# DS — Python pratique (BTS SIO 1re année, Bloc B3 Cybersécurité)

**La rigueur, la rédaction, le soin et le typage des fonctions** sont évalués sur 2 points.
Si un de ces points n'est pas satisfait, **2 points seront retirés de la note**.

**Toutes les fonctions doivent être testées. Si une fonction n'est pas testée, l'intégralité des points ne sera pas comptabilisée.**

## Rappels autorisés

- `int(...)`, `str(...)`, `input(...)`, `print(...)`
- Concaténation de chaînes: `"bonjour" + " monde"` → `"bonjour monde"`, `"A" + "B"` → `"AB"`, `"B" + "A"` → `"BA"`
- Pour tester qu'un caractère appartient à un ensemble (chiffres, lettres, spéciaux), parcourir la chaîne de référence avec une boucle et comparer les caractères.
- Pour vérifier qu'un caractère est dans une chaîne de caractères (mot clef **in**) : `caractere in chaine` -> `"r" in "renard"` → `True`
- On peut accéder au caractère à l'indice `i` de la chaîne `chaine` avec `chaine[i]`, rappel: les indices commencent à 0. -> `"renard"[0]` → `"r"`, `"renard"[1]` → `"e"`
- Pour obtenir la taille d'une chaîne de caractères, on peut utiliser la fonction `len(chaine)`

## Partie A — Code à compléter (8 pts)

### *Exercice A1 (1 pt)*

**Compléter la fonction `nb_occurrences(texte, lettre)` qui prend une chaîne `texte` et un caractère `lettre`, et renvoie le nombre d'occurrences de `lettre` dans `texte`.**

```
def nb_occurrences(texte: str, lettre: str) -> int:
    compteur = 0
    for c in texte:
        if ...:
            compteur = ...
    return ...
```

### *Exercice A2 (1 pt)*

**Compléter la fonction `contient_majuscule(nom)` qui renvoie `True` si `nom` contient au moins une majuscule (A–Z), `False` sinon.**

```python
def contient_majuscule(nom: str) -> bool:
    maj = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    for c in nom:
        if ...:
            return True
    return ...
```

### *Exercice A3 (1.5 pt)*

**Compléter `contient_separateur(code)` qui renvoie `True` si `code` contient `-` ou `_`.**

```python
def contient_separateur(code: str) -> bool:
    seps = "-_"
    for c in code:
        if ...:
            return True
    return ...
```

### *Exercice A4 (1.5 pt)*

**Compléter `contient_deux_types(code)` qui renvoie `True` si le `code` contient au moins deux catégories différentes parmi: lettre, chiffre, séparateur; sinon `False`, en utilisant les fonctions précédentes (`nb_occurrences` pour une lettre donnée, `contient_separateur`) et une chaîne de l'alphabet.**

```
def contient_deux_types(code: str) -> bool:
    alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
    digits = "0123456789"
    a_lettre = False
    # au moins une lettre ?
    i = 0
    while i < len(alphabet):
        if nb_occurrences(code, alphabet[i]) > ...:
            a_lettre = True
        i = i + 1
    a_chiffre = False
    j = 0
    while j < len(digits):
        if nb_occurrences(code, digits[j]) > ...:
            a_chiffre = True
        j = j + 1
    a_sep = contient_separateur(...)
    if (a_lettre and a_chiffre) or (a_lettre and a_sep) or (a_chiffre and a_sep):
        return True
    return ...
```

***Exercice A5 (1.5 pt)***

**Compléter `identifiant_renard_valide(code)` qui renvoie `True` si:**

- longueur ≥ 6
- caractères autorisés: lettres, chiffres, `-`, `_`
- contient au moins une lettre et au moins un chiffre
  Utiliser les fonctions précédentes.

```
def identifiant_renard_valide(code: str) -> bool:
    if len(code) < ...:
        return ...
    alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
    digits = "0123456789"
    a_lettre = False
    a_chiffre = False
    for c in code:
        if c in alphabet:
            a_lettre = True
        if c in digits:
            a_chiffre = True
    if a_lettre and a_chiffre:
        return ...
    return ...
```

***Exercice A6 (1.5 pt)***

**Compléter `score_renard(code)` qui calcule un score:**

- +2 si longueur ≥ 6
- +2 si `contient_separateur` est vrai
- +2 si `contient_majuscule` est vrai
- +2 si `contient_deux_types` est vrai
- +2 si `nb_occurrences(code, 'R') ≥ 1`

```
def score_renard(code: str) -> int:
    score = 0
    if len(code) >= ...:
        score = score + ...
    if contient_separateur(...):
        score = score + ...
    if contient_majuscule(...):
        score = score + ...
    if contient_deux_types(...):
        score = score + ...
    if nb_occurrences(..., ...) >= ...:
        score = score + ...
    return ...
```

## Partie B — Fonctions à écrire (10 pts)

### *Exercice B1 (2 pt)*

**Écrire une fonction `repeter_hurlement(c, n)` qui renvoie une chaîne contenant le caractère `c` répété `n` fois.**

### *Exercice B2 (2 pt)*

**Écrire une fonction `compter_voyelles(nom)` qui renvoie le nombre de voyelles dans `nom`.**
Utiliser une boucle et comparer chaque caractère à l'ensemble des voyelles `"aeiouyAEIOUY"`.

### *Exercice B3 (2 pt)*

**Écrire une fonction `categorie_age_renard(age)` qui renvoie "Jeune" si `age <= 1`, "Adulte" si `2 <= age <= 6`, sinon "Senior".**
Utiliser des conditions.

### *Exercice B4 (2 pt)*

**Écrire une fonction `identifiant_suivi(nom, annee)` qui renvoie une chaîne sous la forme `RX-ANNEE-NOM`.**
Exemple: `identifiant_suivi("Milo", 2025)` → `RX-2025-Milo`

### *Exercice B5 (2 pt)*

**Écrire une fonction `inverser_chaine(chaine)` qui renvoie `chaine` inversée.**
Utiliser une boucle pour construire la chaîne inversée (ne pas utiliser de slicing).