université PARIS-SACLAY | FACULTÉ DES SCIENCES D'ORSAY

jeudi 14 janvier 2021

TSVP ▶▶|

Introduction à l'Informatique
L1 MI, LDD1 IM, LDD1 MNSI

Examen

Durée 2h00, tiers temps additionnel 40 minutes

Consignes l'examen dure 2h00 et est sur 20 points. Aucun document n'est autorisé. Un aide mémoire Unix et Python est disponible à la page **??**. Le barème est indicatif et proportionnel à la difficulté des exercices.

Conditions spéciales d'examen en raison de la situation sanitaire et des règles en vigueur :

- le port du masque en salle d'examen comme sur le reste du campus est obligatoire
- ne pas anonymiser vos copies
- il est interdit d'échanger du matériel entre étudiants
- signer une fois la feuille d'émargement avec votre stylo personnel, lors du passage du surveillant
- déposez votre carte d'étudiant ou pièce d'identité sur la table, de façon à ce qu'elle puisse être lue par le surveillant
- votre copie est ramassée à votre place par les surveillants.
- une fois la copie rendue, merci de quitter la salle sans provoquer d'attroupement et dans le respect de la distanciation

Le non respect de ces règles peut entrainer l'exclusion **immédiate** de la salle d'examen.

Aide-ménoire Unix

TSVP ▶▶|

Expansion de la ligne de commande

On rappelle que lorsqu'une commande de la forme

$$\texttt{com}\ f_1\ \ \ldots\ f_n$$

est entrée dans le *shell* alors :

- Le fichier exécutable `com` est cherché dans les répertoires systèmes. S'il n'est pas trouvé, le *shell* renvoie une erreur
- Les motifs *glob* $f_1\ \ \ldots\ f_n$ sont développées pour essayer de correspondre à des fichiers. Si aucun fichier ne correspond pour un motif *glob* f_i se dernier est laissé tel quel
- La liste des fichiers trouvés est passée en argument à la commande `com` qui est exécutée

Redirections

On rappelle que lorsqu'une commande de la forme

$$\texttt{com}\ f_1\ \ \ldots\ f_n > o$$

est exécutée, alors la sortie standard de la commande est redirigée dans le fichier o. Si ce dernier existe il est écrasé. Les opérateurs de redirection sont :

`>` redirige la sortie standard et écrase le fichier

`>>` redirige la sortie standard et ajoute en fin de fichier

`2>` redirige la sortie d'erreur et écrase le fichier

`2>>` redirige la sortie d'erreur et ajoute en fin de fichier

`<` redirige le fichier vers l'entrée standard du programme

Commandes de base

`cat` f : affiche les lignes du fichier f sur la sortie standard.

`cd` p : le répertoire dénoté par le chemin p devient le répertoire courant.

`chmod` nnn p : modifie les permissions du fichier ou répertoire dénoté par le chemin p. Les permissions sont données comme trois entiers représentant respectivement les permissions pour le propriétaire, le groupe et les autres. Chaque entier est constitué de trois bits : bit en lecture, bit en écriture et bit en exécution.

`cp` f_1 f_2 : si f_2 est un nom de fichier ou un fichier n'existant pas : copie f_1 sous le nom f_2. Si f_2 est un nom de répertoire existant, copie f_1 dans le répertoire f_2.

`echo` s : affiche la chaîne de caractères s sur la sortie standard.

`ls` $f_1\ \ \ldots f_n$: affiche les fichiers f_i (ou une erreur si les f_i ne correspond pas à des noms de fichiers existants).

`mkdir` p : crée le répertoire dénoté par le chemin p.

`mv` $p_1\ \ \ldots\ \ p_n\ \ d$: déplace les fichier ou répertoires p_i dans le répertoire d qui doit exister.

`rm` f : efface le fichier f.

`rm -rf` p : efface le répertoire p.

`sort` f : affiche les lignes triées du fichier f sur la sortie standard.

Aide mémoire Python

TSVP ▶▶|

Entiers

Les entiers Python sont de taille arbitraire. Les constantes entières s'écrivent simplement `0`, `42`, `-233`. Les opérations et fonctions sur les entiers sont :

`+` addition entre deux entiers (opérateur binaire).
`-` soustraction entre deux entiers (opérateur binaire).
`*` multiplication entre deux entiers (opérateur binaire).
`//` division **entière** entre deux entiers (opérateur binaire).
`/` division **flottante** entre deux entiers (opérateur binaire).
`%` reste dans la division entière (opérateur binaire).
`**` puissance entre deux entiers (opérateur binaire)
`int(s)` conversion d'une chaîne `s` en entier. Lève une exception si la chaîne n'est pas au bon format.

Flottants

Le type `float` représente des nombre flottants. Les constantes flottantes s'écrivent en notation scientifique `0.5`, `42e3`, `-233.8e-20`. Les opérations et fonctions sur les flottants sont :

`+` addition entre deux flottants (opérateur binaire).
`-` soustraction entre deux flottants (opérateur binaire)
`*` multiplication entre deux flottants (opérateur binaire)
`/` division entre deux flottants (opérateur binaire)
`**` puissance entre deux flottants (opérateur binaire)

Booléens

Les constantes booléennes sont `True` et `False`. Les opérations et fonctions sur les booléens sont :

`and` « et » logique entre deux booléens (opérateur binaire).
`or` « ou » logique entre deux booléens (opérateur binaire).
`not` négation d'un booléen.

Chaînes de caractères

Le type `str` représente des chaînes de caractères. Les chaînes de caractères constantes sont délimitées par des guillemets simples :`'Hello, world !'`, des guillemets doubles : `"Hello, world !"` ou des triples guillemets doubles. De façon usuelle, la séquence d'échappement `\n` représente un retour à la ligne. Les opérations et fonctions sur les chaînes sont :

`s[i]` permet d'accéder au $\texttt{i}^{\text{ème}}$ caractère de la chaîne. Ce dernier est renvoyé comme une chaîne de taille 1.

`+` concaténation entre deux chaînes (opérateur binaire).

`len(s)` longueur d'une chaîne.

`s.lower()` renvoie une copie de la chaîne `s` où toutes les lettres sont converties en minuscules.

`s.split(c)` sépare la chaîne `s` selon le cararctère de séparation `c` et renvoie un tableau des composantes. Par exemple :

```
1  s = "a,b,cd"
2  t = s.split(",")
3  # t contient [ "a", "b", "cd" ]
```

`s.strip()` renvoie une copie de la chaîne `s` où les blancs (espaces, tabulations, retours à la ligne) en début et en fin de chaîne ont été supprimés.

TSVP ▶▶|

`s.upper()` renvoie une copie de la chaîne `s` où toutes les lettres sont converties en majuscule.

`str(v)` convertit la valeur `v` en chaîne.

n-uplet

On peut regrouper plusieurs valeurs en les mettant dans un n-uplet. Les n-uplets sont délimités par des parenthèses et les valeurs séparées par des virgules : `(1, True, "AB")`.
`p[i]` permet d'accéder au `i`ème élément du n-uplet `p`.

`a, b , c ,d = p` permet de définir les variables `a`, `b`, `c`, `d` comme les composantes correspondantes du n-uplet `p`.

`len(p)` nombre d'éléments d'un n-uplet.

Tableaux

Les tableaux Python permettent de stocker des collections ordonnées de valeurs. Les tableaux constants sont délimités par des crochets et les éléments séparés par des virgules : `["A", "b", "TOTO"]`.
`t[i]` permet d'accéder au `i`ème élément du tableau `t`.

`t[i] = v` permet de remplacer le `i`ème élément du tableau `t` par `v`.

`+` concaténation entre deux tableaux (opérateur binaire).

`t * n` concaténation répétée `n` fois d'un tableau `t`. (opérateur binaire). Par exemple, `[ "A"] * 3` renvoie `["A", "A", "A"]`.

`len(t)` taille d'un tableau.

Tableaux donnés par compréhension

La notation

```
[ e for x in t if c ]
```

renvoie le tableau formé par les expressions `e`. Ces dernières sont calculées tour à tour à partir de tous les éléments `x` du tableau `t` qui vérifient la condition `c`. Par exemple :

```
1  t =  [ (x * 2, str(x)) for x in range(1,5) if x % 2 == 0 ]
2  # t contient
3  # [ (4, "2"), (8, "4") ]
```

Ici, `x` prend tour à tour les valeurs 1, 2, 3, 4 (5 est exclus du `range`). Les seules valeurs vérifiant `x % 2 == 0` sont 2 et 4, on renvoie donc le tableau « `[ (2 * 2, str(2)), (4 * 2, str(4)) ]` ».

Dictionnaires

Les dictionnaires Python permettent d'associer des clés à des valeurs. Les dictionnaires constants sont délimités par des accolades et les éléments séparés par des virgules. Les clés et les valeurs sont séparés par « : » : `d = {"A":1, "B": 42 }`.
`d[k]` permet d'accéder à la valeur associée à la clé `k`.

`d[k] = v` permet de remplacer ou ajouter la valeur associée à la clé `k`.

TSVP ▶▶|

`k in d` renvoie `True` si et seulement si la clé `k` est dans le dictionnaire `d`.

`d.keys()` renvoie le tableau des clés du dictionnaire `d`.

`d.values()` renvoie le tableau des valeurs du dictionnaire `d`.

Entrées sorties

`print(...)` permet d'afficher les valeurs passées en paramètres sur la sortie standard. Les valeurs affichées sont séparées par des espaces.

`open(p)` renvoie un descripteur de fichier correspondant au chemin `p`, donné comme une chaîne de caractères. Lève une erreur si le fichier n'existe pas ou n'est pas accessible en lecture.

`list(f.readlines())` renvoie le tableau des lignes du descripteur de fichier `f`. Toutes les lignes sauf éventuellement la dernière se terminent par `\n`.

`f.close()` referme le fichier associé au lignes du descripteur de fichier `f`.

Comparaisons

En Python les opérations de comparaison permettent de comparer n'importe quelles valeurs du même type :
`<, <=, >, >=, ==, !=` comparaisons entre deux valeurs (respectivement, inférieur, inférieur ou égal, supérieur, supérieur ou égal, égal et différent), (opérateur binaire).

`sorted(t)` renvoie une copie triée du tableau `t` trié par ordre croissant.

`min(a, b)` et `max(a, b)` renvoie le minimum ou maximum de deux valeurs.
Les entiers, flottants et chaînes de caractères sont comparés selon leur ordre usuel. Les booléens peuvent être comparés, et `False` est plus petit que `True`. Les n-upplets sont comparés composante par composante, par exemple `(1, 10)` est plus petit que `(2, 5)` qui est lui même plus petit que `(2, 6)`. Il en va de même pour les tableaux. Les dictionnaires **ne sont pas comparables** (les comparer ou appeler `sorted` sur un tableau de dictionnaires lève une exception).