21_python_directory_and_files_mngt

December 15, 2023

1 Python Directory and Files Management

Grâce au module `os` python est capable de manipuler les fichiers et les répertoires.

1.1 Get current directory

Nous pouvons récupérer le dossier courant avec la fonction `getcwd()` du module `os`

[ ]:
```python
import os

print(os.getcwd())
```

1.2 Change directory

Nous pouvons changer de dossier avec la fonction `chdir()` du module `os`

[ ]:
```python
import os

# change directory
os.chdir('../../python/')

print(os.getcwd())
```

1.3 List directories and files in a directory

Nous pouvons lister les fichiers et les répertoires d'un dossier avec la fonction `listdir()` du module `os`. Il est possible de renseigner un chemin absolu ou relatif. Si aucun chemin n'est renseigné, le dossier courant est utilisé.

[ ]:
```python
import os

os.chdir('../cours/jupyter/')
print(os.getcwd())

# list all sub-directories
os.listdir()
```

1.4 Making a new directory

Nous pouvons créer un nouveau dossier avec la fonction `mkdir()` du module `os`. Il est possible de renseigner un chemin absolu ou relatif. Si aucun chemin n'est renseigné, le dossier courant est utilisé.

[ ]:
```
if not os.path.exists('_test'):
    os.mkdir('_test')

os.listdir()
```

1.5 rename a file or a directory

Nous pouvons renommer un fichier ou un dossier avec la fonction `rename()` du module `os`. Il est possible de renseigner un chemin absolu ou relatif. Si aucun chemin n'est renseigné, le dossier courant est utilisé.

```
import os

os.rename("oldname", "newname")
```

- `oldname` : le nom du fichier ou du dossier à renommer
- `newname` : le nouveau nom du fichier ou du dossier

  Note:La fonction `rename()` peut aussi être utilisée pour déplacer un fichier ou un dossier.

[ ]:
```
import os

os.listdir()

# rename a directory
os.rename('_test','_test_')

os.listdir()
```

1.6 Remove a file or a directory

Nous pouvons supprimer un fichier ou un dossier avec les fonctions `remove()`, `rmdir()` ou `removedirs()` du module `os`. Il est possible de renseigner un chemin absolu ou relatif. Si aucun chemin n'est renseigné, le dossier courant est utilisé.

```
import os

# remove a file
os.remove("filename")

# remove an empty directory
os.rmdir("dirname")
```

[ ]:
```
import os

# delete "myfile.txt" file
os.rmdir("./_test_")
```

Si on veut supprimer des dossier non vides, il faut utiliser la fonction rmtree() du module `shutil`.

```
import shutil

shutil.rmtree("dirname")
```

> **Note**: `/!\` La suppression est définitive, il n'y a pas de corbeille.

1.7 OS Module Methods & Attributes

Method/Attribute	Description
`os.name`	Gives the name of the operating system dependent module imported. The following names have currently been registered: `posix`, `nt`, `mac`, `os2`, `ce`, `java`, `riscos`.
`os.getcwd()`	Returns the current working directory.
`os.chdir()`	Change the current working directory to the given path.
`os.listdir()`	Returns a list of all files and directories in the specified path.
`os.mkdir()`	Create a directory named path with numeric mode mode. The default mode is 0777 (octal).
`os.makedirs()`	Recursive directory creation function. Like mkdir(), but makes all intermediate-level directories needed to contain the leaf directory.
`os.remove()`	Remove (delete) the file path. If path is a directory, OSError is raised. Use rmdir() to remove directories.
`os.rmdir()`	Remove (delete) the directory path. Only works when the directory is empty, otherwise, OSError is raised. In order to remove whole directory trees, shutil.rmtree() can be used.
`os.removedirs()`	Remove directories recursively. Works like rmdir() except that, if the leaf directory is successfully removed, removedirs() tries to successively remove every parent directory mentioned in path until an error is raised (which is ignored, because it generally means that a parent directory is not empty).

Method/Attribute	Description
`os.rename()`	Rename the file or directory src to dst. If dst is a directory, OSError will be raised. On Unix, if dst exists and is a file, it will be replaced silently if the user has permission. The operation may fail on some Unix flavors if src and dst are on different filesystems. If successful, the renaming will be an atomic operation (this is a POSIX requirement).
`os.renames()`	Recursive directory or file renaming function. Works like rename(), except creation of any intermediate directories needed to make the new pathname good is attempted first. After the rename, directories corresponding to rightmost path segments of the old name will be pruned away using removedirs().
`os.stat()`	Perform a stat system call on the given path. The return value is an object whose attributes correspond to the members of the stat structure, namely: st_mode (protection bits), st_ino (inode number), st_dev (device), st_nlink (number of hard links), st_uid (user ID of owner), st_gid (group ID of owner), st_size (size of file, in bytes), st_atime (time of most recent access), st_mtime (time of most recent content modification), st_ctime (platform dependent; time of most recent metadata change on Unix, or the time of creation on Windows)
`os.system()`	Execute the command (a string) in a subshell. This is implemented by calling the Standard C function system(), and has the same limitations. Changes to sys.stdin, etc. are not reflected in the environment of the executed command. If command generates any output, it will be sent to the interpreter standard output stream.
`os.environ`	A mapping object representing the string environment. For example, environ['HOME'] is the pathname of your home directory (on some platforms), and is equivalent to getenv("HOME") in C.
`os.path`	The path module suitable for the operating system Python is running on, for example, posix, nt, or os2. If you want to import the correct path module using import os.path, you need to use the code below instead.

Method/Attribute	Description
`os.path.abspath(path)`	Return a normalized absolutized version of the pathname path. On most platforms, this is equivalent to calling the function normpath() as follows: normpath(join(os.getcwd(), path)).
`os.path.basename(path)`	Return the base name of pathname path. This is the second element of the pair returned by passing path to the function split(). Note that the result of this function is different from the Unix basename program; where basename for '/foo/bar/' returns 'bar', the basename() function returns an empty string (' ').

etc…