Séance 4b: Exercices sur les chaînes de caractères

Université Paris-Diderot

Objectifs:

- Manipuler les chaînes de caractères
- Définir et utiliser des fonctions
- Concevoir et programmer des algorithmes

Dans cette séance, vous résoudrez des exercices et des problèmes sur des chaînes de caractères. Vous écrirez des boucles et définirez des fonctions intermédiaires pour rendre votre code plus lisible et concis.

Exercice 1 (ord et chr, ⋆)

- *Un ordinateur représente un caractère par un entier appelé code Unicode. Par exemple, le caractère* `"j"` *est représenté par le code* 106. *La fonction* `ord`*, qui attend en paramètre une chaîne de caractères avec un seul caractère, renvoie le code associé à ce caractère. Déterminer à l'aide de la fonction* `ord` *les codes associés aux caractères* `"a"`*,* `"m"` *et* `"M"`*.*
- *Écrire une procédure* `minuscule` *qui prend une chaîne de caractères avec un seul caractère en argument et affiche* `"minuscule"` *si ce caractère est une lettre minuscule,* `"MAJUSCULE"` *si le caractère est une majuscule, et* `"caractère spécial"` *sinon.*

  Contrat:

  `minuscule("a")` *doit afficher* `"minuscule"`*,* `minuscule("H")` *doit afficher* `"MAJUSCULE"` *et* `minuscule("é")` *doit afficher* `"caractère spécial"`*.*

  Indice : *Les codes des caractères* `"a"` *à* `"z"` *se suivent, idem pour les codes des caractères* `"A"` *à* `"Z"`*.*
- *La fonction inverse de* `ord` *est la fonction* `chr`*. Cette fonction prend en argument un code Unicode et renvoie le caractère associé. Par exemple,* `chr(106)` *renvoie* `"j"`*. Écrire une procédure* `alphabet` *qui affiche* `abcdefghijklmnopqrstuvwxyz` *à l'aide d'une boucle.*

□

Exercice 2 (Recherche de caractère, ⋆)

- *Écrire une fonction* `cherche` *qui prend en argument une chaine de caractères avec un seul caractère* `c` *et une chaîne de caractères* `s` *et qui renvoie* `True` *si* `c` *apparaît dans* `s`*, et* `False` *sinon.*

  Contrat:

  Par exemple, `cherche ("a", "cheval")` *renvoie* `True` *et* `cherche ("a", "école")` *renvoie* `False`*.*
- *Modifier la fonction* `cherche` *afin qu'elle renvoie la position de la première occurrence du caractère* `c` *dans la chaîne* `s`*, ou* `-1` *si le caractère n'est pas présent.*

  Contrat:

  `cherche("a","école")` *doit renvoyer* `-1` *et* `cherche("a","cheval")` *doit maintenant renvoyer 4.*

□

Exercice 3 (Distance de Hamming, ⋆)

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 (Scrable et anagrammes, ⋆ ⋆ ⋆)

— *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"`.

— *Écrire une fonction* `scrabble` *qui prend en argument deux chaînes de caractères* `mot` *et* `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`.

— *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 5 (ADN, ⋆⋆)

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).*

— *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`.

— *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.*

— *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"`.

— *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 6 (Calculatrice, ⋆⋆⋆)

Écrire une fonction `somme` *qui prend en argument une chaîne de caractères comprenant des entiers séparés par des symboles +, comme "7+52+3" ou "2+6+74+13" et qui renvoie le résultat de la somme. Si l'argument n'a pas le bon format,* `somme` *doit renvoyer* `-1`.

> ***Contrat:***
> *Par exemple,* `somme("3+8")` *renvoie 11 mais* `somme("+7+8")` *et* `somme("4+3+9+")` *renvoient* `-1`.

□