Bases de programmation en Python – Exercices – Devoirs

Exercice 1

Donner les lignes d'une fonction Python, la plus simple possible permettant de 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. Réaliser l'échantillonnage de la face 1 d'un dé à 6 côtés; effectuer un test de décision pour savoir si la face 1est truquée en utilisant un intervalle de fluctuation à 95 %
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

Exercice 2

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

1. Retourner la valeur du n^{ième} terme de la suite de Fibonacci
2. Indiquer les solutions réelles de l'équation $ax^2+bx+c=0$
3. Savoir si 2 vecteurs du plans sont perpendiculaires ?

Exercice 3

La distance de Hamming entre deux mots est une notion utilisée dans de nombreux domaines (télécommunications, traitement du signal, ...). Elle est définie, pour deux mots de même longueur, comme le nombre de positions où les deux mots ont un caractère différent. Écrire une fonction `hamming` *qui calcule la distance de Hamming entre deux mots lorsqu'ils ont la même longueur, et qui renvoie* `-1` *sinon.*

Contrat:
Par exemple, `hamming("aaba", "aaha")` *renvoie 1,* `hamming("poire", "pomme")` *renvoie 2 et* `hamming("stylo", "bouteille")` *renvoie* `-1`.

Exercice 4

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 5

1. *Écrire une fonction* `suppression` *qui prend en argument une chaîne de caractères avec un seul caractère* `c` *et une chaîne de caractères* `s` *et qui renvoie* `s` *dans laquelle on a supprimé la première occurrence de* `c`. *Si* `c` *n'apparaît pas dans* `s`, `suppression` *renvoie la chaîne inchangée.*

   Contrat:

   Par exemple, `suppression("a", "baldaquin")` *renvoie* `"bldaquin"` *et* `suppression ("d", "fleur")` *renvoie* `"fleur"`.

2. *Écrire une fonction* `scrabble` *qui prend en argument une chaîne de caractère* `mot` *et une chaîne de caractère* `lettres_disponibles` *et qui renvoie* `True` *si on peut écrire* `mot` *en utilisant au plus une fois chaque lettre de la chaîne* `lettres_disponibles` *et qui renvoie* `False` *sinon.*

   Contrat:

   Par exemple, `scrabble("maison", "auiysmzanpo")` *renvoie* `True` *et* `scrabble("bungalows", "hbteslo")` *renvoie* `False`.

   Indice *: Parcourir les caractères de* `mot` *et les supprimer successivement dans* `lettres_disponibles` *avec la fonction* `suppression`.

3. *Deux mots sont des anagrammes si on peut obtenir l'un à partir de l'autre en permutant les lettres. Par exemple,* `"police"` *et* `"picole"`.
   Écrire une fonction `anagramme` *qui prend en argument deux chaînes* `u` *et* `v` *et qui renvoie* `True` *si* `u` *et* `v` *sont des anagrammes et qui renvoie* `False` *sinon.*

   Contrat:

   Par exemple, `anagramme("parisien", "aspirine")` *renvoie* `True` *et* `anagramme("chaise", "disque")` *renvoie* `False`.

Exercice 6

En 1202, le mathématicien Leonardo Fibonacci inventa l'énigme suivante : Un homme met un couple de lapins dans un lieu isolé de tous les côtés par un mur. Combien de couples obtient-on en un an si chaque couple engendre tous les mois un nouveau couple à compter du troisième mois de son existence ?

1. *Si l'on connaît le nombre de couples de lapins au mois* n *et au mois* $n+1$*, comment connaître le nombre de couples de lapins au mois* $n+2$ *?*
2. *Résoudre l'énigme.*
3. *Plus généralement, écrire une fonction* `fibonacci` *qui calcule le nombre de couples de lapins au bout d'un nombre* n *de mois donné en argument.*

Exercice 7

Écrire une fonction `factorielle` *qui renvoie le produit des entiers jusqu'à l'entier entré en paramètre. Ainsi,* `factorielle(5)` $= 1 \times 2 \times 3 \times 4 \times 5 = 120$.

On rappelle que par définition, `factorielle(0)` $= 1$.

Exercice 8

Écrire une procédure `splitLetters` *qui prend en entrée une chaine de caractères* `s` *et affiche le nombre de voyelles et le nombre de consonnes en minuscule et sans accent contenues dans* `s`. *Si* `s` *contient des caractères qui ne sont pas des lettres en minuscule et sans accent, alors* `splitLetters` *affichera tout simplement le message* `"On accepte que des lettres en minuscule sans accent"`.

Contrat:

— `splitLetters("initiation")` *affiche* `"voyelles = 6, consonnes = 4"`
— `splitLetters("Initiation")` *affiche* `"On accepte que des lettres en minuscule "`