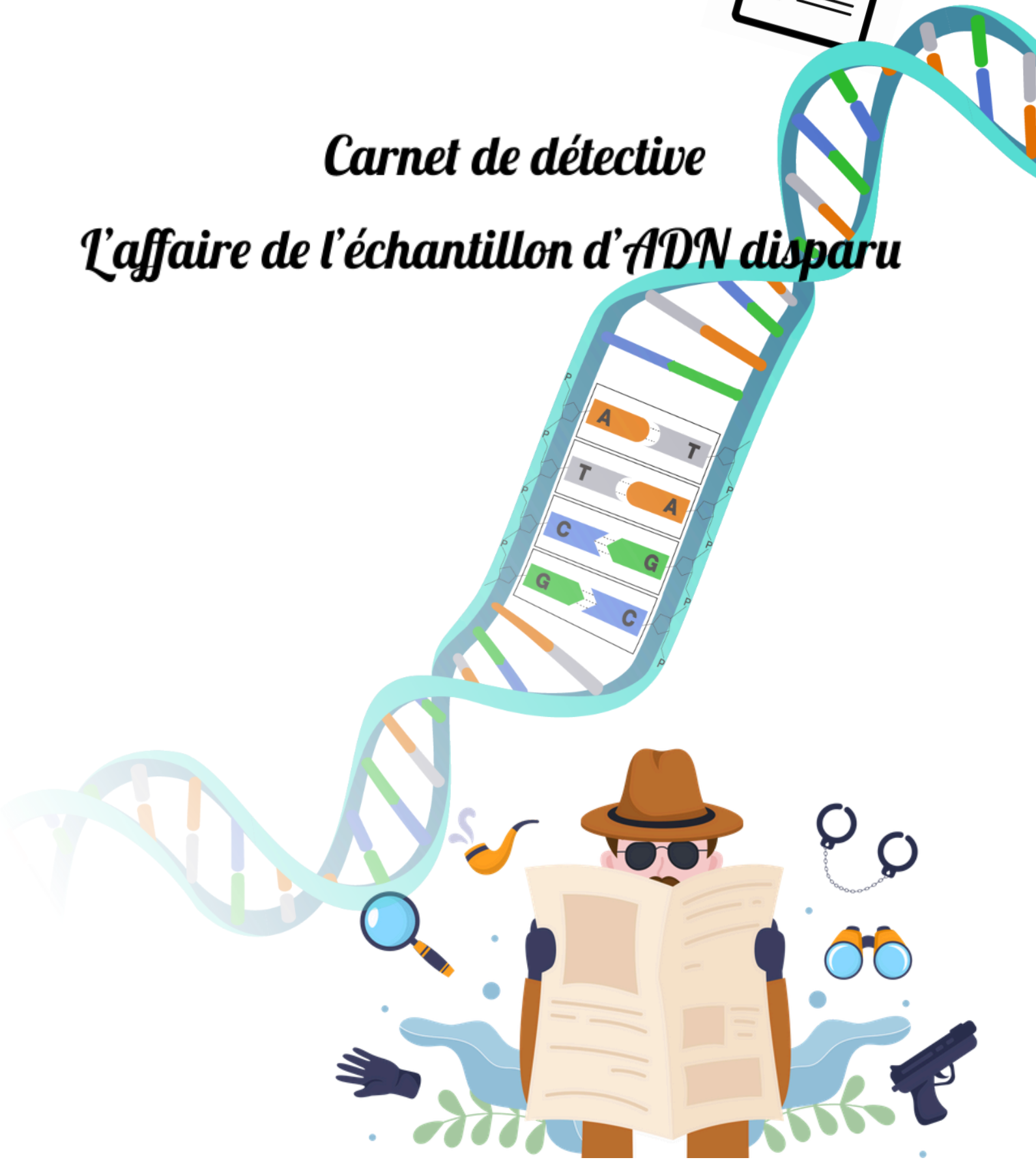
Nom du détective :

Date de l'enquête :

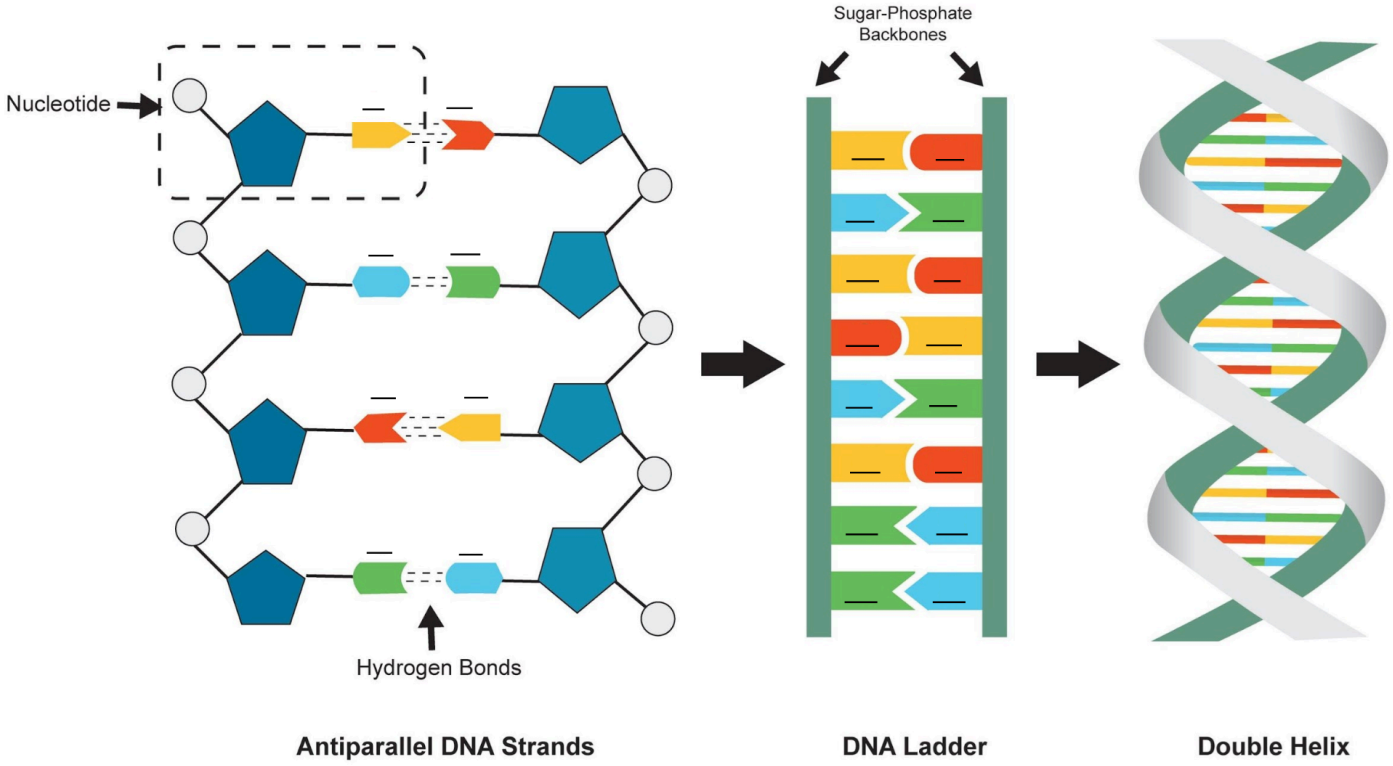
Carnet de détective

L'affaire de l'échantillon d'ADN disparu



Remplissez le schéma avec les bases correspondantes

L’ADN est composé de 4 bases : A, T, C, ou G



Liste des suspects

L’étudiante en recherche :

- A un chien
- Yeux marrons
- Sexe féminin
- Brune

Le technicien :

- A un chat
- Yeux bleus
- Sexe masculin
- Brun

Le visiteur :

- A un chien
- Yeux marrons
- Sexe masculin
- Brun

La responsable des stocks :

- Sans animaux
- Yeux verts
- Sexe féminin
- Rousse

La professeure :

- Yeux bleus
- A un lapin
- Sexe féminin
- Blonde

Ancienne employée :

- Yeux marrons
- A un furet
- Sexe féminin
- Brune

Activité Assemblage :

L'objectif de cette activité est de réussir à assembler nos morceaux d'ADN ensemble pour obtenir une longue séquence, comme dans cet exemple :

Résultat du niveau 1 : (0 erreur)

Résultat du niveau 2 : (4 erreurs min)

Résultat du niveau 3 : (4 erreurs min)

Résultat du niveau 4 :

Comparaison de la séquence obtenue avec le génome de différentes espèces

Pseudo-code :

Quelle espèce a été trouvée ?

Aligner des morceaux d'ADN sur un gène

Pseudo-code :

Quel attribut a été trouvé ?

Aligner des morceaux d'ADN, même s'ils sont coupés en deux

Pseudo-code :

Quel attribut a été trouvé ?

Comparer une séquence de référence avec deux gènes et trouver celui qui a le moins d'erreurs

Pseudo-code :

Quel attribut a été trouvé ?

Reconnaître les erreurs

Dans les activités précédentes, nous avons remarqué que des erreurs s'étaient glissées dans nos séquences. Il existe trois types d'erreurs, saurez-vous les identifier ?

Voici la séquence de référence : AATGCTGCA

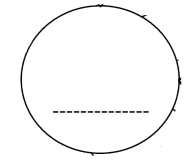
Quelles sont les types d'erreurs pour :

AAGGCTGCA : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

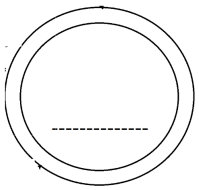
AATGCGCA : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

AATGCATGCA : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

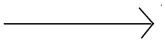
Légende:



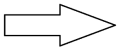
état dans lequel je me situe



état dans lequel je peux me situer à la fin de la lecture



passage d'un état à un autre



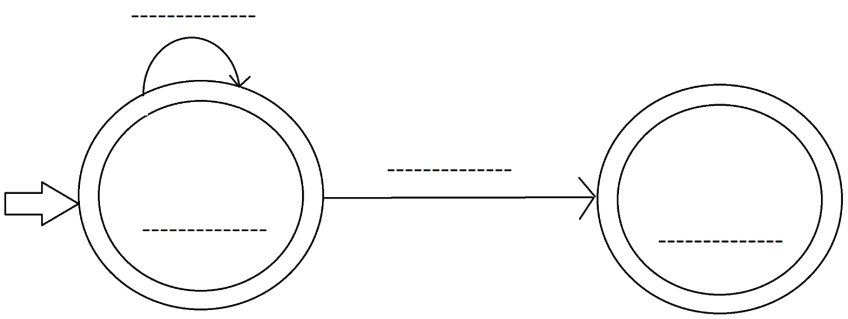
la lecture commence par cet état

E_ref = E_ech — condition que l'élément lu dans la référence soit égale à l'élément lu dans l'échantillon

Objectif : créer un schéma capable d'identifier si la séquence trouvée contient une erreur

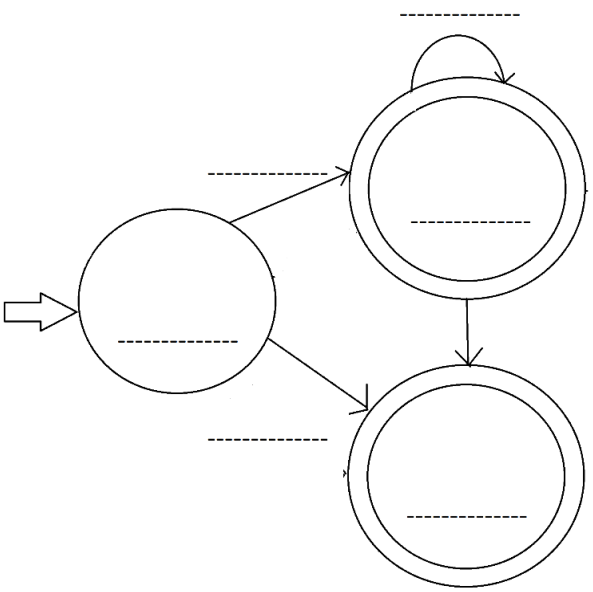
Pour ce faire compléter ce mécanisme de détecteur d'erreur avec les blocs suivants :

Pas d'erreur | E_ref = E_ech | E_ref ≠ E_ech | Erreur



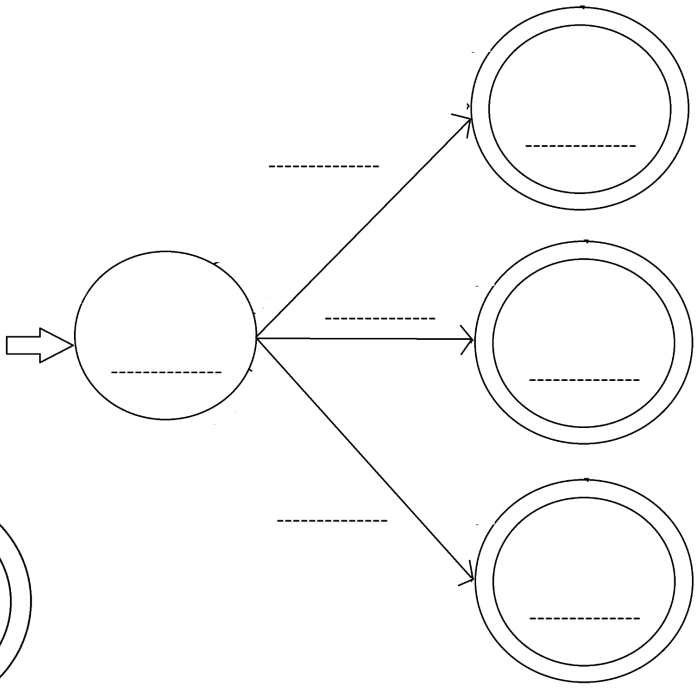
Objectif : créer un schéma capable d'identifier la position de l' erreur

E_ref = E_ech | E_ref ≠ E_ech | Position = 0 | Poisition = Position +1



Objectif : créer un schéma capable d'identifier le type d'erreur

E_X+1_ref = E_X+1_ech | E_X+1_ref = E_X_ech | E_X+1_ech = E_X_ref | substitution | insertion | délétion | Erreur en position X



A partir de votre dernier mécanisme, essayez d'identifier l'erreur de la séquence suivante :

réf : ACGTTA

échantillon : ACTTA

Que se passe-t-il ? _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _

ref : ACTGTA

échantillon : ACTTA

Que se passe-t-il ? _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _

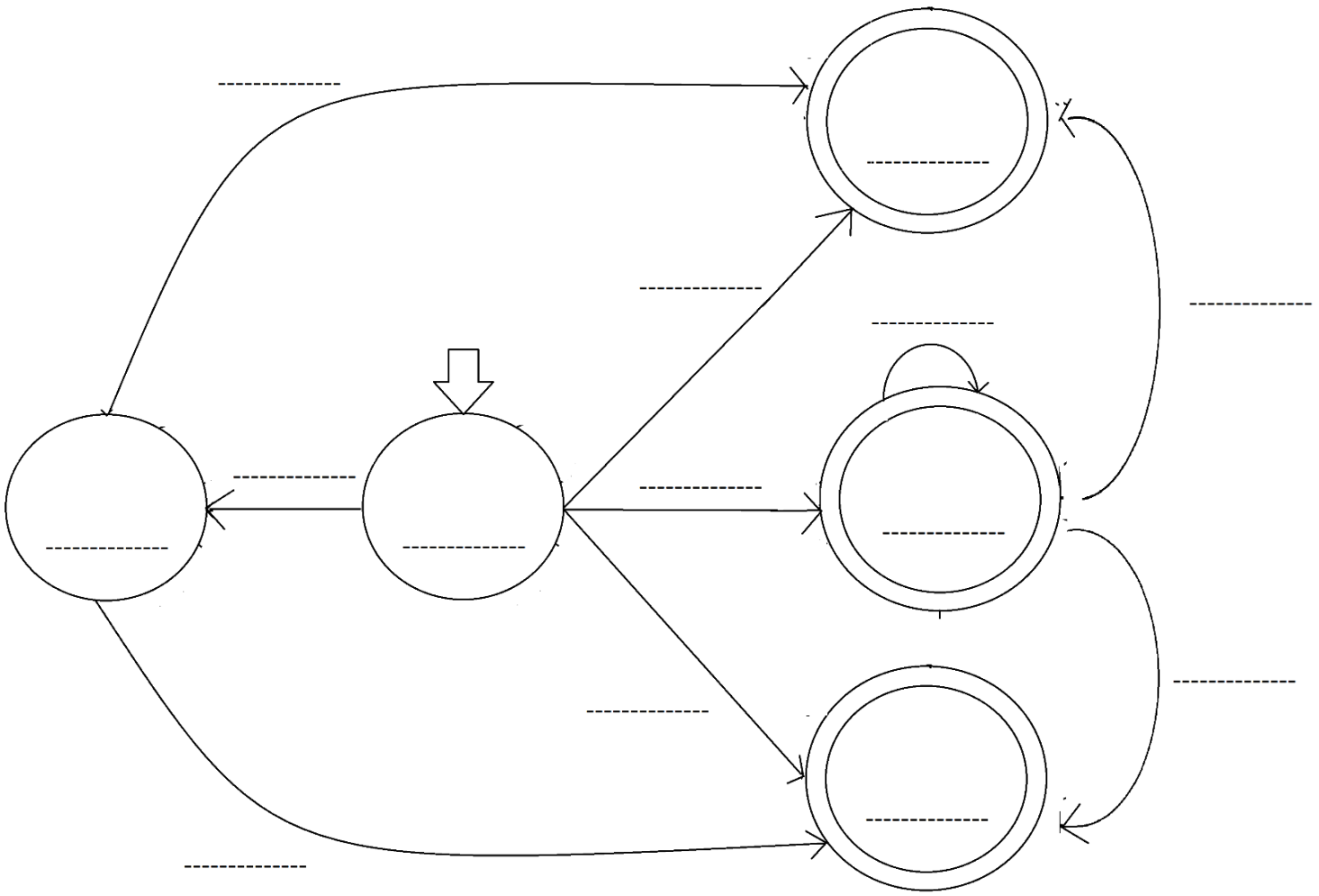
NB : on rappelle qu'il n'y a qu'une seule erreur dans notre séquence

L'objectif : créer un schéma capable d'identifier le type d'erreur (version expert)

E_X+1_ref = E_X+1_ech | E_X+1_ref = E_X_ech | E_X+1_ech = E_X_ref | E_X+n_ref = E_X+n_ech | E_X+n+1_ref=E_X+n_ech | E_X+n_ref=E_X+n+1_ech | substitution | insertion | délétion| Erreur en position X | et | ou | et pas |

NB : certaines étiquettes correspondent à plusieurs cases

NB2 : certaines cases comportent plusieurs étiquettes



Rappels Python - Bases de la Programmation

1. Variables et Types de Données

En Python, on peut stocker des informations dans des **variables**.
Exemple de types de données :

```
nom = "Mettre_un_nom" # Chaîne de caractères (str)

age = 28        # Nombre entier (int)

taille = 1.65   # Nombre décimal (float)

est_etudiant = True   # Booléen (True/False)
```

2. Les Conditions (if, elif, else)

Les conditions permettent d'exécuter du code **seulement si une condition est vraie**.

```
age = 28

if age >= 18:

    print("Majeur")

else:

    print("Mineur")
```

3. Les Boucles (for, while)

Les **boucles** permettent de répéter des actions plusieurs fois.

a. Boucle `for` (utilisée pour parcourir une liste ou une chaîne)

```
for lettre in "ADN":

   print(lettre)  # Affiche A puis D puis N
```

b. Boucle `while` (s'exécute tant que la condition est vraie)

```
x = 0

while x < 3:

    print(x)

    x += 1  # Affiche 0, 1, 2
```

4. Les Listes

Une **liste** permet de stocker plusieurs valeurs.

```
nucleotides = ["A", "T", "C", "G"]

print(nucleotides[0])  # Affiche 'A'
```

a. Manipulation des listes

```
bases = ["A", "T", "C", "G"]

print(bases[0])  # Premier élément -> A

print(bases[-1]) # Dernier élément -> G

bases.append("U")  # Ajoute l'Uracile (ARN)

print(bases)
```

b. Parcourir une liste avec une boucle `for` :

```
for base in nucleotides:

    print(base)  # Affiche A, T, C, G
```

```
sequence = "ATGCGT"

for i in range(len(sequence)):  # Itère sur les indices

    print(f"Position {i}: {sequence[i]}")
```

c. Utilisation de len( )

La fonction `len()` permet de connaître le nombre d'éléments dans une liste ou la longueur d'une chaîne.

```
nucleotides = ["A", "T", "C", "G"]

print(len(nucleotides))  # Affiche 4 car la liste a 4 éléments
```