

Séance 9

Manipulation de documents CSV, XML, PDF...



Ce(tte) œuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution – Pas d'Utilisation Commerciale – Pas de Modification 4.0 International.

Objectifs

- Manipuler le **système de fichiers**
 - Création/modification/suppression de fichiers et répertoires
 - Chemin d'un fichier/répertoire
- **Lire et éditer** des documents spécifiques
 - ZIP, CSV, XML, images...



Système de fichiers

Système de fichiers

- Façon de **stocker les informations** et les organiser en fichiers

Stockage sur mémoire secondaire (HDD, SSD, DVD, clé USB...)

- Plusieurs **systèmes de fichiers** existants

FAT32, HFS, NTFS, ext4, ReiserFS...

- Plusieurs **caractéristiques** aux fichiers

- Nom et chemin du fichier
- Taille et type du fichier
- Permissions
- Autres métadonnées

Manipuler le système de fichiers

- **Trois modules** pour manipuler le système de fichiers
 - `os` : fonctions d'interface avec le système d'exploitation
 - `os.path` : manipulation de chemins dans le système de fichiers
 - `shutil` : opérations sur des fichiers et répertoires

```
1 import os
2
3 print('Votre OS :', os.name)
4 print('Répertoire courant :', os.getcwd())
```

```
Votre OS : posix
Répertoire courant : /Users/combefis/Desktop
```

Module os

- Certaines fonctions dépendent du **système d'exploitation**

Unix, Windows, certains Unix, les Unix récents...

- Plusieurs fonctions pour **obtenir des informations**

- `getcwd()` : renvoie le répertoire courant
- `chdir(path)` : change le répertoire courant

```
1 import os
2
3 print(os.getcwd())
4 os.chdir('/Users')
5 print(os.getcwd())
```

```
/Users/combefis/Desktop
/Users
```

Répertoire (1)

- Fonctions de manipulation de **répertoires**
 - `mkdir(path)` : crée un nouveau répertoire
 - `rmdir(path)` : supprime un répertoire (doit être vide)
 - `listdir(path)` : renvoie la liste du contenu d'un répertoire
 - `rename(src,dst)` : renomme un répertoire

```
1 import os
2
3 os.chdir('/tmp')
4 os.mkdir('test')
5 os.mkdir('test/dota')
6
7 os.rename('test/dota', 'test/data')
8 print(os.listdir('/tmp/test'))
```

```
['data']
```


Répertoire (2)

- **Erreur** lorsque les dossiers intermédiaires n'existent pas

Les erreurs sont de type `OSError`

- Fonctions qui gèrent les **répertoires intermédiaires**
 - `makedirs(path)` : crée un nouveau répertoire
 - `removedirs(path)` : supprime un répertoire (que les vides)
 - `renames(src,dst)` : renomme un répertoire

```
1 import os
2
3 os.chdir('/tmp')
4 os.makedirs('test/dota')
5
6 os.rename('test/dota', 'test/data')
7 print(os.listdir('/tmp/test'))
```

Fichier

- Fonctions de manipulation de **fichiers**
 - `remove(path)` : supprime un fichier
 - `rename(src,dst)` : renomme un fichier
- Création d'un **fichier vide** avec `open`

```
1 import os
2
3 os.chdir('/tmp/test/data')
4 with open('log.txt', 'w') as f1, open('main.cmd', 'w') as f2:
5     pass
6
7 os.remove('log.txt')
8 with open('/tmp/test/log.txt', 'w') as file:
9     pass
```

Parcourir un répertoire

■ Balade dans un répertoire avec `os.walk`

Renvoie une séquence de triplets (racine, répertoires, fichiers)

```
1 import os
2
3 for root, dirs, files in os.walk('/tmp/test'):
4     print(root)
5     print('\t', dirs)
6     print('\t', files)
```

```
/tmp/test
    ['data']
    ['log.txt']
/tmp/test/data
    []
    ['main.cmd']
```

Informations sur un fichier/répertoire

- Utilisation de la **fonction stat** pour obtenir des infos

Fonctionne aussi bien pour un fichier qu'un répertoire

- Renvoie un objet avec **plusieurs champs**

- `st_size` : taille
- `st_atime`, `st_mtime` : dernier accès/modification

```
1 import os
2 import time
3
4 info = os.stat('/tmp/test/data')
5 print(info.st_size)
6 print(time.ctime(info.st_mtime))
```

```
102
Sun Mar 20 16:01:51 2016
```

Variables d'environnement

- **Variable d'environnement** avec `os.environ`

Dictionnaire qu'il est possible de modifier

- Les variables d'environnement **dépendent du système**

Nécessité de tester la présence d'une variable avec `in`

```
1 import os
2
3 print('Répertoire home :', os.environ['HOME'])
4 print('Utilisateur connecté :', os.environ['LOGNAME'])
5 print('Langue :', os.environ['LANG'])
6 print('Répertoire courant :', os.environ['PWD'])
```

```
Répertoire home : /Users/combefis
Utilisateur connecté : combefis
Langue : fr_BE.UTF-8
Répertoire courant : /Users/combefis/Desktop
```

Variable d'environnement PATH

- Répertoires où chercher un **programme à lancer**

Séquence de répertoires à inspecter

- Stockés dans la **variable d'environnement PATH**

Utilisation d'un séparateur pour les répertoires

```
1 import os
2
3 print(os.environ['PATH'])
```

```
/usr/local/bin:/usr/local/sbin:/usr/bin:/bin:/usr/sbin:/sbin:usr/
local/go/bin
```

Informations propre au système

- Variables **spécifiques au système** dans le module `os`
 - `os.sep` : séparateur entre répertoires dans les chemins
 - `os.pathsep` : séparateur de la variable d'environnement `PATH`
 - `os.linesep` : séparateur des lignes dans un fichier texte

```
1 import os
2
3 print(os.sep)
4 print(os.pathsep)
5 print(os.linesep == '\n')
```

```
/
:
True
```

Module `os.path`

- Fonctions pour manipuler des **chemins** de fichier/répertoire

Opérations spécifiques au système (`posixpath`, `ntpath`, `macpath`)

- Construction d'un **chemin** spécifique

- `os.path.join(*paths)` : construit un chemin avec plusieurs
- `os.path.split(path)` : découpe un chemin en deux

```
1 import os.path
2
3 p = os.path.join('/tmp/test', 'data', 'main.cmd')
4 print(p)
5 print(os.path.split(p))
```

```
/tmp/test/data/main.cmd
('/tmp/test/data', 'main.cmd')
```


Chemin relatif et absolu (1)

- **Deux façons** de décrire le chemin d'un fichier/répertoire
 - Chemin décrit à partir de la racine (absolu)
 - Chemin décrit à partir du répertoire courant (relatif)
- **Conversion** de description de chemin
 - `os.path.abspath(path)` : obtient le chemin absolu
 - `os.path.relpath(path)` : obtient le chemin relatif

```
1 import os.path
2
3 print(os.path.relpath('/tmp/test/data/main.cmd'))
4 print(os.path.abspath('data.txt'))
```

```
../../../../tmp/test/data/main.cmd
/Users/combefis/Desktop/data.txt
```

Chemin relatif et absolu (2)

- Deux **dossiers spéciaux** en mode relatif
 - `.` représente le répertoire courant
 - `..` représente répertoire parent

```
1 import os.path
2
3 print(os.path.abspath('.'))
4 print(os.path.abspath('..'))
5 print(os.path.abspath('../ ../../../../combefis'))
```

```
/Users/combefis/Desktop
/Users/combefis
/Users/combefis
```

Parties d'un chemin

- Deux **dossiers spéciaux** en mode relatif
 - `os.path.basename(path)` : partie de base du chemin
 - `os.path.dirname(path)` répertoire du chemin
- Extraction du **nom et de l'extension** d'un fichier
 - `os.path.splitext(path)` : renvoie le nom et l'extension

```
1 import os.path
2
3 p = '/tmp/test/log.txt'
4 print(os.path.basename(p))
5 print(os.path.dirname(p))
6 print(os.path.splitext(p))
```

```
log.txt
/tmp/test
('/tmp/test/log', '.txt')
```

Existence et type

- Tester le **type** d'un chemin
 - `os.path.isfile(path)` : teste si le chemin est un fichier
 - `os.path.isdir(path)` teste si le chemin est un dossier
- Tester l'**existence** d'un chemin
 - `os.path.exists(path)` : teste si le chemin existe

```
1 import os.path
2
3 p = '/tmp/test/log.txt'
4 if os.path.exists(p):
5     print(os.path.isfile(p))
```

True

Module shutil

- Opérations de **haut-niveau** sur les fichiers et répertoires

Souvent utilisé pour opérations sur ensemble de fichiers

- Attention, ne copie pas forcément toutes les **méta-données** !
 - `shutil.copyfile(src, dst)` : copie un fichier
 - `shutil.copy(src, dst)` : copie un fichier dans un répertoire
 - `shutil.copytree(src, dst)` : copie un répertoire
 - `shutil.rmtree(src, dst)` : supprime un répertoire
 - `shutil.move(src, dst)` : déplace un fichier/répertoire



Manipulation de documents

Manipulation de documents CSV (1)

- Un **enregistrement par ligne**, valeurs séparées par des virgules

Entête de la table sur la première ligne

- **Séparateur décimal** est par défaut le point

Convention anglaise de notation des nombres

- Peu utilisé car **non standardisé**

```
1 Nom,Prix,Code
2 Banane,5.99,77
3 Pomme,2.99,99
4 Poire,7.99,170
```

Manipulation de documents CSV (2)

- Le **point-virgule** comme séparateur au lieu de la virgule
- Encadrer les valeurs avec des **guillemets doubles** "

```
1 Titre;Auteurs;Éditeur;Prix
2 NoSQL;Rudi;Eyrolles;17,99
3 SQL;"Ronald;Ryan;Arie";Pearson;21,99
4 "Sabina, "la juive" de Jung";Alain;PGdR Editions;10,50
```

Titre	Auteur	Éditeur	Prix
NoSQL	Rudi	Eyrolles	17,99 €
SQL	Ronald ; Ryan ; Arie	Pearson	21,99 €
Sabina, "la juive" de Jung	Alain	PGdR Editions	10,50 €

Module csv

- Lecture avec un `csvreader` en spécifiant le délimiteur

Lecture des lignes avec `for` qui renvoie des listes de `str`

```
1 import csv
2
3 src = 'cart.csv'
4 with open(src, 'r') as file:
5     csvreader = csv.reader(file, delimiter=',')
6     next(csvreader) # Ignore header line
7     totalprice = 0
8     for line in csvreader:
9         totalprice += float(line[3].replace(',','.'))
10    print('Prix total :', totalprice, 'euros')
```

Prix total : 50.48 euros

Écrire un fichier CSV

- Écriture avec un `csvwriter` avec `,` comme délimiteur

Écriture des lignes une à la fois en fournissant une liste de `str`

```
1 import csv
2
3 with open('result.csv', 'w') as file:
4     csvwriter = csv.writer(file)
5     csvwriter.writerow(['Prénom', 'Nom', 'Catégorie'])
6     csvwriter.writerow(['Victor', 'Puissant Baeyens', 'T3'])
7     csvwriter.writerow(['Amaury', 'Lekens', 'V2'])
```

```
Prénom,Nom,Catégorie
Victor,Puissant Baeyens,T3
Amaury,Lekens,V2
```

Manipulation de documents XML

- **eXtensible Markup Language** (XML) est un langage balisé

Représentation d'information structurée (comme JSON)

- Deux manières de **parcourir** un fichier XML

- De manière événementielle avec SAX
- En chargeant tout l'arbre avec DOM

```
1 <library>
2   <book title="NoSQL" editor="Eyrolles" price="17.99">
3     <author name="Rudi"/>
4   </book>
5   <book title="SQL" editor="Pearson" price="21.99">
6     <author name="Ronald"/>
7     <author name="Ryan"/>
8     <author name="Arie"/>
9   </book>
10  <book title="Sabina, &#34;la juive&#34; de Jung" editor="PGdR Editions"
11    price="10.50">
12    <author name="Alain"/>
13  </book>
</library>
```

Parcours avec SAX (1)

■ Définition d'une classe pour gérer les événements SAX

Méthodes à l'entrée/sortie d'une balise, et lecture d'un texte

```
1 class LibraryHandler(xml.sax.ContentHandler):
2     def __init__(self):
3         self.__total = 0
4
5     @property
6     def total(self):
7         return self.__total
8
9     def startElement(self, tag, attributes):
10        if tag == 'book':
11            self.__total += float(attributes['price'])
12
13    def endElement(self, tag):
14        pass
15
16    def characters(self, content):
17        pass
```

Parcours avec SAX (2)

- Création d'un **parser SAX** et d'un gestionnaire

Association du gestionnaire au parser

- **Lancement de l'analyse** du fichier XML avec méthode `parse`

Ensuite récupération du résultat via le gestionnaire

```
1 parser = xml.sax.make_parser()
2 handler = LibraryHandler()
3 parser.setContentHandler(handler)
4
5 src = 'cart.xml'
6 parser.parse(src)
7 print('Prix total :', handler.total, 'euros')
```

Parcours avec DOM

- Chargement de l'**arbre complet** du document

Navigation ensuite possible en recherchant dans l'arbre

- **Trois accès** différents

- Récupération de l'élément racine avec documentElement
- Recherche des fils de type book
- Récupération de l'attribut price des livres

```
1  src = 'cart.xml'
2  doc = xml.dom.minidom.parse('cart.xml')
3  library = doc.documentElement
4  totalprice = 0
5  for book in library.getElementsByTagName('book'):
6      totalprice += float(book.getAttribute('price'))
7  print('Prix total :', totalprice, 'euros')
```

Manipulation de documents XLSX

- **Document Excel** (XLSX) représente des tableurs

Contient principalement des données et des formules

- Fichier compressé contenant des **ressources** (XML)

Il s'agit essentiellement d'une simple archive ZIP

- **Structure** d'un document XLSX

- Un document excel est appelé un workbook
- Possède plusieurs worksheets (feuille)
- Chaque feuille est découpée en lignes et colonnes

Créer un document XLSX

- Création d'un **Workbook vide** puis remplissage

Possibilité d'ajouter des formules dans les cellules

```
1 import openpyxl
2
3 wb = openpyxl.Workbook()
4 sheet = wb.active
5 sheet.title = 'June 2016'
6
7 sheet['B1'] = 'Labo'
8 sheet['C1'] = 'Examen'
9 sheet['D1'] = 'Moyenne'
10
11 sheet['A2'] = 'Tom'
12 sheet['B2'] = '18'
13 sheet['C2'] = '11'
14 sheet['D2'] = '=0.3*B2+0.7*C2'
15
16 print(sheet['D2'].value)
17
18 wb.save('results.xlsx')
```


Lire un document XLSX

- Ouverture d'un **Workbook** en mode brut ou données

L'option `data_only=True` permet de calculer les formules

```
1 wb = openpyxl.load_workbook('results.xlsx', data_only=True)
2 sheet = wb.active
3
4 for i in range(1, 5):
5     cell = sheet.cell(row=1, column=i)
6     print(cell.value, end=' ')
7
8 cell = sheet['D2']
9 print('\nCell {}{} : {}'.format(cell.column, cell.row, cell.value))
```

```
None Labo Examen Moyenne
Cell D2 : None
```

Manipulation de documents PDF

- **Portable Document Format** (PDF) représente des textes

Fichier au format binaire, pas manipulable directement

- Contient **plusieurs éléments**

- Du texte
- Des polices de caractères
- Des images

- **Extraction** du texte d'un document PDF

Aussi bien que possible, extraction pas parfaite

- **Création** d'un nouveau PDF à partir d'un existant

Récupération de page, rotation, recouvrement

```
1 import PyPDF2
2
3 with open('thinkpython.pdf', 'rb') as file:
4     pdfreader = PyPDF2.PdfFileReader(file)
5     pdfwriter = PyPDF2.PdfFileWriter()
6     for i in range(min(10, pdfreader.numPages)):
7         pdfwriter.addPage(pdfreader.getPage(i))
8     with open('summary.pdf', 'wb') as output:
9         pdfwriter.write(output)
```

Ajout d'un filigrane *Top Secret*

```
1 import os.path
2 import sys
3
4 import PyPDF2
5
6 if __name__ == '__main__' and len(sys.argv) == 2:
7     src = sys.argv[1]
8     if os.path.exists(src):
9         # Adds a 'Top Secret' watermark to all the pages
10        with open(src, 'rb') as file1, open('topsecret.pdf', 'rb')
11        as file2:
12            pdfreader = PyPDF2.PdfFileReader(file1)
13            pdfwriter = PyPDF2.PdfFileWriter()
14            for i in range(pdfreader.numPages):
15                watermark = PyPDF2.PdfFileReader(file2).getPage(0)
16                page = pdfreader.getPage(i)
17                watermark.mergePage(page)
18                pdfwriter.addPage(watermark)
19            # Writes the result file
20            name, ext = os.path.splitext(src)
21            dst = os.path.join(os.path.dirname(src), '{}_secret.{}'.
22                               .format(name, ext))
23            with open(dst, 'wb') as output:
24                pdfwriter.write(output)
```

Librairie PyFPDF

- Possibilité de **créer un document** PDF

Similaire à une librairie de dessin des éléments à insérer

- **Trois étapes** à suivre
 - Ajout d'une page dans le document
 - Définition du style à appliquer
 - Ajout des éléments dans la page (texte, image...)

```
1 import fpdf
2
3 doc = fpdf.FPDF()
4 doc.add_page()
5 doc.set_font('Arial', 'B', 28)
6 doc.cell(40, 10, 'Hello Seb!')
7 doc.output('result.pdf', 'F')
```

Manipulation de documents DOCX

- **Document Word** (DOCX) représente des documents texte
Contient principalement du texte avec du style et des images
- Fichier compressé contenant des **ressources** (XML)
Il s'agit essentiellement d'une simple archive ZIP
- **Structure** d'un document DOCX
 - Un document est constitué de *paragraph*
 - Un paragraphe est composé de *runs*

Créer un document DOCX

- Ajout de paragraphe avec la méthode `add_paragraph`

Et ajout de runs dans le paragraphe avec `add_run`

```
1 import docx
2
3 doc = docx.Document()
4
5 doc.add_paragraph('Mon premier document', 'Title')
6
7 para = doc.add_paragraph('Hello')
8 doc.add_paragraph('By Combéfis & Lurkin, we love Word!')
9 para.add_run('World!')
10
11 doc.save('helloworld.docx')
```

Manipulation d'images

■ Utilisation de la librairie **Pillow**

Chargement, obtention d'information et modification d'images

■ **Découpe sous-image** avec `crop(left, upper, right, lower)`

```
1 from PIL import Image
2
3 image = Image.open('cute-cat.jpg')
4 cropped = image.crop((100, 50, 200, 150))
5 cropped.save('cropped-cat.jpg')
```



Création d'une mosaïque

- Coller des images sur une autre avec paste

Découpe d'une sous-image pour création d'une mosaïque

```
1 from PIL import Image
2
3 image = Image.open('cute-cat.jpg')
4 cropped = image.crop((100, 50, 200, 150))
5 mosaic = Image.new('RGBA', (200, 200))
6 for (i, j) in [(i, j) for i in range(2) for j in range(2)]:
7     mosaic.paste(cropped, (i * 100, j * 100))
8 mosaic.save('mosaic-cat.jpg')
```



Manipulation de documents ZIP (1)

- Possibilité de créer un **fichier ZIP** et de naviguer dedans

Extraction d'un fichier depuis le ZIP

- Création d'un **objet ZipFile** avec l'instruction `with`

Ajout de fichiers dans l'archive avec `write`

```
1 import os.path
2 import zipfile
3
4 with zipfile.ZipFile('data.zip', 'w') as file:
5     file.write('data.txt')
6     file.write('cart.xml')
7     file.write('cart.csv')
```

Manipulation de documents ZIP (2)

- Obtention du **répertoire home** avec `os.path.expanduser`

Transforme le ~ en le répertoire de l'utilisateur courant

```
1 file = zipfile.ZipFile('data.zip')
2 file.printdir()
3 print(file.getinfo('data.txt'))
4
5 file.extractall(os.path.expanduser("~"))
```

File Name	Modified	Size
data.txt	2016-03-20 12:28:28	18
cart.xml	2016-03-20 22:12:12	411
cart.csv	2016-03-20 19:23:06	147

<ZipInfo filename='data.txt' filemode='-rwxr-xr-x' file_size=18>

Crédits

- <https://www.flickr.com/photos/kleuske/8366881617>
- <https://www.flickr.com/photos/58579259@N06/5602206159>
- <http://i.imgur.com/gbwgfw6.jpg>