répondra dans l'ordre qu'on le souhaite aux deux questions suivante :

- Écrire une implémentation de cette spécification.
- Décrire la nature du problème posé, et en déduire un nom pertinent pour la fonction à écrire.

Contrainte : Ne pas utiliser de type `float` qui limiterait le domaine de validité de la fonction.

Exercice 10 : complément sur les ternaires

On souhaite écrire la fonction `nombre_chiffres` qui accepte en paramètre un entier positif ou nul `n` et retourne son nombre `k` de chiffres (en base 10).
1) Écrire la **pré-condition** de cette fonction.
2) a) Écrire la fonction auxiliaire `est_nombre_chiffres(n,k)` qui retourne le booléen indiquant si `k` est le nombre de chiffres (en base 10) de l'entier positif ou nul `n`.
Indications :

- Un nombre entier positif a k chiffres si et seulement s'il appartient à l'intervalle :

$$\left[10^{k-1}; 10^{k}\right[$$

- L'exception à cette propriété est l'entier 0 qui possède un chiffre.

b) En déduire la **post-condition** de la fonction `nombre_chiffres`
3) Pour éviter l'utilisation d'une fonction auxiliaire dans ce cas on peut utiliser une **expression conditionnelle** encore appelée **ternaire**.

- **Syntaxe :** `valeur_vrai if condition else valeur_faux`
- **Sémantique :** si `condition` est évaluée à `True`, alors l'expression vaut `valeur_vrai` sinon l'expression vaut `valeur_faux`.
- **Exemple :** L'expression
  `n//2 if n%2 else 3*n+1`
  vaut 13 si `n` vaut 26, et 22 si `n` vaut 7.

Réécrire la **post-condition** de la fonction `nombre_chiffres` sous forme d'un ternaire.
4) Écrire la fonction `nombre_chiffres`.
Attention !!! Le paramètre `n` ne doit pas être modifié par la fonction `nombre_chiffres`.
5) (Pour aller plus loin) Généraliser dans le cas des nombres relatifs.