# BACCALAURÉAT

## SESSION 2025

---

## Épreuve de l'enseignement de spécialité

# NUMÉRIQUE et SCIENCES INFORMATIQUES

## Partie pratique

## Classe Terminale de la voie générale

---

## Sujet n°48

---

DURÉE DE L'ÉPREUVE : 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 (10 points)

Programmer la fonction `recherche`, prenant en paramètre un tableau non vide `tab` (type `list`) d'entiers et un entier `n`, et qui renvoie l'indice de la **dernière** occurrence de l'élément cherché. Si l'élément n'est pas présent, la fonction renvoie `None`.

Exemples

```
>>> recherche([5, 3],1) # renvoie None
>>> recherche([2,4],2)
0
>>> recherche([2,3,5,2,4],2)
3
```

## EXERCICE 2 (10 points)

On souhaite programmer une fonction indiquant le point le plus proche d'un point de départ dans un tableau de points non vide. Les points sont tous à coordonnées entières et sont donnés sous la forme d'un tuple de deux entiers. Le tableau des points à traiter est donc un tableau de tuples.

On rappelle que la distance $d$ entre deux points du plan de coordonnées $(x; y)$ et $(x'; y')$ vérifie la formule :

$$d^2 = (x - x')^2 + (y - y')^2$$

Compléter le code des fonctions `distance_carre` et `point_le_plus_proche` fournies ci-dessous pour qu'elles répondent à leurs spécifications.

```
def distance_carre(point1, point2):
    """ Calcule et renvoie la distance au carre entre
    deux points."""
    return (...)**2 + (...)**2

def point_le_plus_proche(depart, tab):
    """ Renvoie les coordonnées du premier point du tableau tab se
    trouvant à la plus courte distance du point depart."""
    min_point = tab[0]
    min_dist = ...
    for i in range(1, len(tab)):
        if distance_carre(tab[i], depart) < ...:
            min_point = ...
            min_dist = ...
    return min_point
```

Exemples :

```
>>> distance_carre((1, 0), (5, 3))
25
>>> distance_carre((1, 0), (0, 1))
2
>>> point_le_plus_proche((0, 0), [(7, 9), (2, 5), (5, 2)])
(2, 5)
>>> point_le_plus_proche((5, 2), [(7, 9), (2, 5), (5, 2)])
(5, 2)
```