NSI Fin de première : Corrigé

Lycée Turgot | 2024-2025

Consignes pour les exercices

Les exercices suivants demandent de recopier et compléter des codes de fonctions.

Les commentaires entourés par "' ou précédés de # n'ont pas besoin d'être recopiés.

Les exemples de l'exercice 1 n'ont pas besoin non plus d'être recopiés.

Exercice 1

Recopier et compléter la fonction suivante qui vérifie si une liste d'entiers `tab` est triée dans l'ordre croissant :

```
def verifie(tab: list) -> bool:
    '''Renvoie True si la liste tab est triée dans l'ordre croissant, False sinon.'''
    for i in range(len(tab) - 1): # Arrêt de la boucle à l'avant dernier élément
        if tab[i] > tab[i + 1]: # compare un élément et son suivant
            return False
    return True

# Exemples :
print(verifie([0, 5, 8, 8, 9]))     # True
print(verifie([8, 12, 4]))          # False
print(verifie([-1, 4]))             # True
print(verifie([]))                 # True
```

Exercice 2

La fonction `tri_insertion` suivante prend en argument une liste `tab` (type `list`) et trie cette liste en utilisant la méthode du tri par insertion.

Remarque : Cette fonction n'a pas de `return`. Elle modifie directement la liste `tab` passée en argument.

```python
def tri_insertion(tab: list):
    '''Trie la liste tab par ordre croissant
    en appliquant l'algorithme de tri par insertion'''
    n = len(tab)
    for i in range(1, n):
        valeur_insertion = tab[i]
        j = i
        while j > 0 and valeur_insertion < tab[j - 1]:
            tab[j] = tab[j - 1]
            j = j - 1
        # fin des décalages, la bonne place est trouvée
        tab[j] = valeur_insertion
```

question bonus : Quelle est la complexité temporelle, dans le pire des cas, de cette fonction ? **Réponse :** La complexité est **O(n²)** dans le pire des cas (liste triée dans l'ordre décroissant).

Exercice 3

Le but de l'exercice est de compléter une fonction qui détermine si une valeur est présente dans une liste de valeurs triées dans l'ordre croissant.

Recopier et compléter la fonction `dichotomie` donnée ci-après :

```python
def dichotomie(tab: list, x: int) -> bool:
    '''applique une recherche dichotomique pour déterminer
    si x (entier) est dans la liste triée tab.
    La fonction renvoie True si tab contient x et False sinon'''
    debut = 0
    fin = len(tab) - 1 # indice du dernier élément
    while debut <= fin:
        m = (debut + fin) // 2 # indice de l'élément médian
        if x == tab[m]:
            return True
        if x > tab[m]:
            debut = m + 1
        else:
            fin = m - 1
    return False
```

Exercice 4

On considère un algorithme glouton pour le rendu de monnaie. Les pièces de monnaie utilisées sont : `[1, 2, 5, 10, 20, 50, 100, 200]`.

Recopier et compléter la fonction `rendu_monnaie` ci-dessous.

```python
pieces = [1, 2, 5, 10, 20, 50, 100, 200]

def rendu_monnaie(somme_due: int, somme_versee: int) -> list:
    '''Renvoie la liste des pièces à rendre pour rendre la monnaie
    lorsqu'on doit rendre somme_versee - somme_due'''
    rendu = [] # initialise une liste vide
    a_rendre = somme_versee - somme_due
    i = len(pieces) - 1
    while a_rendre > 0: # Tant qu'il reste à rendre
        while pieces[i] > a_rendre:
            i = i - 1
        rendu.append(pieces[i])
        a_rendre = a_rendre - pieces[i] # décrementation de la somme à rendre
    return rendu
```

Exercice 5

Un mot palindrome peut se lire de la même façon de gauche à droite ou de droite à gauche : `kayak, radar, et non`.

Recopier et compléter les deux fonctions ci-dessous.

```python
def inverse_chaine(chaine: str) -> str:
    '''Retourne la chaine inversée'''
    resultat = "" # initialisation de la chaîne vide
    for caractere in chaine:
        resultat = caractere + resultat
    return resultat

def est_palindrome(chaine: str) -> bool:
    '''Renvoie un booléen indiquant si la chaine est un palindrome'''
    inverse = inverse_chaine(chaine)
    return chaine == inverse # retourne True si l'égalité est vérifée, False sinon
```