université PARIS-SACLAY | FACULTÉ DES SCIENCES D'ORSAY

vendredi 9 janvier 2026

Introduction à l'Informatique
L1 MI, LDD1 IM

Examen

Durée 2h00, tiers temps additionnel 40 minutes

Consignes l'examen dure 2h00 et est sur 20 points. Les notes personnelles sont autorisées. Un aide mémoire Unix et Python est disponible à la page 5. Le barème est indicatif et proportionnel à la difficulté des exercices. Les téléphones portables doivent être éteints et rangés.

1 Question de cours (1 point)

Qu'est-ce qu'un *protocole* (répondre en 5 lignes maximum).

2 Unix et ligne de commande (5 points)

Situation initiale : L'utilisateur effectuant les commandes et `toto`, son répertoire personnel est `/home/toto`. Ce dernier contient :

- un répertoire `images` avec les fichiers `img000.jpg`, `img001`, `img002`, ..., jusqu'à `img250.jpg` (compris).
- un répertoire `document` contenant le fichier `cv.txt`

Écriture de commandes Unix

Pour chaque question, le répertoire de départ est indiqué et la commande doit fonctionner **sans changer de répertoire** (`cd` interdit). Une seule ligne de commande est autorisée par action.

1. *Répertoire de départ :* `/home/toto` Créer un nouveau répertoire `backup`.
2. *Répertoire de départ :* `/home/toto` Afficher les lignes du fichier `document/cv.txt` triées dans la console.
3. (a) *Répertoire de départ :* `/home/toto/images` Afficher le contenu de `cv.txt` avec un chemin relatif.
   (b) *Répertoire de départ :* `/home/toto/images` Afficher le contenu de `cv.txt` avec un chemin absolu.
   (c) *Répertoire de départ :* `/home/toto/document` Créer un fichier `images_backup.txt` contenant la liste de tous les fichiers `.jpg` de `images` en utilisant un chemin relatif.
4. *Répertoire de départ :* `/home/toto` Changer les permissions du fichier `document/cv.txt` pour que le propriétaire ait lecture/écriture, le groupe lecture seule et les autres aucun droit.
5. *Répertoire de départ :* `/home/toto` Créer un fichier `images_pairs.txt` contenant la liste de tous les fichiers `.jpg` dans `images` dont le numéro est pair.
6. *Répertoire de départ :* `home/toto` Supprimer tous les fichiers `.jpg` de `images` dont le numéro est compris entre 200 et 250.

Explication de commandes Unix

On se place de nouveau dans la situation initiale entre chaque question, dans le répertoire `/home/toto`. Pour chacune des commandes ci-dessous, **expliquez en une phrase ce que fait la commande**, et indiquez **si elle réussit ou si elle échoue**.

1. `chmod 710 images`
2. `ls -l images/[a-d]*.jpg`
3. `ls -l images/[^0]*.jpg`
4. `cp document/cv.txt images/img005.jpg`

3 Réseaux (4 points)

1. On considère les adresses IP : `192.148.140.17` et `192.152.2.3`, avec le masque réseau `255.252.0.0`.
   — Combien de bits valent zéro dans le masque réseau ?
   — Les deux adresses sont-elles sur le même sous-réseau ? Justifier en donnant l'adresse réseau correspondant à chaque IP.
2. Quel est le protocole qui s'occupe de déterminer la route entre deux machines séparées par un nombre arbitraire de machines intermédiaires sur Internet ?

4 Programmation en Python (10 points)

On considère un fichier texte contenant uniquement des livres. Chaque ligne contient :
— un identifiant du livre (un entier)
— le titre du livre (une chaîne de caractères)
— l'auteur du livre (une chaîne de caractères)
— couple d'entiers (séparés par une virugle). Si ce couple est `0,0` alors le livre est disponible (il n'est pas emprunté). Sinon, le couple d'entiers est de la forme `id,AAAAMMJJ` où `id` est l'indentifiant d'un usager et `AAAAMMJJ` est un entier représentant la date de retour du livre. Ce couple signifie que l'usager dont l'identifiant est `id` doit rendre son livre le `JJ/MM/AAAA`. On remarque que si on a deux entiers représentant des dates dans ce format, alors on peut simplement les comparer avec les comparaisons usuelles de Python, on a bien que `20260201 > 20260125` car le 1er février 2026 (`20260201`) est bien postérieur au 25 janvier 2026.

Toutes ces informations sont séparées par des caractères « `|` ». Voici un exemple de fichier :

```
301|Le Hobbit|J.R.R. Tolkien|0,0
302|1984|George Orwell|102,20251201
303|Le Hobbit|J.R.R. Tolkien|102,20260105
305|Le Seigneur des Anneaux|J.R.R. Tolkien|0,0
307|1984|George Orwell|0,0
309|Le Petit Prince|Antoine de Saint-Exupéry|0,0
299|1984|George Orwell|0,0
316|Fondation|Isaac Asimov|102,20260201
310|Le Hobbit|J.R.R. Tolkien|0,0
311|1984|George Orwell|0,0
```

On fera attention, le même livre (même titre) peut être présent en plusieurs exemplaires (identifiant dans la première colonne différent). Par exemple pour le livre dont le titre est « Le Hobbit », il y a trois exemplaires, d'identifiant 301, 303, et 310. Deux sont disponible et celui d'identifiant 303 a été emprunté par l'usager d'identifiant 102 qui devait le rendre le 5 janvier 2026.

On dispose en plus de deux **variables globales** contenant la date et les noms des usagers. Par exemple ces deux variables peuvent contenir :

```
DATE = 20260120
USAGERS = {
  101: "Alice Dupont",
  102: "Bob Martin",
  103: "Claire Petit"
}
```

La date est un entier au format `AAAAMMJJ` et la variable `USAGERS` est un dictionnaire dont les clés sont des entiers (l'identifiant d'un usager) et les valeurs son prénom et nom.

Questions

1. (2 points) Écrire une fonction `charge_fichier(f)` qui renvoie un tableau de dictionnaires contenant les clés : `id` (associé à un entier), `"titre"` et `"auteur"` associés à des chaînes de caractères et `"emprunt"`, associé un couple d'entiers. Vous pouvez supposer que le fichier est bien formé. Dans la suite on appelle un tel tableau un **tableau d'exemplaires**.

2. (1 point) Écrire une fonction `nb_livres_dispo(tab)` qui prend en argument un tableau d'exemplaires et renvoie le nombre **d'exemplaires** dont le champ `emprunt` vaut `(0,0)`.
3. (1.5 point) Écrire une fonction `plus_grand_id(tab, titre)` qui prend en argument un tableau d'exemplaires et un titre et qui renvoie l'identifiant de l'exemplaire dont le titre est `titre`. Si plusieurs exemplaires ont le même titre, vous devez renvoyer l'identifiant le plus grand. Attention, le tableau d'exemplaires `tab` n'est pas forcément trié. Si aucun exemplaire ne correspond, votre fonction doit renvoyer `None`.
4. (1.5 point) Écrire une fonction `nb_retards(tab, nom)` qui prend en argument un tableau d'exemplaires et un nom de personne et renvoie le **nombre d'exemplaires en retard** pour cette personne. Si aucune personne n'a ce nom dans `USAGERS`, renvoyer `None`. Pour le fichier d'exemple et les variables `USAGERS` et `DATE` décrites précédemment, `nb_retards("Bob Martin")` . envoie `2`
5. (2 points) Écrire une fonction compte_exemplaires(tab) qui prend en argument un tableau d'exemplaires et renvoie le titre du libre qui a le plus d'exemplaires. En cas d'égalité, vous pouvez renvoyer le titre de votre choix. Pour le fichier d'exemple, votre fonction renvoie `"1984"` qui est disponible en 4 exemplaires (on compte tous les exemplaires qu'ils soient disponibles ou empruntés)
6. (2 points) Écrire une fonction `affiche_html(tab)` qui prend en argument un tableau d'exemplaires et qui affiche sur la sortie standard une table HTML avec les colonnes : `Titre`, `Auteur`, `Emprunteur`, `Retard`.
   — La case `Emprunteur` contient le nom si le livre est emprunté, ou reste vide si disponible.
   — La case `Retard` contient le mot `Retard` si le livre est en retard par rapport à `DATE`, ou reste vide sinon.
   — Les lignes doivent être affichées par ordre alphabétique du titre.

   On ne demande que les balises correspondant à la table HTML, pas tout le fichier.

   Indications : On rappelle qu'une table HTML est constitué d'une balise `<table> ... </table>` dans laquelle se trouve des balises `<tr> ... </tr>` (une par ligne). Dans chacune de ces balises se trouve une balise `<td>...</td>` (une par case).

   De plus, pour trier le tableau par titre, vous pouver le transformer en tableau de tuples de la forme `(titre, id, dico)` où `titre` est le titre de l'exemplaire `id` est l'identifiant d'exemplaire et `dico` est le dictionnaire original. Ce tableau de tuple peut ensuite être trié avec la fonction **`sorted`** (qui va trier par titre, puis pour le même titre par identifiant d'exemplaires).

Aide-ménoire Unix

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 développés pour essayer de correspondre à des fichiers. Si aucun fichier ne correspond pour un motif *glob* f_i, ce 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 compris entre 0 et 7, qui représentent 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 correspondent pas à des noms de fichiers existants). L'option `-l` affiche la liste détaillée.

`mkdir` p : crée le répertoire dénoté par le chemin p. Tous les répertoires du chemin sauf le dernier doivent exister.

`mv` $p_1\ \ldots\ p_n\ d$: déplace les fichiers 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

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)
`float(s)` conversion d'une chaîne `s` en flottant. Lève une exception si la chaîne n'est pas au bon format.

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. Les indices commencent à 0.

`+` 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.

`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`$^{\text{ème}}$ élément du n-uplet `p`. Les indices commencent à 0.

`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`$^{\text{ème}}$ élément du tableau `t`. Les indices commencent à 0.

`t[i] = v` permet de remplacer le `i`$^{\text{ème}}$ 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`.

`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 aux 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), (opérateurs binaires).

`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-uplets sont comparés composante par composante en suivant l'ordre lexicographique, 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).