Langages de script (Python)
CMTP n° 5 : Module `pathlib`

Plusieurs outils sont fournis dans la bibliothèque standard de Python pour accéder au système de fichiers. Ici nous allons principalement utiliser le module `pathlib` qui permet de manipuler facilement les chemins (quel que soit le système d'exploitation [1]). Nous utiliserons aussi parfois le module `shutil` qui propose des opérations de haut niveau sur les fichiers.

La présentation de `pathlib` se trouve ici :
https://docs.python.org/3/library/pathlib.html

Dans les exemples ci-dessous, on verra quelques façons de créer des chemins, de les composer et d'accéder aux différentes parties d'un chemin.

```
>>> from pathlib import Path
>>> Path()
PosixPath('.')
>>> PurePath('toto/bip/aux.pdf')
PurePosixPath('toto/bip/aux.pdf')
>>> Path.home()
PosixPath('/Users/bob')
>>> Path.cwd()
PosixPath('/Users/bob/Cours/L3-LS6/Cours/Cmtp-5')
>>> p=Path() / 'toto' / 'bip' / 'aux.pdf'
>>> p
PosixPath('toto/bip/aux.pdf')
>>> p.name
'aux.pdf'
>>> p.stem
'aux'
>>> p.suffix
'.pdf'
>>> p.parent
PosixPath('toto/bip')
>>> p.parent.parent
PosixPath('toto')
>>> p.parents[0]
PosixPath('toto/bip')
>>> p.parents[1]
PosixPath('toto')
```

On dispose d'une méthode `mkdir` qui permet de créer un nouveau dossier (deux options importantes à comprendre : `parents=True` et `exist_ok=True`). Pour lister le contenu du dossier désigné par `p`, on dispose d'un itérateur `iterdir()` :

```
>>> p=Path() / 'toto'
>>> for x in p.iterdir() :
...   print(x)
...
toto/bip
toto/cmtp4.out
```

On peut aussi parcourir une arborescence de fichiers avec la méthode `glob` (pour laquelle la double étoile `**` signifie "le dossier courant et tous ses sous-dossiers") :

1. Une autre possibilité consiste à utiliser les outils du module du `os.path` mais cette approche est moins pratique que `pathlib`

```
>>> p=Path() / 'toto'
>>> for x in p.glob('**/*') :
...   print(x)
...
toto/bip
toto/cmtp4.out
toto/bip/toto
toto/bip/graphe1
toto/bip/titi
toto/bip/cmtp4.out
toto/bip/toto/ert
toto/bip/toto/cmtp4.out
toto/bip/titi/cmtp4.log
toto/bip/titi/cmtp4.aux
toto/bip/titi/cmtp4.pdf
```

La manipulation de chemin se fait de manière simple, il faut lire attentivement les attributs disponibles et les méthodes de l'objet `Path` présenté dans la page web donnée précédemment.

Exercice 1 :

Ecrire les fonctions suivantes :

1. `listfichNomC(p,nom)` qui liste tous les fichiers qui se trouvent dans l'arborescence de racine `p` et dont le nom complet est `nom`.
2. `listfichNomS(p,nom)` qui liste tous les fichiers qui se trouvent dans l'arborescence de racine `p` et dont le nom sans le suffixe est `nom`.
3. `listfich(p,s)` qui affiche tous les fichiers (et uniquement eux) dont le nom commence par `s` situés dans le dossier désigné par le chemin `p` (on prendra le dossier courant et la chaîne vide en l'absence d'argument). Ecrire une version `listfichr` qui recherche aussi les fichiers dans les sous-dossiers de `p`.

Exercice 2 :

On dispose d'une liste `L` de n chaines de caractères et on souhaite créer un dossier `ARCHIVE` dans le dossier courant contenant un fichier `fich_0.txt` contenant la première chaine de `L`, un fichier `fich_1.txt` contenant la seconde chaine de `L`, ... un fichier `fich_n-1.txt` contenant la dernière chaine de `L`. Ecrire une fonction `Sauvegarde(L)` pour le faire.
(*Suggestion : on pourra utiliser* `Path.write_text`)

Exercice 3 :

Écrire une fonction `listfichNiv(p,suf)` qui liste tous les fichiers de suffixe `suf` situés à une profondeur d'au plus 3 depuis le dossier `p`.

Exercice 4 : Modifications récentes

Écrire une fonction `recent_files(p)` qui prend en entrée un chemin `p` désignant un répertoire et affiche la liste des dix (s'il y en a) fichiers de ce répertoire qui ont été modifiés le plus récemment.
Même question mais on cherchera les fichiers y compris dans les sous-répertoires de `p`. On appellera cette fonction `recent_files_r(p)`
(*Suggestion : on pourra utiliser* `Path.stat`)

Exercice 5 :

Écrire une fonction `AncetreC` qui étant donné deux chemins `p1` et `p2` renvoie un chemin vers le dossier `p` le plus proche de `p1` et `p2` et tel que les deux soient situés dans l'arborescence de racine `p`.

Exercice 6 :

Écrire une fonction `listeGF(p,k)` qui liste tous les fichiers dans l'arborescence de racine `p` dont la taille est supérieure à `k` où `k` est une chaine de caractères composée d'un entier suivi du symbole `K`, `M` ou `G`.

Exercice 7 :

Écrire une fonction `rangement(p)` qui déplace tous les fichiers de l'arborescence de racine `p` de manière à ce que les fichiers de taille inférieure à 1K soient dans le dossier `ARCHIVE/F1K`, ceux dont la taille est entre 1K et 1M soient dans `ARCHIVE/F1M` et ceux dont la taille est entre 1M et 100M soient dans `ARCHIVE/F100M` et tous les autres dans `ARCHIVE/Fsup`.