Programmation en Python – Exercices – Devoirs

Exercice 1 corrigé disponible

Donner les lignes d'une fonction Python, la plus simple possible permettant d'apporter une solution aux problèmes mathématiques suivants :

1. retourner le PGCD et le PPCM de deux nombres entiers A et B
2. indiquer les coordonnées d'un vecteur du plan défini par 2 points
3. savoir si 2 vecteurs du plan sont-ils colinéaires ?
4. 3 points du plan sont-ils alignés ?
5. Quelle est une valeur approchée de π en utilisant la méthode Monte-Carlo ?
6. Quelle est une valeur approchée de $\sqrt{x}$ en utilisant la méthode de Héron ?
7. demander une valeur initiale et une valeur finale puis retourner le taux d'évolution en %.
8. Indiquer la moyenne, premier quartile, médiane et troisième quartile d'une série statistique
9. Indiquer l'écart type, écart interquartile d'une série statistique
10. Retourner la solution de l'équation $x^3 = k$ en utilisant la méthode de dichotomie

Exercice 2 corrigé disponible

1. Définir une fonction Python **de(nface)** simulant le lancer d'un dé à n face et retournant le numéro du dé
2. Définir une fonction Python **echantillon(numero,n)** simulant n lancés de dé à 6 faces et retournant la fréquence du numéro
3. Définir une fonction Python **decision(f,p,n)** retournant Vrai ou Faux pour vérifier la validité d'une condition d'échantillonnage au seuil de 95 %

Exercice 3 corrigé disponible

On représente un brin d'ADN par une chaîne de caractères qui peut contenir quatre caractères différents : "A" (Adénine), "C" (Cytosine), "G" (Guanine) et "T" (Thymine).

1. *Écrire une fonction* `estADN` *qui prend en argument une chaîne de caractères et renvoie* `True` *si cette chaîne correspond à un brin d'ADN et qui renvoie* `False` *sinon.*

   Contrat:
   Par exemple, `estADN("TTGAC")` *et* `estADN("GCAATAG")` *renvoient* `True`
   mais `estADN("AMOG")` *et* `estADN("CaTg")` *renvoient* `False`.

2. *Ecrire une fonction* `masseMolaire` *qui calcule la masse molaire d'une séquence ADN passée en argument. Chaque lettre a une masse donnée :* `"A"` *(135 g/mol);* `"T"` *(126 g/mol);* `"G"` *(151 g/mol);* `"C"` *(111 g/mol). La masse totale est la somme des masses des lettres de la séquence.*

   Contrat:
   Par exemple, `masseMolaire("AGATC")` *renvoie (135 + 151 + 135 + 126 + 111) g/mol, c'est-à-dire 658 g/mol.*

3. *Chaque base possède une base complémentaire avec laquelle elle peut s'associer :* `"A"` *et* `"T"` *sont complémentaires, et* `"C"` *et* `"G"` *sont complémentaires. Écrire une fonction* `brinComp` *qui étant donné un brin d'ADN* `b` *calcule et renvoie son brin complémentaire, c'est à dire le brin constitué des bases complémentaires de* `b`.

   Contrat:
   Par exemple, `brinComp("A")` *renvoie* `"T"` *et* `brinComp("AAGT")` *renvoie* `"TTCA"`.

4. *(Bonus, ⋆⋆⋆) Écrire une fonction* `sous_sequence` *qui prend en argument deux chaînes de caractères représentant des brins d'ADN et renvoie* `True` *si le premier brin est une sous-séquence du deuxième, et qui renvoie* `False` *sinon.*

   Contrat:
   Par exemple, `sous_sequence("ATC", "GGTATCG")` *renvoie* `True`
   et `sous_sequence("GC", "AAT")` *renvoie* `False`.

Exercice 4

1. Quel est le résultat donné par cet algorithme ? Retrouver cette valeur par le calcul

```
def pourcent(t1,t2):
    return ((1-t1/100)*(1+t2/100)-1)*100


>>> pourcent(25,35)
```

2. Voici le programme suivant :

```
def algo(n):
    s=0
    for k in range (n):
      s+=(-1)**k*4/(2*k+1)
    return s
```

a. Exécuter algo(10000)

b. Construire le tableau suivant et expliquer le rôle de ce programme (arrondir les résultats à 10^{-3} près)

n	1	2	3	4	5	6	7	8	9	10
algo(n)										
$\lvert algo(n)-\pi \rvert$										