Session 2: Fondamentaux informatique pour bien démarrer les cours et tests de positionnements AP1/APP1

Dans ce TP vous allez :

— Être initié à la pratique du terminal Unix : interface Shell, système de fichiers, redirection, archivage et droits ;

— Utiliser le terminal pour exécuter un notebook Jupyter et un programme Python ;

— Manipuler l'IDE Thonny ainsi que PythonTutor qui seront utilisés pour les cours de Python (AP1/APP1) ;

— Présentation de l'exerciseur PLaTon.

1 Pratique du terminal et de l'arborescence des fichiers Unix

Dans cette section, nous allons parler du Terminal, un outil puissant et incontournable sous unix ! Dans les exercices suivants, votre souris fait la sieste, vous ne devez surtout pas la réveiller ! Essayez de jouer le jeu SVP.

L'interaction avec un système d'exploitation (OS) peut se faire sous deux formes : soit une interface graphique utilisateur avec une souris et tout (Session 1), soit une interface en ligne de commande uniquement avec le clavier (Session 2) : il s'agit du Terminal.

Il est parfois plus efficace d'interagir avec le système via le clavier. Le terminal vous met en communication avec un interpréteur de commandes UNIX : le shell (notez cependant que la plupart des gens utilisent le terme "shell" pour désigner l'interface en ligne de commande uniquement).

Il existe plusieurs types de Shell, sous Linux, le nôtre s'appelle bash. Il en existe d'autres comme sh, csh, tcsh, ksh, zsh. Le shell vous permet de lancer des commandes, de les enchaîner, et surtout d'automatiser l'exécution d'un certain nombre de tâches.

Une interface en ligne de commande est beaucoup plus austère et peut paraître primitive par rapport à une interface graphique. Celle-ci se présente sous la forme d'une grande boîte noire dans laquelle l'utilisateur va taper des commandes (généralement une commande par ligne) pour interagir avec son OS.

Dans une interface en ligne de commande, il n'y a aucun usage de la souris : tout se fait via l'entrée de commandes tapées au clavier. Les interfaces en ligne de commande peuvent être en fait très puissantes et permettre de réaliser des opérations complexes beaucoup plus rapidement qu'avec une interface graphique.

Les shells exigent donc que l'utilisateur connaisse certaines commandes et leur syntaxe et comprenne les concepts relatifs au langage de script spécifique au shell utilisé.

Dans la suite de cette rapide initiation, nous verrons quelques commandes de bases pour le shell Linux.

▸ Exercice 1 : Premier pas avec le terminal

1. En utilisant le menu des applications, ouvrez un terminal. Le terminal vous indique qu'il est prêt en affichant une invite (ou *prompt*), en général le caractère dollar `$` ou supérieur `>`, en début de ligne.
2. Le terminal s'ouvre automatiquement dans votre répertoire personnel, pour le vérifier taper la commande `pwd`. Vous devriez avoir le retour suivant `/home/login` (ou "login" est votre login) `pwd` est une commande très importante qui vous permet de toujours savoir où vous êtes. Cette commande vous indique votre répertoire de travail (ou répertoire courant). Le résultat est toujours un chemin absolu.

3. Essayez les commandes `echo bonjour`, puis `echo -n bonjour`. Ici, `echo` est le nom de la commande, `bonjour` est un *argument*, et `-n` est une *option*. Pouvez-vous deviner à quoi sert la commande `echo` ? Et l'option `-n` ?
4. La commande `man` sert à consulter la documentation de n'importe quelle commande du système. Pour l'utiliser, on tape `man` suivi du nom de la commande souhaitée. Utilisez cette commande pour consulter la documentation de la commande `echo`.
5. Testez la commande `ls` et regarder la page du manuel correspondant à cette commande pour expliquer quelle est sa fonction et ses différentes options.
6. À l'aide de cette commande et des options vues dans le manuel, déterminez à quel moment le répertoire "preRentree" (créée en session 1) a été modifié pour la dernière fois.

1.1 Quelques astuces pour aller plus vite

1. Vous pouvez utiliser les flèches "haut" et "bas" pour faire défiler l'*historique des commandes* que vous avez déjà tapées.
2. Vous pouvez utiliser la touche "tab" pour compléter un nom de commande ou un nom de fichier.
3. Pour interrompre une commande on peut utiliser la combinaison `ctrl+c`.
4. Pour lier les commandes, on sépare les commandes par le caractère " ; " exemple `who ; date`
5. Pour se déconnecter, il suffit de taper `exit` ou la combinaison de touches `ctrl+d`.

1.2 La commande man et quelques commandes utiles

Vous l'avez vu lors de l'exercice précédent, une des premières commandes à connaître est la commande `man`. Comme vous avez pu le constater, elle affiche dans le terminal le manuel de la commande que vous placez en argument. Elle permet bien plus et nous allons le voir dans la suite.

▸ Exercice 2 : Comprendre la commande `man`

1. La documentation de `man` est très complète... Faites afficher un résumé de ce que vous pouvez faire avec la commande `man`.
   Indication : `man` est une commande comme les autres et ... comment affiche-t-on l'aide d'une commande ?
2. Utilisez la commande `touch` avec `preRentree` comme argument.
3. À quoi sert la commande `touch` ? Si vous ne voyez pas, pensez à utiliser `ls`
4. Que se passe-t-il si vous tapez `touch un_autre_fichier.txt` ?
5. Vous voulez maintenant trouver comment *effacer* un fichier mais vous ne savez pas comment s'appelle la commande.
   Utilisez `man -k` pour la trouver. Attention cette commande permet de chercher une suite de caractères donc la cass, les accents et l'orthographe sont importants. Vous pouvez par exemple chercher avec `man -k file` puis penser à comment dire effacer en anglais
   À noter : `apropos` est aussi une commande qui peut remplacer `man -k`
6. Effacez le fichier `un_autre_fichier.txt`.
   Attention ! L'effacement en utilisant la commande du terminal est *définitif*. Vous ne retrouverez pas votre fichier dans la corbeille ensuite...
7. Comment faire pour que cette commande vous demande confirmation avant de détruire un fichier ?

▸ **Exercice 3 : Quelques commandes très utiles pour la suite**

1. Regardez ensuite les pages du manuel de `cp`, `mkdir`, `rmdir`, `rm` et `cat`. Notez qu'il est possible que certaines de ces pages soient en anglais. Rappelez-vous bien des actions de ces commandes.
2. La commande `cd` (pour Change Directory) est aussi une commande très importante. C'est grâce à elle que vous pourrez vous déplacer dans l'arborescence de fichiers. Elle n'a cependant pas de page de manuel dédiée car sa description se trouve dans la page de manuel de `bash`. Ouvrez la page de manuel de `bash` (avec `man bash`) et cherchez la description de la commande `cd` en utilisant la fonctionnalité de recherche suivante : tapez `/cd` pour chercher le mot "cd" dans la page puis utiliser la touche `n` pour trouver l'occurence suivante et `N` pour trouver l'occurence précédente. (Si vous ne connaissez pas le mot occurence, cherchez sa définition sur internet.)
3. En utilisant la commande `cd (suivi de quelque chose)`, déplacez vous dans le répertoire "preRentree" créé précédemment.
4. Utiliser à nouveau la commande `pwd` pour vérifier que vous avez bien changé de répertoire de travail.
5. Avec la commande `ls` faites apparaître le contenu de ce répertoire. Normalement vous y trouverez deux fichiers créés en session 1.
6. Avec la commande `cat`, affichez le contenu des deux fichiers du répertoire (en une seule ligne SVP).
7. À présent comment faire pour remonter à la racine de votre répertoire personnel ? Essayez la commande `cd ..`

1.3 Manipuler l'arborescence des fichiers dans le terminal

Vous allez créer une arborescence de fichiers que vous utiliserez dans les exercices suivants. Quand vous ne savez pas quelle commande appeler, pensez à utiliser `man -k` et surtout à regarder l'exercice précédent.

Étant donné que le but de l'exercice est d'apprendre à se servir du terminal pour manipuler une arborescence de répertoire, continuez à laisser votre souris tranquille !

▸ **Exercice 4 : Création et manipulation de répertoire**

1. Vérifiez que vous êtes bien dans votre répertoire personnel
2. À l'aide du terminal, créez-y deux sous-répertoires `Session_1` et `Session_2`.
3. En utilisant la commande `mv`, déplacez les fichiers `"presentationNomPrenom.txt"` et `"contacts.txt "` créés au TP précédent dans le répertoire `Session_1`.
4. Vérifiez que vous avez bien déplacé les fichiers comme il le fallait
5. Placez-vous à présent dans le sous-répertoire `Session_2` et créez le fichier `cours_Session_2.txt` que nous utiliserons dans un autre exercice.
6. Remontez d'un cran dans la hiérarchie des fichiers et créez un nouveau sous-répertoire `Repertoire_A_Supprimer`
7. Dans ce répertoire, créez le fichier `fichier_exemple.txt` puis remontez d'un cran dans la hiérarchie.
8. Et maintenant .... supprimer ce répertoire ! Le pouvez-vous ? Pourquoi ?

▸ **Exercice 5 : Un point sur la notion de chemin relatif VS absolu**

Un chemin est la façon dont vous vous référez aux fichiers et aux répertoires. Il fournit l'emplacement d'un fichier ou d'un répertoire dans la structure de répertoires Linux. Un chemin se compose d'un nom et d'une syntaxe de barre oblique : `/home/login` par exemple (ou login est votre nom de login).

En tant qu'utilisateur, vous devrez utiliser un chemin lorsque vous souhaitez accéder à un certain fichier ou répertoire ou lorsque vous devez donner l'emplacement d'un fichier ou d'un répertoire à une commande ou à un script.

Comprendre la différence entre les chemins absolus et relatifs

La structure des répertoires sous Linux ressemble à la racine d'un arbre. Tout commence à la racine et se ramifie à partir de là.

Si le chemin commence par une barre oblique "/", cette première barre indique la racine (c'est-à-dire root). Le reste des barres obliques dans le chemin ne sont que des séparateurs.

Le chemin absolu commence toujours à partir du répertoire racine (/). Par exemple, `/home/login/preRentree/Session_1`

1. Un chemin relatif commence à partir du répertoire courant (le répertoire de travail). Par exemple, si vous vous trouvez dans le répertoire `/preRentree` et souhaitez afficher les informations du fichier `"contacts.txt"` du répertoire `Session_1` vous pouvez utiliser `ls -l Session_1/contacts.txt`. Il s'agit du chemin relatif (à partir du répertoire courant, sans "/"). Pour mieux comprendre regardez les commentaires marqués d'un (1) sur le schéma de la figure 1.

2. Si vous êtes dans le répertoire `Session_1/` et que vous voulez accéder directement au répertoire `Session_2/`, vous pourriez utiliser `cd /home/login/preRentree/Session_2`, soit le chemin absolu. Pour mieux comprendre regardez les commentaires marqués d'un (2) sur le schéma de la figure 1.

Faites-vous même les tests pour vérifier que vous avez bien compris.

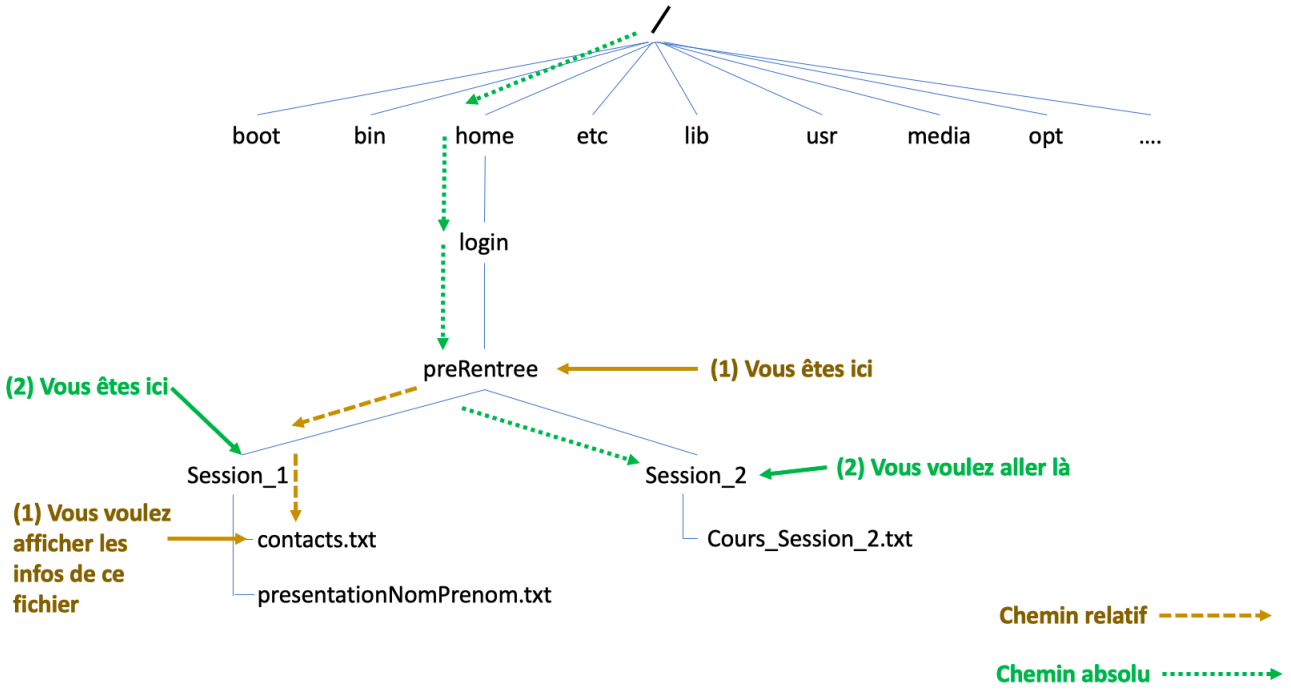

FIGURE 1 – Schéma simplifié de l'arborescence de vos fichiers

Utiliser les chemins relatif spéciaux avec . et .. répertoires

Il existe d'autres moyens pour utiliser le chemin relatif :

. (un seul point) indique le répertoire courant dans le chemin.

.. (deux-points) indique le répertoire parent, c'est-à-dire un niveau au-dessus.

1. Vérifiez dans quel répertoire courant vous vous trouvez avec la commande `pwd`, puis tapez la commande `cd ..` Tapez à nouveau `pwd`, que se passe-t-il ?
   Il peut sembler étrange d'avoir une commande spéciale représentant un état sans changement, mais c'est une directive utilement explicite. Imaginez par exemple, que dans les répertoires `Session_1` et `Session_2` il y ait un fichier exécutable portant le même nom. Imaginons que ce soit des programmes de configurations de votre environnement, avec des configurations radicalement différentes. À chaque fois que vous utiliseriez ces programmes personnalisés, vous voudriez faire attention à ce que votre ordinateur sache exactement quel commande exécuter.

2. Si vous êtes dans le répertoire `Session_2` et que vous voulez accédez directement au répertoire `Session_1`, au lieu d'utiliser `cd /home/login/preRentree/Session_1`, soit le chemin absolu, vous pourriez utiliser la commande `cd ../Session_1`. Vous l'avez compris les ".." vous emmènent directement au répertoire parent qui est `preRentree`.

3. À présent à partir de votre répertoire courant (normalement `Session_1`) utilisez, en une seule fois, le chemin relatif ".." pour remonter à la racine "/". Pour vous aidez, regardez à nouveau la hiérarchie de la figure 1.

4. Et maintenant place à la découverte, vous avez le droit de vous "balader" dans les différents répertoires en dessous de la racine et de vous déplacer de répertoire en répertoire avec les chemins relatifs et/ou absolus. N'oubliez pas non plus de lister le contenu de ces répertoires. Naturellement ne touchez à rien, ne supprimez pas de fichiers et n'en créez pas d'autre ailleurs que dans votre dossier personnel.

Astuce : la commande `cd` (seule) permet de retourner à votre répertoire personnel.

2 Redirections

Les redirections sont l'une des plus importantes possibilités offertes par le shell. Unix utilise des canaux d'entrées/sorties pour lire et écrire ses données. Par défaut le canal d'entrée est le clavier, et le canal de sortie, l'écran. Un troisième canal, le canal d'erreur, est aussi redirigé vers l'écran. Il est donc possible de rediriger ces canaux vers des fichiers, ou du flux texte de manière transparente pour les commandes Unix.

▸ Exercice 6 : En sortie

On se sert du caractère " > " pour rediriger la sortie standard (celle qui va normalement sur écran). On indique ensuite le nom du fichier où seront placés les résultats de sortie.

1. Revenez dans le répertoire de la `Session_2` et rappelez-vous que vous avez créé un fichier `cours_Session_2.txt` encore vide.
   Vous allez le remplir en utilisant la commande suivante :
   `echo "Je suis les cours de la _Session_2" > cours_Session_2.txt`
2. En tapant la commande `cat cours_Session_2.txt` que voyez vous ?
3. À présent taper la commande `echo " Et je suis très heureux !" > cours_Session_2.txt`
4. En tapant la commande `cat cours_Session_2.txt` que voyez vous ?
5. Pour éviter cette situation utiliser la double redirection "`>>`" (refaites donc toutes les opérations précédentes)

▸ Exercice 7 : En entrée

Les commandes qui attendent des données ou des paramètres depuis le clavier peuvent aussi en recevoir depuis un fichier, à l'aide du caractère "< ". Un exemple avec la commande "wc" (word count) qui permet de compter le nombre de lignes, de mots et de caractères d'un fichier.

1. Testez donc la commande `wc < cours_Session_2.txt` et regardez le résultat.

2. Utilisez la commande `cat` pour afficher le contenu du fichier `Session_2.txt` et comparez "manuellement" le résultat précédent.
3. On peut aussi utiliser à la fois les deux types de redirection : `wc < cours_Session_2.txt > compte.txt`.

À noter : Si le fichier résultant n'existe pas la commande `cat` le créera.

3 Lancer un serveur jupyter notebook, exécuter un programme python et créer une archive

Ici on passe à une pratique en rapport plus "direct" avec les cours que vous allez "subir" par la suite :-) en particulier AP1 et APP1. Pour les autres promotions cela ne fera pas de mal de manipuler un peu aussi.

▸ Exercice 8 : Décompresser et exécuter un notebook jupyter à partir du terminal

Vous avez à nouveau le droit d'utiliser votre souris mais juste pour cet exercice ! Allez sous Elearning dans la partie réservée à la session 2 et téléchargez le fichier compressé `preRentree_Jupyter.zip` que vous placerez dans votre répertoire `Session_2`.
Fin de l'utilisation de la souris et retour au terminal !

1. À partir du terminal décompresser ce fichier avec la commande `unzip`
2. Une fois décompressé, vous devriez voir apparaître un nouveau répertoire `preRentree_Jupyter`. Déplacez vous à l'intérieur de ce répertoire pour voir apparaître le notebook jupyter `preRentree_Jupyter.ipynb`. Il s'agit du notebook source de celui que vous avez vu en `Session_1`.
3. Il y a deux façons de procéder pour lancer un notebook jupyter :

   a. La première méthode vous permettra de voir et de manipuler le serveur de notebook jupyter installé sur les machines des salles de TP. Pour ce faire tapez la commande suivante `jupyter notebook`.
   Une fois le serveur lancé cliquez sur le notebook `preRentree_Jupyter.ipynb`.
   Si vous vous souvenez de la `Session_1`, vous devriez retrouver des similitudes avec l'application MyBinder (voir vidéo "serveur_jupyter.mov").
   Pour arrêter un serveur jupyter, la meilleure méthode est de retourner sur le terminal et de taper la combinaison `ctrl+c` puis de valider par "y" pour dire "yes".

   b. La deuxième méthode lance directement le serveur jupyter et le notebook jupyter. Il suffit de taper la commande `jupyter notebook preRentree_Jupyter.ipynb`. La suite est exactement la même qu'avec la méthode précédente.

▸ Exercice 9 : Mon premier programme Python

Parlons encore plus sérieusement ! Ici vous allez écrire et exécuter votre premier programme python. Pour l'occasion vous allez apprendre à lancer l'éditeur de texte à partir du terminal.

1. Replacez-vous dans le répertoire de la `Session_2`
2. Créer un fichier `monProgramme.py`
3. Ouvrez ce fichier dans un éditeur de texte avec la commande `gedit monProgramme.py`. Via cette commande vous ouvrez l'application "Éditeur de texte" de votre Linux Debian.
4. Écrivez la seule phrase suivante `print("Hello World !")` et enregistrez le fichier (vous avez le droit à la souris).
5. À présent vous allez exécuter votre premier programme python à l'UGE et voici comment faire : tapez la commande `python3 monProgramme.py`.
   Si tout c'est bien passé vous devriez voir apparaître dans le terminal la phrase `Hello World !`.

À noter : la commande utilisée pour exécuter ce script python (on dit bien script) est particulière. Sans rentrer dans les détails il y a deux versions (ou branche) du langage Python : 2 et 3. La version actuellement la plus utilisée est la numéro 3. Il est nécessaire de faire appel à cette commande pour permettre au système de comprendre que l'utilisateur souhaite exécuter un script Python.

▸ Exercice 10 : Créer une archive

Il est temps de boucler la boucle ! Puisque vous avez téléchargé une archive compressée au format ZIP à l'exercice 8, vous allez créer une nouvelle archive au format ZIP que vous uploaderez sur le Elearning. Toujours à l'aide du terminal, dans cet exercice nous vous demandons :

1. De vous déplacez dans le répertoire de la `Session_2` et de créer un nouveau répertoire `Nom_Prenom_Session_2` (où Nom et Prenom sont les votres)
2. De déplacez les fichiers `monProgramme.py` et `cours_Session_2.txt` dans le répertoire `Nom_Prenom_Session_2` que vous avez créé précédemment.
3. De compresser ce même répertoire à l'aide de la commande `zip -r Nom_Prenom_Session_2.zip Nom_Prenom_Session_2` (consulter man zip)
4. Avec la souris cette fois, uploader cette archive sous Elearning dans le répertoire "Zone de rendu de l'exercice 10"

À noter : il existe plusieurs façons de compresser et d'archiver sous unix, mais cela nécessiterait plusieurs chapitres et ce n'est pas le but ici. Pour les plus curieux et curieuses il y a la commande `tar` que vous pourriez aller voir ...

4 (En fonction du temps) Notions de propriétés et droits d'accès

Dans tout système d'exploitation, en particulier Unix, il est nécessaire de protéger le fonctionnement du système en interdisant certaines actions à l'utilisateur.
Pour cela, il existe une notion de "droits" (ou "permissions") et de "groupe".
Chaque fichier et répertoire, possède des droits qui lui sont propre, des autorisations d'accès individuelles. Lors d'un accès, le système vérifie si celui-ci est permis selon la catégorie (le groupe) de l'utilisateur.

4.1 Droits des répertoires et des fichiers

▸ Exercice 11 : Affichage des droits

Pour afficher à l'écran les droits alloués à un répertoire ou un fichier, il faut utiliser la commande `ls -l` qui permet de lister toutes les informations connexes possibles dont les droits des répertoires et des fichiers.

1. Déplacez-vous dans votre répertoire `preRentree` et tapez la commande `ls -l` pour voir apparaître les droits sur vos répertoires `Session_1` et `Session_2`. Vous devriez voir apparaître les lignes suivantes.
   Dans la figure 2 suivante, seules les informations intéressantes ont été affichées.

2. Faites de même sur le répertoire `Session_1` pour voir apparaître les droits sur les fichiers `"presentationNomPrenom.txt"` et `"contacts.txt "`. Dans la figure 3 suivante, seules les informations intéressantes ont été affichées.

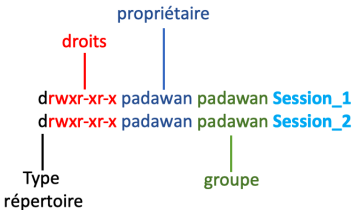

FIGURE 2 – Droits sur les répertoires `Session_1` et `Session_2`

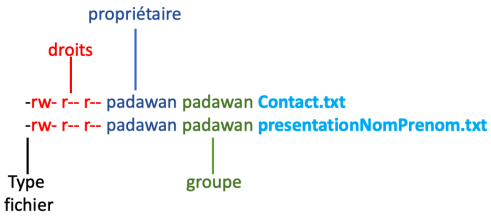

FIGURE 3 – Droits sur les fichiers du répertoire `Session_1`

Les droits sont représentés par une suite de 9 caractères.

1. Les trois premiers sont les permissions de l'utilisateur propriétaire du fichier ou du répertoire, seul à même de changer les droits ou le groupe de son fichier et/ou répertoire.
2. Les trois suivants celles du groupe auquel appartient le propriétaire (cela pourrait être le domaine de l'UGE par exemple). Un groupe d'utilisateurs est un ensemble d'utilisateurs privilégiés ayant en général des permissions moindres que le propriétaire d'un fichier mais plus grandes que la catégorie qui suit.
3. Les trois dernières celles de tous les autres utilisateurs.

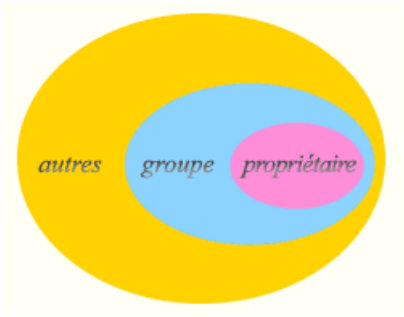

FIGURE 4 – Modèle concentrique des droits d'accès

Le schéma ci-haut montre qu'un ensemble de propriétaires forme un groupe, qu'un ensemble de groupes forme la catégorie "autres" (qui sont tous ceux qui prétendent à accéder aux données). L'accès à un sous-ensemble concentrique suppose a priori d'obtenir des droits supplémentaires.

À chacune des 3 catégories d'utilisateurs, on associe une des 8 combinaisons différentes possibles pour l'allocation des droits que la figure suivante récapitule.

Triplet	Droits correspondants
`---`	aucun
`--x`	exécution
`-w-`	écriture
`-wx`	écriture et exécution
`r--`	lecture
`r-x`	lecture et exécution
`rw-`	lecture et écriture
`rwx`	lecture, écriture et exécution

FIGURE 5 – Combinaison des droits

4.2 Modifier les droits d'accès

Lors de sa création, un fichier ou un répertoire dispose de droits par défaut. Il est offert au propriétaire d'un fichier (et seulement à lui) de modifier les droits du fichier. C'est-à-dire qu'il peut supprimer des droits ou bien en rajouter de nouveaux à chacune des trois catégories d'utilisateurs. Pour cela, on utilise la commande `chmod` (change mode) selon la syntaxe suivante : `chmod` *liste_droits* fichier(s)

`chmod` permet d'accorder ou de retirer au(x) fichier(s) et répertoires passé(s) en arguments les permissions correspondant à *liste_droits* pour une/des catégorie(s) d'utilisateur(s) cible(s) (propriétaire, groupe ou autre).

Pour affecter à chaque catégorie la liste des droits voulus, on utilise souvent une notation symbolique selon la syntaxe : `chmod` *catégorie + opérations + liste_droits* fichier(s).

Les termes catégorie, opération et liste des droits doivent êtres respectivement remplacés par leur notation décrite dans la figure suivante :

Degré de propriété	Action	Type d'accès
`u` (utilisateur)	`+` (ajoute le droit)	`r` (lecture)
`g` (groupe)	`-` (enlève le droit)	`w` (écriture)
`o` (autres)	`=` (définit le droit)	`x` (exécution
`a` (tout le monde)		

FIGURE 6 – chmod en notation symbolique

catégorie + opérations + liste_droits peuvent être par exemple :

- `u+r` pour rajouter au propriétaire le droit en lecture,
- `g-w` pour retirer aux membres du groupe le droit en écriture,
- `o+x` pour donner aux autres utilisateurs le droit en exécution,
- ou une combinaison de ces possibilités (ex : `u-wx`, `ug-w`, `ug-wx`).

4.2.1 Protéger complètement un fichier

Ici, vous interdisez à tout autre que vous l'accès à votre script `monProgramme.py` (c'est ce qu'on appelle couramment le "mode parano", Figure 7). Vous retirez les droits de lecture au groupe (g) et aux autres (o)

La commande utilisée est : `chmod` *og-r* fichier(s)

```
(base) padawan@DebianUPEM2019:~/prerentree/Session_2/Nom_Prenom_Session_2$ ls -l
total 0
-rw-r--r-- 1 padawan padawan 0 août  26 22:56 monProgramme.py
(base) padawan@DebianUPEM2019:~/prerentree/Session_2/Nom_Prenom_Session_2$ chmod og-r monProgramme.py
(base) padawan@DebianUPEM2019:~/prerentree/Session_2/Nom_Prenom_Session_2$ ls -l
total 0
-rw------- 1 padawan padawan 0 août  26 22:56 monProgramme.py
```

FIGURE 7 – "mode parano"

4.2.2 Rétablir des droits normaux pour un fichier

```
(base) padawan@DebianUPEM2019:~/prerentree/Session_2/Nom_Prenom_Session_2$ chmod og+r monProgramme.py
(base) padawan@DebianUPEM2019:~/prerentree/Session_2/Nom_Prenom_Session_2$ ls -l
total 0
-rw-r--r-- 1 padawan padawan 0 août  26 22:56 monProgramme.py
```

FIGURE 8 – Rétablir les droits normaux du script `monProgramme.py`

4.2.3 Protéger un répertoire

Comme pour un fichier on utilise aussi la commande `chmod`

```
(base) padawan@DebianUPEM2019:~/prerentree/Session_2$ ls -l -d Nom_Prenom_Session_2/
drwxr-xr-x 2 padawan padawan 4096 août  26 22:56 Nom_Prenom_Session_2/
(base) padawan@DebianUPEM2019:~/prerentree/Session_2$ chmod go-rx Nom_Prenom_Session_2/
(base) padawan@DebianUPEM2019:~/prerentree/Session_2$ ls -l -d Nom_Prenom_Session_2/
drwx------ 2 padawan padawan 4096 août  26 22:56 Nom_Prenom_Session_2/
```

FIGURE 9 – "mode parano"

4.2.4 Rétablir des droits normaux pour un répertoire

```
(base) padawan@DebianUPEM2019:~/prerentree/Session_2$ chmod go+rx Nom_Prenom_Session_2/
(base) padawan@DebianUPEM2019:~/prerentree/Session_2$ ls -l -d Nom_Prenom_Session_2/
drwxr-xr-x 2 padawan padawan 4096 août  26 22:56 Nom_Prenom_Session_2/
```

FIGURE 10 – Rétablir les droits normaux d'un répertoire

4.3 Entraînez-vous

Dans cette partie deux exercices pour vous entraîner sur la gestion des droits :

— un exercice sur la gestion des droits pour les fichiers

— un exercice sur la gestion des droits pour les répertoires

▸ **Exercice 12 : Exécuter un fichier.**

1. Créez un répertoire `Essai` (si vous n'en possédez pas déjà un), et un fichier `mon_ls` dans ce répertoire, et écrivez-y quelques mots de votre choix.
2. Notez à l'aide de `ls -l` les permissions actuelles du répertoire `Essai` et du fichier `mon_ls`.
3. En utilisant la commande `chmod`, retirez-vous le droit en lecture et en écriture sur le fichier `mon_ls`. Vérifiez l'effet obtenu en essayant d'afficher le contenu du fichier sur la fenêtre du terminal avec la commande `cat`, puis de remplacer ce contenu par une phrase différente.
4. Un fichier exécutable est simplement un fichier dont vous possédez le droit en exécution. Rétablissez le droit en écriture puis remplacez le contenu du fichier `mon_ls` par le texte `"echo Contenu du dossier; ls"`. Ajoutez-vous le droit en exécution, et exécutez le fichier `mon_ls` en tapant `./mon_ls` dans le terminal (depuis le répertoire qui le contient). Quel est le problème ?
5. Rétablissez enfin le droit en lecture et tentez à nouveau d'exécuter le fichier. Que se passe-t-il ?

Si le sens de chaque droit pour un fichier ordinaire est assez clair, il est parfois moins intuitif pour un répertoire, comme vous le verrez dans l'exercice suivant.

▸ **Exercice 13 : Un test.**

1. Placez-vous dans le répertoire `Essai`, et retirez-vous le droit en lecture pour ce répertoire. Listez le contenu du répertoire avec `ls`, puis exécutez ou affichez le contenu du fichier `mon_ls`. Qu'en déduisez-vous ? Rétablissez le droit en lecture sur `mon_ls` et sur `Essai`.
2. Créez dans `Essai` un fichier `nouveau` ainsi qu'un répertoire `Test`. Retirez au fichier `nouveau` **et** au répertoire `Essai` le droit en écriture. Tentez de modifier le fichier `nouveau`, de le déplacer, de le supprimer. Rétablissez ensuite le droit en écriture au répertoire `Essai`. Tentez de modifier le fichier `nouveau`, puis de le déplacer ou de le supprimer. Que constatez-vous ?
3. Positionnez-vous dans votre répertoire personnel, puis retirez le droit en exécution du répertoire `Essai`. Tentez de créer, supprimer, ou modifier un fichier dans le répertoire `Essai`, de vous y déplacer, d'en faire la liste, etc. Qu'en déduisez-vous quant au sens du droit en exécution pour les répertoires ?

5 Bien suivre les cours et se former à Python

Vous allez bientôt commencer votre formation en L1 ainsi que vos cours en algorithmique et programmation avec le langage Python. Ce chapitre est dédié à la présentation des outils et applications qui vous permettront de bien démarrer.

5.1 Thonny : éditeur Python pour les cours d'AP1 (APP1 et autre ...)

Quand on découvre un langage de programmation et surtout la logique algorithmique, il faut utiliser des outils simples, facile d'accès et de compréhension.

À la vérité quand on débute en Python, un simple éditeur de texte est très suffisant, comme vous avez pu le voir à l'exercice 9.

Cependant, plus on progresse dans les subtilités du langage et plus on a envie d'aller un peu plus "vite" et de bénéficier d'un éditeur de texte plus convivial, plus riche de fonctionnalités d'assistance comme par exemple la complétion de code ou la coloration syntaxique du langage (Il s'agit d'un formatage automatique du texte avec une couleur et une fonte spécifique en fonction de son type : variable, fonction, ...).
Bref on parle ici d'IDE (Integrated Development Environment), soit en français ; Environnement Intégré de Développement.

Sur internet, dans votre entourage et auprès de vos collègues des années précédentes vous trouverez et entendrez parler d'une multitude d'IDEs tous plus pointus et complets les uns les autres. Néanmoins, plus pointu et plus complet veut dire aussi plus complexe voir beaucoup plus complexe et il y a de fortes chances pas très utile pour le niveau d'une L1 (qu'on soit en MIASHS, AP1 ou en APP1) !
Des IDEs comme PyCharm, Visual Studio Code (VS Code) ou Spyder peuvent être très séduisants et ont pignon sur internet (ça veut dire qu'internet ne jurent que par eux). Ces IDEs font littéralement le café ... à la condition de savoir s'en servir ... à la condition d'en avoir réellement besoin aussi.
Pour résumer, un IDE aussi fantastique soit-il ne codera jamais pour vous ! Il s'agit avant tout d'un assistant éditeur "intelligent" pour faciliter la vie des développeurs.

Soyons clair, personne ne vous empêchera de choisir l'IDE que vous voudrez tant que vous faites, avec ce dernier, ce qui est demandé pendant vos cours.

Pour vous faciliter la vie, l'équipe enseignante d'AP1 vous propose l'IDE Thonny : https://thonny.org/. Thonny est un IDE très simple et très adapté à vos premiers pas sur Python.

Dans cette section, l'objectif n'est pas de vous former à Thonny mais de vous donner les moyens de le faire et éventuellement de l'installer sur vos machines perso. Il y a d'ailleurs un tuto sympa en français que vous pourriez suivre pour vous former : https://www.codeflow.site/fr/article/python-thonny.

5.2 Python Tutor

Dans la série des outils qui vous aideront à survivre à vos cours de L1 et potentiellement même après (avec d'autres langages), Python Tutor est un must (https://pythontutor.com/) !

Cet outil permet de visualiser en live le déroulement d'un programme ligne par ligne en présentant les variables dans un graphique très intuitif. Il a aussi l'avantage de ne pas nécessiter d'installation.

Python Tutor est un outil pédagogique qui est très largement utilisé par les enseignants. Savoir l'utiliser serait un plus.

À ce sujet regardez la vidéo sous Elearning : *"PythonTutor.mov"*.

ATTENTION : Il ne s'agit pas d'un IDE, vous ne pourrez pas programmer avec ! Vous ne pourrez pas sauvegarder votre code naturellement non plus ... puisque ce n'est pas un IDE.

5.3 La plateforme PLaTon

PLaTon pour **PLatform for Learning and Teaching online** est un exerciseur pédagogique multi-matières open-source crée à l'Institut Gaspard Monge (soit l'UGE).

C'est une manière de s'entrainer via des exercices qui vous permettront de conforter et de développer vos connaissances en Python (parfois même de s'entraîner avant les examens ...). PLaTon peut être vu un peu comme WIMS, les exercices sont à faire en autonomie et les corrections sont automatiques (exercices auto-corrigés).

Les exercices PLaTon ne sont pas notés et pas du tout obligatoires, c'est cependant un bon moyen de s'entraîner et de pratiquer, voire d'aller plus loin.

L'accès à PLaTon (pour les exos Python) se fait en étant connecté à Elearning via le lien présent sous Elearning de la "Pré Rentrée L1 2022/23" (et pas autrement ! ! !). C'est une question de droit d'accès.

Cliquez sur le lien "Python_L1_Entrainement-22_23", une fois connecté vous verrez apparaître la fenêtre suivante :

La fenêtre de la figure 11 met à disposition l'ensemble des chapitres d'exercices Python actuellement disponibles.

Pour démarrer une série d'exercices il faut cliquer sur la flèche orange d'un chapitre afin d' accéder à la page d'accueil du chapitre en question (figure 12).

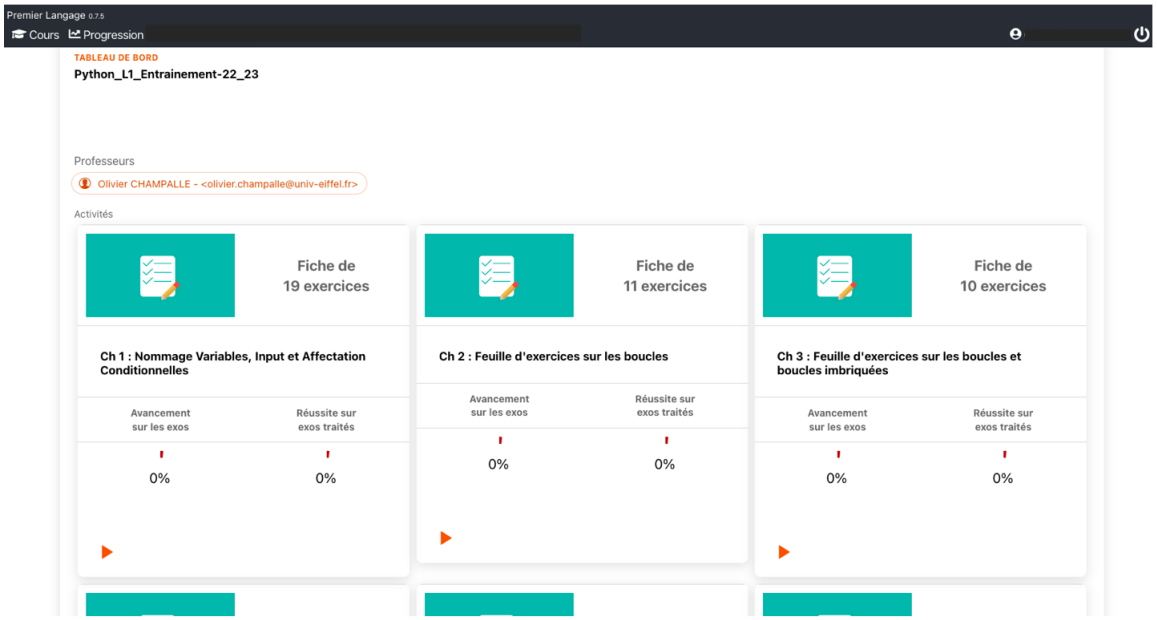

FIGURE 11 – Page des exercices de Python en libre service pour les L1

Un clique sur le bouton vert “Commencer” (figure 12) lancera le premier exercice de la série. Vous pouvez même directement cliquer sur n’importe quel exercice dans l’ordre que vous désirez.

La vidéo “Presentation_PLaTon.mov” sous Elearning vous présente le fonctionnement sur deux exercices.

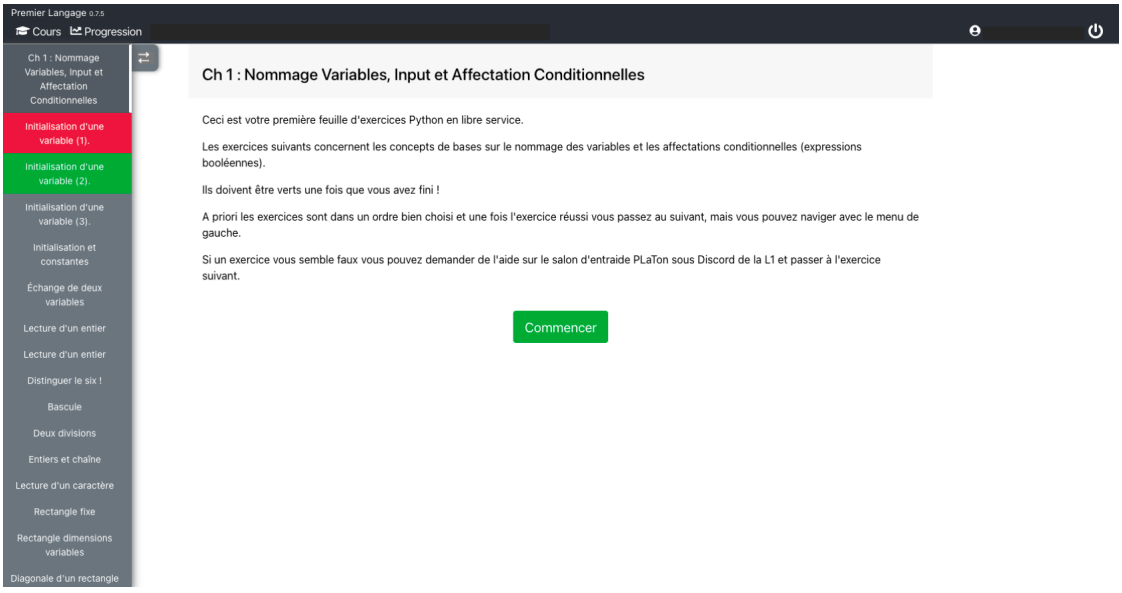

FIGURE 12 – Ch1 : Nommage Variables, Input et Affectation Conditionnelles

Voilà c’est la fin de la Session_2 !

A présent vous êtes formés et préparés pour entrer de plein pied dans votre formation !

Nous vous souhaitons une pleine et complète réussite.