université PARIS-SACLAY | FACULTÉ DES SCIENCES D'ORSAY

13/12/2017

Langages Dynamiques
Master 1 ISD

Examen

durée 2h00

Modalités

L'examen dure 2h00. Les notes personnelles sont autorisées. Un aide-mémoire Javascript et Python est disponible en fin d'énoncé. Tous les exercices sont faisables en utilisant uniquement l'aide mémoire (en plus de connaissances basiques sur les langages). Le barème est indicatif et **proportionnel à la difficulté et au temps de rédaction prévus pour la question**.

1 Itérateurs en Python (5 points)

Dans cet exercice, il est **interdit** d'utiliser des boucles (`for`, `while`) ou des fonctions récursives pour parcourir les listes. Vos réponses ne doivent utiliser que les opérateurs `sorted`, `reduce`, `map` ou les compréhension de liste `[` e `for . . .in . . .]`.

1. (1.5 point) Écrire une fonction Python `three_grams(l)` qui étant donné une liste de taille au moins 3 de chaînes de caractères toutes de taille 1, génère la liste des $3-grams$ correspondante, c'est à dire la liste de toutes les sous-chaînes de taille trois, dans l'ordre de la liste initiale. Par exemple :

   ```
   > three_grams([ "a", "b", "c", "d", "e", "f", "g" ])
   [ "abc", "bcd", "cde", "def", "efg" ]
   ```

2. (0.5 point) Écrire une fonction Python `moyenne` qui prend en argument une liste de nombres supposée non-vide et qui renvoie la moyenne de ces nombres.

3. (3 points) Écrire une fonction Python `moyenne_par_matiere` qui prend en argument une liste de triplets $(\textit{nom d'étudiant}, \textit{matière}, \textit{note})$ où *nom d'étudiant* et *matière* sont des chaînes de caractères et *note* est un nombre et qui renvoie pour chaque matière la moyenne des notes obtenues dans cette matière (donc une liste de paires chaînes de caractères, nombre). La liste d'entrée est dans un ordre quelconque, par exemple :

   ```
   > moyenne_par_matiere([ ("Michel", "Maths", 10), ("Jean", "Français", 12),
                           ("Paul", "Maths", 14), ("Anne", "Histoire", 18),
                           ("Vladimir", "Français", 15) ])
   [ ("Français", 13.5), ("Histoire", 18), ("Maths", 12) ]
   ```

   Vous pouvez réutiliser la fonction « `moyenne` » de la question précédante dans la ré-écrire.

2 Question de cours Javascript (2 points)

On considère le code Javascript ci-dessous (exécuté dans la console d'un navigateur Web) :

```
var x = 0;
for (var j = 0; j < 2; j++) {
    setTimeout(function () { x = x + j; console.log(x); }, 1000);
}
```

On suppose que le temps d'exécution de la boucle est négligeable (i.e. qu'elle se termine en moins d'une seconde). Dire ce qu'affiche le code (vous justifierez brièvement en disant ce que valent `x` et `j` à chaque mise à jour de `x`).

```
class Expr { }

class Const extends Expr {
   constructor (n) {
    this.value = n;

   eval() { return this.value; }
}

class Binop extends Expr {

   constructor (e1, e2) {
    this.left = e1;
    this.right = e2;
   }
}

class Add extends Binop {
   eval() {
     return this.left.eval() + this.right.eval();
   }
}
```

```
class Sub extends Binop {
   eval() {
    return this.left.eval() - this.right.eval();
   }
}

class Mult extends Binop {
   eval() {
    return this.left.eval() * this.right.eval();
   }
}

class Div extends Binop {
   eval() {
    return this.left.eval() / this.right.eval();
   }
}
```

Figure 1 – Une bibliothèque Javascript d'expressions arithmétiques.

3 Le compte est bon (9 points)

On se propose d'implémenter un algorithme de résolution pour le jeu « le compte est bon ». Dans ce jeu, un nombre est tiré au hasard entre 100 et 999 (inclus). On tire ensuite au hasards **6** cartes parmi 24 possibles :

— les vingt cartes 1, 1, 2, 2, …, 9, 9, 10, 10

— les quatre cartes 25, 50, 75 et 100

le but du jeu est de pouvoir calculer le nombre tiré au hasard en utilisant les cartes au plus une fois chacune et en les combinant avec les quatre opérations : addition, soustraction, multiplication, division. On a le droit d'effectuer une soustraction uniquement si le résultat est positif et on a le droit d'effectuer une division uniquement si le résultat tombe juste (*i.e.* le reste dans la division euclidienne vaut 0). Par exemple, pour le nombre 218 et les cartes 25, 2, 2, 10, 1, 9, on veut trouver que

$$218 = 25 \times (10 - 2) + 9 \times 2$$

(le 1 n'ayant pas été utilisé).

Expressions arithmétiques

On se donne le code Javascript de la figure 1, représentant des expressions arithmétiques. On rappelle qu'en Javascript, un constructeur absent (par exemple dans `Add`, `Sub`, `Mult`, `Div`) est équivalent à :

```
constructor(x1, . . ., xn) {
  super(x1, ..., xn);
}
```

c'est à dire un constructeur qui appelle automatiquement celui de la classe parente avec tous ses paramètres.

Questions

1. (1 point) Écrire une fonction `genNumber()` renvoyant un entier aléatoire compris entre 100 et 999 (inclus).
2. (2 point) Écrire une fonction `genCards()` renvoyant un tableau de 6 objets de type `Const`, étant chacun une constante tirée avec les même probabilité que les cartes du jeu (i.e. ces 6 cartes sont choisies successivement parmi 24 telle que décrites dans les règles du jeu). On pourra d'abord générer un tableau contenant les 24 valeurs, puis en tirer 6 au hasard.

3. (2 points) Ajouter aux classes `Const`, `Add`, `Sub`, `Mult` et `Div` une méthode `.toString()` qui convertit l'objet en chaîne de caractères. Chaque sous-expression est parenthésées, sauf les constantes, même quand c'est inutile. Par exemple, on obtient : `(3+(2*5))` pour l'objet crée par

   ```
   new Add(new Const(3), new Mult(new Const(2), new Const (5)))
   ```

   De plus, on demande que la méthode soit ajoutée *a posteriori* à la classe, c'est à dire sans modifier le code donné en Figure 1.

4. (1 point) Écrire une fonction `removeElements(a, i, j)` qui étant donné un tableau `a` et deux entiers `i` et `j` supposés être deux indices valides et distincts pour `a`, renvoie une **copie** de `a` privé des deux éléments situés aux indices `i` et `j`.

5. (3 points) Écrire une fonction récursive `solve(n, cards)` qui étant donné un entier `n` et un tableau `cards` contenant des expressions essaye de résoudre le compte est bon en appliquant l'algorithme suivant :
   — pour chaque paire d'éléments distincts e_1 et e_2 de `cards`
     — retirer `e1` et `e2` du tableau
     — pour chacune des quatre opérations *op*
     — créer l'expression $e = e_1$ *op* e_2.
     — calculer la valeur de v de e.
     — Si $v = n$, e est une solution, la renvoyer.
     — sinon si `v` est un entier strictement positif, ajouter e à la liste des cartes et appeler récursivement l'algorithme.
   — si on a testé toutes les paires d'expressions et toutes les opérations renvoyer `null`.

6. (1 point) On suppose que la page Web contenant ce code dispose d'un bouton cliquable permettant d'arrêtre la recherche de solution. Cliquer sur le bouton fait passer une variable globale `interrupt` à `false`. On modifie le code pour que le début de la fonction `solve` soit :

   ```
   function solve(n, cards) {
     if (intterupt) return throw "Interrupted";
     ...
   }
   ```

   et on protège l'appel à `solve` par un `try catch` approprié.

   Un utilisateur lance d'abord le calcul récursif, puis clique sur le bouton. Cependant le calcul ne s'arrête pas et l'exception n'est jamais levée. Proposez une explication (on suppose que le code de gestion de bouton n'a pas de bug, c'est à dire que quand le handler est appelé, la variable `interrupt` est effectivement mise à `true`).

4 RGB (3 points)

On considère la page HTML ci-dessous :

```
<html>
  <head>
    <title>couleur</title>
    <script type="text/javascript" src="color.js"></script>
  </head>
  <body>
  <div id="mydiv" style="width:6cm; height: 2cm; border: 1pt solid black;"></div>
  <input id="myinput" type="text"></input>
  <button id="mybutton">Afficher</button>
  </body>
</html>
```

et dont le rendu graphique est le suivant :

On souhaite que lorsque l'utilisateur clique sur le bouton « Afficher », la chaîne de caractères du champs de texte soit extraite. Si cette chaîne représente trois nombres r, v, b, compris entre 0 et 255 (inclus) sans espace, séparés par des virgules, alors la couleur de fond du `div` (propriété CSS `background`) est mise à `"rgb(`r`,`v`,`b`)"` et le contenu de la boîte reste vide. Sinon, la boite contient le message `"Couleur invalide"`.

Donner intégralement le code Javascript du fichier `color.js` pour qu'il implémente la fonctionalité demandée.

Aide-mémoire

Python

On rappelle qu'en Python, l'opérateur « + » est surchargé et permet de faire l'addition entre deux nombres, la concaténation de deux chaînes de caractères et la concaténations de deux listes.

La notation e`[`i`:`j`]` renvoie une sous-liste (resp. une sous-chaîne ou un tuple) de la liste (resp. de la chaîne de caractères ou du tuple) e, comprise entre les indices i (inclus) et j exclus. Si j est absent, il est égal à la longueur de e. Si i est absent il est égal à 0. Si $i = j$, on peut utiliser la notation e`[`i`]`. Les indices commencent à 0 et un indice négatif représente une position par rapport à *la fin* de la liste (*i.e.* le dernier élément est à l'indice -1, l'avant dernier -2, *etc.*).

Fonctions utiles

l`.append(`v`)` ajoute la valeur v à la fin de la liste l (qui est modifiée en place).

`len(`e`)` renvoie la taille d'un objet e (qui peut être une liste, une chaîne de caractères, ...).

`map(`f`,`l`)` renvoie la liste résultant de l'application de f à chaque élément de la liste l.

`reduce(`f`,`l`, initializer=None)` Si $l = [v_1, \ldots, v_n]$ alors si `initializer` est une valuer v_0 différente de `None`, calcule $f(\ldots(f(f((v0, v1), v2), v3), \ldots), v_n)$ sinon calcule $f(\ldots(f(f(v1, v2), v3), \ldots), v_n)$.

`sorted(`l`, `cmp`)` trie la liste l selon la fonction de comparaison cmp qui doit renvoyer -1, 0, ou 1 selon que son premier argument est respectivement plus petit, égal ou plus grand que son deuxième.

`[ `$f(x)$` for `x` in `l` if `$c(x)$` ]` : équivalent à calculer d'abord la liste l' de tous les éléments x de l vérifiant la condition $c(x)$, puis renvoyer `map(`f`, `l'`)`.

Javascript

Tableaux

t`.forEach(`f`)` appelle f pour chaque élément du tableau par ordre d'indice croissant. f reçoit en paramètre la valeur de la case courante, l'indice de la case courante et enfin le tableau tout entier.

t`.length` renvoie la longueur du tableau t.

t`.map(`f`)` appelle f pour chaque élément du tableau par ordre d'indice croissant. f reçoit en paramètre la valeur de la case courante, l'indice de la case courante et enfin le tableau tout entier. Renvoie le tableau des résultats de f pour chaque application.

t`.pop()` renvoie la dernière case du tableau et la supprime (lève une exception si le tableau est vide).

t`.push(`v`)` ajoute la valeur v en dernière position du tableau.

t`.splice(`i`, `n`, `v_0`, . . . , `v_k`)` supprime n éléments de t à partir de l'indice i et y insère ensuite v_0, ..., v_k à la position i, $i+1$, ..., $i+k$ (en décalant les éléments restants). Le tableau est modifié en place.

Chaînes de caractères

s_1 `+` s_2 concatène deux chaines de caractères. Si l'une des deux opérande n'est pas une chaine, appelle la la méthode `.toString()` de cette opérande pour la convertir en chaîne.

s`.length` renvoie la longueur de la chaîne s.

s`.match(`r`)` renvoie un tableau où l'indice 0 contient la première sous-chaîne de s vérifiant l'expression régulière r, et ou les cases i suivante contiennent le résultat des sous-chaînes capturées par le $i^{\text{ème}}$ groupe de parenthèse. La fonction renvoie `null` si aucune sous-chaîne de s ne vérifie r.

s`.split(`v`)` sépare la chaîne s selon le séparateur v et renvoie le tableau des chaînes séparées.

Mathématiques

`Math.random()` renvoie un nombre flottant entre 0 (inclus) et 1 (exclus).

`Math.trunc(`n`)` renvoie la partie entière (arrondie vers 0) du nombre flottant n : `Math.trunc(2.5)` $\rightarrow$ `2` et `Math.trunc(-2.5)` $\rightarrow$ `-2`.

`Number.isNaN(`n`)` renvoie `true` si et seulement si n est la valeur spéciale `NaN`.

`Number.parseInt(`s`)` convertit la chaîne s en nombre (renvoie `NaN` si s ne représente pas un nombre).

DOM et évènements

`document.getElementById(`i`)` renvoie l'objet DOM correspondant à la balise d'identifiant i, ou `null` si une telle balise n'existe pas.

`document.createElement(`s`)` renvoie un nouvel objet DOM correspondant à une balise <s></s>

o`.addEventListener(`e`,` f`)` associe le gestionnaire d'évènement f à l'évènement e de l'objet o. e est un chaîne de caractère décrivant l'évènement (`"load"`, `"click"`, `"mousemove"`, `"keydown"`, `"keyup"`, ...) et f est une fonction.

d`.appendChild(`c`)` accroche le nœud DOM c comme dernier fils de d.

d`.children` renvoie un tableau contenant la liste des nœuds fils de d.

d`.innerHTML` représentation textuelle du code source HTML de l'intérieur de l'objet d.

d`.parent` renvoie le nœud parent de d ou `null` si d n'a pas de parent.

d`.style` permet d'accéder aux propriétés CSS de l'objet d.

d`.value` permet d'accéder à la chaîne de caractères contenue dans l'objet DOM d supposé être un élément de formulaire (champs de texte, menu,...).

`window.setInterval(`f`,` t`)` appelle la fonction f toutes les t millisecondes.

`window.setTimeout(`f`,` t`)` appelle la fonction f une seule fois, au bout de t millisecondes.