TP 02

Savoirs et compétences :

- ❒ *AA.C1 : Manipuler un OS ou un IDE*
- ❒ *AA.S1 : Se familiariser aux principaux composants d'une machine numérique*
- ❒ *AA.S3 : Se familiariser à la manipulation d'un IDE*

1 Applications

1. **Lisez attentivement tout l'énoncé avant de commencer.**
2. Après la séance, vous devez rédiger un compte-rendu de TP et l'envoyer au format électronique à votre enseignant.
3. Le seul format accepté pour l'envoi d'un texte de compte-rendu est le format PDF. Votre fichier s'appellera impérativement `tp02_kleim_durif.pdf`, où « `kleim` » et « `durif` » sont à remplacer par les noms des membres du binôme.
4. Ce TP est à faire en binôme, vous ne rendrez donc qu'un compte-rendu pour deux.
5. Vous préciserez en préambule de votre compte-rendu les noms des membres du binôme ainsi que le système d'exploitation sur lequel vous avez travaillé.
6. Ayez toujours un crayon et un papier sous la main. Quand vous réfléchissez à une question, utilisez-les!
7. Vous devez être autonome. Ainsi, avant de poser une question à l'enseignant, merci de commencer par :
   - relire l'énoncé du TP (beaucoup de réponses se trouvent dedans) ;
   - relire les passages du cours [1] relatifs à votre problème ;
   - effectuer une recherche dans l'aide disponible sur votre ordinateur (ou sur internet) concernant votre question.

   Il est alors raisonnable d'appeler votre enseignant pour lui demander des explications ou une confirmation!

Le but de ce TP est de vous faire de manipuler un système d'exploitation par lignes de commandes, puis de prendre en main `Python`.

Attention : les étudiants travaillant sous Unix (Linux et MacOS) répondront d'abord aux questions des parties 1.2 et 1.3, ceux travaillant sous Windows répondront d'abord aux questions des parties 1.4 et 1.5. Tous les étudiants finiront par répondre aux questions de la partie 1.6.

Pour les manipulations de fichier, vous utiliserez le code source du logiciel `Python`, que vous pourrez trouver dans le dossiers groupes. à l'adresse suivante (où `X` est à remplacer par 1 ou 2) :

```
~/groupes/mpsX/données/TP02/cpython-4243df51fe43
```

1.1 Introduction : terminal et shell

Unix a été inventé à un moment où l'utilisateur avait la possibilité d'interagir avec l'ordinateur via un *terminal*, c'est-à-dire la combinaison d'un clavier et d'un écran pouvant écrire (en général) 25 lignes de 80 caractères (en une seule couleur, généralement vert ou orange sur fond noir). Cette façon d'interagir avec la machine peut paraître archaïque de nos jours mais elle est pourtant d'une puissance diabolique.

Vous trouverez un émulateur de terminal dans le menu « Applications », sous-menu « Accessoires », et sélectionnez « LXTerminal ». Ceci démarre un programme, appelé *shell* ou *interprète de commandes*. Ce *shell* vous donne quelques informations, et affiche un symbole `$`, appelé invite (ou *prompt* en anglais) signe qu'il attend vos ordres.

Pour lui donner un ordre, il suffit de taper le nom de la commande désirée, éventuellement suivie d'un espace puis d'options ou d'arguments séparés par des espaces, puis de valider par la touche Entrée.

1. Dans le cas fort improbable où vous ne vous en souviendriez pas.

Sous Windows, le programme `cmd.exe` permet d'ouvrir une console fonctionnant sous le même principe. Une remarque : les noms de dossiers comportant des espaces devront être écrits entre guillemets (par exemple : `"Mes Documents"`). De manière générale, c'est une très mauvaise idée de placer des espaces dans des noms de dossiers (ou de fichiers).

Vous trouverez dans la table 1 les équivalents Windows de quelques commandes Unix.

Commande Unix	Commande Windows	Description rapide
`cd`	`cd`	Change le répertoire courant
`cp`	`copy`	Copie un ou des fichiers
`exit`	`exit`	Ferme la console
`less`	`more`	Lit un document texte
`ls`	`dir`	Liste les fichiers dans un répertoire
`man`	`help`	Ouvre une page de documentation
`mkdir`	`mkdir`	Crée un répertoire
`mv`	`move`	Déplace un ou des fichiers
`pwd`	`cd`	Affiche le répertoire courant
`rm`	`del`	Supprime un ou des fichiers

TABLE 1 – Correspondances des commandes Unix et Windows

Variable Unix	Variable Windows
~	`%UserProfile%`
/	\

TABLE 2 – Correspondances des variables Unix et Windows

1.2 Unix : utilisation d'un terminal.

Une commande très pratique est `man` : elle permet d'obtenir le manuel de quasiment toutes les commandes. On l'utilise sous la forme `man` *page* où *page* est la page de manuel désirée.

Q 1 : Tapez `man less`. Que se passe t-il?

Vous pouvez faire défiler le texte ligne par ligne avec Entrée ou page par page avec la barre d'espace et quitter `man` avec la touche q.

Q 2 : Quelle commande permet d'afficher la page suivante dans `less`? de quitter `less`?

1.3 Unix : fichiers et répertoires

Sous Unix (dont la distribution GNU/Linux est un représentant) les fichiers sont organisés hiérarchiquement en une arborescence unique de répertoires. La racine de cette arborescence, c'est-à-dire le répertoire supérieur de la hiérarchie contenant tous les fichiers auxquels à accès le système, est noté `/`. Ses sous-répertoires directs (de l'ordre de la dizaine ou quelques dizaines de répertoires), comme `home`, `media`, ... sont notés `/home`, `/media`, ...

Le *chemin absolu* d'un fichier est l'adresse complète de son emplacement, débutant de la racine et passant par tous les sous-répertoires requis pour atteindre le fichier visé.

Le *chemin relatif* d'un fichier est l'adresse de son emplacement, écrite à partir d'un emplacement de l'arborescence que l'on appelle *répertoire courant* (en anglais : *current working directory*). Ce répertoire courant est initialisé par défaut à un point prédéterminé de l'arborescence (répertoire « maison », ou *home*), mais peut ensuite être modifié.

Q 3 : Que fait précisément la commande `ls`?

La commande `cd` permet de changer de répertoire courant, `pwd` permet d'afficher le répertoire courant. En particulier, la commande `cd` *d*,où *d* est le nom absolu ou relatif d'un répertoire, change le répertoire courant en *d*. Essayez avec `cd /usr/bin` par exemple.

Q 4 : Que fait `cd` sans argument (*i.e.* `cd` non suivi du nom d'un répertoire)?
Q 5 : Changer le répertoire courant (par exemple en `/usr/bin`). Que fait `cd ~`? De quoi ~ est-il l'abréviation? Quel est le chemin absolu du répertoire ~?

Q 6 : Que fait la commande `mkdir ?` En utilisant une ligne de commande, créer dans le répertoire ~ un sous-répertoire nommé TP02. Qu'observe-t-on en exécutant la commande `ls ~`?

Q 7 : Après avoir éxécuter la commande `cd TP02` à partir du répertoire ~, qu'affiche la commande `pwd` ?

Copier le dossier `cpython-4243df51fe43` depuis le dossier `groupes/mpsX/TP02` dans le répertoire `~/TP02` en passant par un explorateur de fichiers. Exécuter la commande `cd` dans le terminal.

Pour Unix : dans un répertoire, les fichiers et répertoires dont le nom commence par un point sont dits *cachés*.

Q 8 : En consultant le manuel de `ls`, trouver la commande qui permet d'afficher les fichiers et répertoires cachés.

Q 9 : Dans `~/TP02`, vous pouvez alors voir deux répertoires cachés. Quels sont leurs noms ?

En fait, dans chaque répertoire du système, il existe deux répertoires cachés avec ces deux mêmes noms.

Q 10 : Que désignent les répertoires `.` et `..` ? Que donne un `cd` sur chacun de ces répertoires ?

Q 11 : Changez le répertoire courant en `~/TP02/cpython-4243df51fe43/Lib/test/capath`. Que fait alors `cd ../../multiprocessing` ?

Q 12 : Comment obtenir grâce à la commande `ls` et l'option `-l` la taille de tous les fichiers de `TP02/cpython-4243df51fe43/M` en les triant par ordre croissant de taille ?

Q 13 : *Facultatif :* Avec la commande précédente, que remarquez-vous quant à la taille des sous-répertoires de `TP02/cpython-4243df51fe43/Modules` ?

Q 14 : *Facultatif :* En utilisant la commande `du`, donner la taille du répertoire `cjkcodecs`. En comparant ce résultat à celui de la question précédente, que pouvez-vous dire de la manière dont Linux considère les répertoires ?

1.4 Windows : utilisation d'un terminal.

Une commande très pratique est `help` : elle permet d'obtenir le manuel de quasiment toutes les commandes. On l'utilise sous la forme `help` *page* où *page* est la page de manuel désirée.

Q 15 : Tapez `help more`. Que se passe t-il ?

Vous pouvez faire défiler le texte ligne par ligne avec Entrée ou page par page avec la barre d'espace et quitter `man` avec la touche q.

Q 16 : Quelle commande permet d'afficher la page suivante dans `more` ? de quitter `more` ?

1.5 Windows : fichiers et répertoires

Sous Windows, les fichiers sont organisés hiérarchiquement en plusieurs arborescences de répertoires. La racine de chacune de ces arborescences, c'est-à-dire le répertoire supérieur de la hiérarchie contenant tous les fichiers auxquels à accès le système, est nommé par une lettre (`A:`, `B:` *etc.*).

Le séparateur de nom de dossier est alors `\`. Par exemple, le sous-répertoire `Home` de la racine `C:` est noté `C:\Home`.

Le *chemin absolu* d'un fichier est l'adresse complète de son emplacement, débutant de la racine et passant par tous les sous-répertoires requis pour atteindre le fichier visé.

Le *chemin relatif* d'un fichier est l'adresse de son emplacement, écrite à partir d'un emplacement de l'arborescence que l'on appelle *répertoire courant* (en anglais : *current working directory*). Ce répertoire courant est initialisé par défaut à un point prédéterminé de l'arborescence (répertoire « maison », ou *home*), mais peut ensuite être modifié.

Q 17 : Que fait précisément la commande `dir` ?

La commande `cd`, avec un argument, permet de changer de répertoire courant. Sans argument, elle permet d'afficher le répertoire courant. En particulier, la commande `cd` *d*,où *d* est le nom absolu ou relatif d'un répertoire, change le répertoire courant en *d*. Essayez avec par exemple.

Q 18 : Changer le répertoire courant. Que fait `cd %UserProfile%` ? De quoi `%UserProfile%` est-il l'abréviation ? Quel est le chemin absolu du répertoire `%UserProfile%` ?

Q 19 : Que fait la commande `mkdir` ? En utilisant une ligne de commande, créer dans le répertoire `%UserProfile%` un sous-répertoire nommé `TP02`. Qu'observe-t-on en exécutant la commande `dir %UserProfile%` ?

Q 20 : Après avoir éxécuter la commande `cd TP02` à partir du répertoire ~, qu'affiche la commande `cd` ?

Copier le dossier `cpython-4243df51fe43` depuis le dossier `groupes/mpsX/TP02` dans le répertoire `%UserProfile%\TP02` en passant par un explorateur de fichiers.

Pour Unix : dans un répertoire, les fichiers et répertoires dont le nom commence par un point sont dits *cachés*. Ce n'est pas la même chose sous Windows.

Q 21 : En consultant le manuel de `dir`, trouver la commande qui permet d'afficher les fichiers et répertoires cachés.

Q 22 : Faire un inventaire des fichiers de `%UserProfile%\TP02`? Y en a-t-il des cachés?

Q 23 : Que désignent les répertoires `.` et `..`? Que donne un `cd` sur chacun de ces répertoires?

Q 24 : Changez le répertoire courant en `%UserProfile%\TP02\cpython-4243df51fe43\Lib\test\capath`. Que fait alors `cd ..\..\multiprocessing`?

Q 25 : Comment obtenir grâce à la commande `dir` et l'option `-l` la taille de tous les fichiers de `TP02\cpython-4243df51fe43\` en les triant par ordre croissant de taille?

1.6 Prise en main élémentaire de `Python`.

Lancer IDLE (via un terminal, n'oubliez pas l'autocomplétion, ou le menu). Un *interpréteur de commandes*, ou `shell`, s'affiche. Le symbole >>> signifie que `Python` attend vos instructions.

Sitôt une instruction tapée et validée (par la touche « Entrée »), le `shell` effectue le calcul demandé puis affiche un résultat, ou un message d'erreur. Il est extrêmement important de bien lire ces messages d'erreur, et de les comprendre!

Q 26 : Taper dans le *shell* les instructions suivantes.

```
x = 42
y = 42.
type(x)
type(y)
x = x+y
x
type(x)
```

Que se passe-t-il? Qu'est-ce que cela signifie?

Q 27 : Décrire ce que font les opérations suivantes, après les avoir étudiées sur des exemples numériques.

```
+    -    *    **    /    //    %
```

Q 28 : Taper dans le *shell* les instructions suivantes.

```
B = 42 > 41.
type(B)
```

Que se passe-t-il? Qu'est-ce que cela signifie?

Q 29 : Décrire ce que font les opérations suivantes, après les avoir étudiées sur des exemples numériques.

```
==    !=    <    >    <=    >=
```

Q 30 : Taper dans le *shell* les instructions suivantes.

```
3/0 > 5
```

Que se passe-t-il? Qu'est-ce que cela signifie?

Q 31 : Taper dans le *shell* les instructions suivantes.

```
B = (42 > 41) or ( 3/0 > 5).
type(B)
```

Que se passe-t-il? Qu'est-ce que cela signifie?

Q 32 : Décrire ce que font les opérations suivantes, après les avoir étudiées sur des exemples logiques.

```
or    and    not
```

Q 33 : Taper dans le *shell* les instructions suivantes.

```
x = -3
abs(x)
help(abs)
```

Que se passe-t-il? Qu'est-ce que cela signifie?

Q 34 : Taper dans le *shell* les instructions suivantes.

```
import math as m
import numpy as np
m.sin(m.pi)
np.sin(np.pi)
np.sin([0,np.pi])
m.sin([0,m.pi])
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Q 35 : Taper dans le *shell* les instructions suivantes.

```
L = [0,1,2,3,4,5,6]
type(L)
L[0]
L[6]
L[-1]
L[-2]
L[7]
L[1:4]
L[2:8]
L.append(7)
L
L = L.append(8)
L
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Nous avons vu comment utiliser des fonctions et des bibliothèques. Nous pouvons bien entendu créer nos propres fonctions (et bibliothèques).

Dans IDLE, ouvrir un nouveau fichier (CTRL+N ou File / New file). L'enregistrer (CTRL + S ou File / Save) sous le nom `TP02.py`.

Q 36 : Taper dans cette fenêtre le script suivant.

```
"""TP n°02"""

def somme(n) :
    """Renvoie 0 + 1 + 2 + ... + n
    Précondition : n entier naturel"""
    return n*(n+1) // 2
```

Enregistrer puis appuyer sur la touche F5 ou (Run / Run Module). Le *shell* doit s'afficher. Taper dans le *shell* les instructions suivantes.

```
somme(42)
somme(42.)
somme(-1515)
help(somme)
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Q 37 : Comment peut-on utiliser la fonction écrite précédemment dans un *autre* script `Python` ?