BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°21

DUREE DE L'EPREUVE : 1 heure

Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le candidat doit traiter les 2 exercices.

EXERCICE 1 (4 points)

Programmer la fonction `multiplication`, prenant en paramètres deux nombres entiers `n1` et `n2`, et qui renvoie le produit de ces deux nombres.

Les seules opérations autorisées sont l'addition et la soustraction.

Exemples :

```
>>> multiplication(3,5)
15
>>> multiplication(-4,-8)
32
>>> multiplication(-2,6)
-12
>>> multiplication(-2,0)
0
```

EXERCICE 2 (4 points)

Recopier et compléter sous Python la fonction suivante en respectant la spécification. On ne recopiera pas les commentaires.

```
def dichotomie(tab, x):
    """
        tab : tableau d'entiers trié dans l'ordre croissant
        x : nombre entier
        La fonction renvoie True si tab contient x et False sinon
    """

    debut = 0
    fin = len(tab) - 1
    while debut <= fin:
        m = ...
        if x == tab[m]:
            return ...
        if x > tab[m]:
            debut = m + 1
        else:
             fin = ...
    return ...
```

Exemples :

```
>>> dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],28)
True
>>> dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],27)
False
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 1
    * du sujet 19 des épreuves pratiques NSI 2022
    * du sujet 21 des épreuves pratiques NSI 2022
"""


from doctest import testmod


# version itérative
def multiplication(n1: int, n2: int) -> int:
    """Renvoie le résultat de la multiplication de n1 par n2
    (avec n1 et n2 entiers relatifs).
    Petit défi : fonction programmée uniquement avec
                 les opérateurs + et -.

    Args:
        n1 (int): premier facteur
        n2 (int): deuxième facteur

    Returns:
        int: résultat de n1 × n2

    Tests et exemples:
    >>> multiplication(3,5)
    15
    >>> multiplication(-4,-8)
    32
    >>> multiplication(-2,6)
    -12
    >>> multiplication(-2,0)
    0
    """
    # résultat du produit initialisé à 0
    # car si aucune boucle ne s'exécute (n1 ¿ 0)
    # alors le résultat reste correct
    produit = 0

    # cas de n1 positif:
    #   faire n1 additions successives pour obtenir "n1 fois n2"
    if n1 >= 0:
        for i in range(n1):
            produit = produit + n2

    # cas de n1 négatif:
    #   faire n1 soustractions successives pour obtenir "n1 fois n2"
    else:
        for i in range(-n1):
            produit = produit - n2

    return produit


# version récursive (pour se faire plaisir ;) )
def multiplication_r(n1: int, n2: int) -> int:
    """Renvoie le résultat de la multiplication de n1 par n2
    (avec n1 et n2 entiers relatifs).
    Petit défi : fonction programmée uniquement avec
                 les opérateurs + et -.

    Args:
        n1 (int): premier facteur
        n2 (int): deuxième facteur

    Returns:
        int: résultat de n1 × n2

    Tests et exemples:
    >>> multiplication_r(3,5)
    15
    >>> multiplication_r(-4,-8)
```

```
    32
    >>> multiplication_r(-2,6)
    -12
    >>> multiplication_r(-2,0)
    0
    """
    # l'idée est de remarquer pour n1 POSITIF:
    # ¿ n1 × n2 = n2 + (n1 - 1) × n2
    # ce qui s'écrit de façon fonctionnelle:
    # ¿ multiplication(n1, n2) = n2 + multiplication(n1-1, n2)
    #
    # pour le cas n1 NÉGATIF:
    # ¿ n1 × n2 = (n1 + 1) × n2 - n2
    #
    # cette remarque permet une définition récursive de la multiplication
    #   ¿ avec n1 qui diminue de 1 à chaque appel récursif
    #   ¿ jusqu'à atteindre le cas de base 1

    # cas de bases (1) et (2):
    if n1 == 0 or n2 == 0:
        return 0

    # cas de base (3)
    if n1 == 1:
        return n2

    # cas positif:
    if n1 > 1:
        return n2 + multiplication(n1 - 1, n2)
    # cas négatif
    else:
        return multiplication(n1 + 1, n2) - n2


# tests avec des affichages
print(multiplication(3,5))
print(multiplication(-4,-8))
print(multiplication(-2,6))
print(multiplication(-2,0))


# tests de la fonction avec doctest
testmod()
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 2 du sujet 21 des épreuves pratiques NSI 2022

Remarque:
    * lorsque la valeur cherchée est présente dans le tableau, la
    fonction ne renvoie pas l'indice (ce qui est classique) mais True.
    La fonction indique si l'élément est présent ou pas dans le tableau
"""


def dichotomie(tab, x):
    """
        tab : tableau d¿entiers trié dans l¿ordre croissant
        x : nombre entier
        La fonction renvoie True si tab contient x et False sinon
    """
    # début de la zone à explorer
    debut = 0
    # fin de la zone à explorer
    fin = len(tab) - 1

    # tant que la zone à explorer n'est pas vide
    while debut <= fin:
        # indice médian de la zone à explorer
        m = (fin + debut) // 2

        # la valeur cherchée est égale à la valeur médiane
        # alors c'est trouvé : renvoyer True
        if x == tab[m]:
            return True

        # si présente, la valeur cherchée est après la valeur médiane
        # changement de zone de recherche en modifiant le début
        if x > tab[m]:
            debut = m + 1

        # si présente, la valeur est avant la valeur médiane
        # changement de la zone de recherche en modifiant la fin
        else:
             fin = m - 1

    # si on sort de la boucle c'est que la zone de recherche est
    # désormais vide et donc l'élément cherché n'y est pas
    return False


# tests avec des assertions
assert dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],28) is True
assert dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],27) is False

# tests avec des affichages
print(dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],28))
print(dichotomie([15, 16, 18, 19, 23, 24, 28, 29, 31, 33],27))
```