Manipuler des PDF en Python

CourtBouillon – Python AFPy – 25 novembre 2021

Salut ! Je m’appelle Guillaume Ayoub.

Je fais partie de CourtBouillon, des gens authentiques qui font pousser du code libre.

Notre projet principal s’appelle WeasyPrint. Il transforme du HTML et du CSS en documents imprimables.

1. Comment fonctionne un PDF
2. Extraire les informations d’un PDF
3. Afficher un PDF
4. Modifier un PDF
5. Créer un PDF

Nous parlerons majoritairement des outils écrits en Python.

Un très grand nombre de bibliothèques existent, mais leurs niveaux de qualité et de maintenance sont très aléatoires.

Nous ne verrons pas tout !

Comment fonctionne un PDF

Rassurez-vous, on ne va pas s’infliger les 750 pages de spécification…

Un format « ouvert »

PDF est un format créé en 1992 par Adobe.

D’abord propriétaire, il devient officiellement ouvert en 2008 avec la publication d’un standard ISO accessible librement et l’attribution de licences gratuites pour les brevets d’Adobe relatifs au standard.

La version 2.0 du format est débarrassée de ses éléments propriétaires. Il est désormais la propriété de l’ISO : n’importe qui peut participer à son élaboration, mais il est payant.

Un format « lisible »

On peut ouvrir un PDF avec un éditeur de texte.

Un PDF contient une liste d'objets, représentant les différents contenus, et une table des objets à la fin, donnant la position de chacun des objets dans le fichier.

Les objets contiennent souvent du texte, mais ils peuvent être compressés (et donc illisibles).

Un format « hackable »

Modifier un objet est complexe, mais totalement possible.

Le plus simple reste d’ajouter de nouveaux objets et de les ajouter à la fin, avec une nouvelle table.

Début de fichier

```
%PDF-1.7

1 0 obj
<<
/Type /Pages
/Kids [ 6 0 R 8 0 R … ]
/Count 5
>>
endobj

2 0 obj
<<
…
```

Fin de fichier

```
xref
0 21
0000000000 65535 f
0000000015 00000 n
…
0000011672 00000 n
trailer
<<
/Size 21
/Root 3 0 R
>>
startxref
11709
%%EOF
```

Extraire des informations d’un PDF

Dites-moi ce qu’il y a là-dedans !

Récupérer de nombreuses informations

On peut vouloir récupérer les informations d’un PDF, que ce soit les métadonnées ou le contenu : texte, images…

Récupérer le texte

```
from pdfminer.high_level import extract_text

text = extract_text('document.pdf')
print(text)
```

Récupérer des métadonnées

```
from PyPDF2 import PdfFileReader

with open('document.pdf', 'rb') as fd:
    pdf = PdfFileReader(fd)
    information = pdf.getDocumentInfo()

print(information.author)
print(information.producer)
print(information.title)
```

Afficher un PDF

Sur un écran, ou sur du papier si l’imprimante le veut bien…

C’est compliqué

Rendre un document à l’écran ou sur une page de papier est très difficile. La norme PDF fait environ 750 pages, et il faut idéalement tout gérer.

Il n’existe pas d’outil en Python pour faire cela. Les implémentations libres connues sont Ghostscript et Poppler (basé sur xpdf). Il existe des *bindings* Python.

Transformer un PDF en image

```
from poppler import PageRenderer, load_from_file

pdf_document = load_from_file('document.pdf')
page_1 = pdf_document.create_page(0)
renderer = PageRenderer()
image = renderer.render_page(page_1)
image.save('page_1.png', 'png')
```

Afficher un PDF sur l'écran

```
from os import system

system('gs -sDevice=x11 -dBATCH document.pdf')
```

Oui, il existe un *binding* Python pour Ghostscript, mais il est vraiment « limité ».

Modifier un PDF

On coupe, on tourne, on réorganise…

Modifier, c’est comprendre la structure

Certaines bibliothèques sont capables de lire le contenu d’un PDF, de fournir une interface objet Python correspondant aux objets du PDF, et de générer un nouveau PDF à partir de la structure Python.

Généralement, les objets de base de PDF ont une correspondance avec les objets Python. Il vaut mieux connaître la spécification PDF (ou lire les exemples de la bibliothèque) pour pouvoir faire ce que l’on veut.

Appliquer une rotation sur les pages d’un PDF

```
from pdfrw import PdfReader, PdfWriter

trailer = PdfReader('document.pdf')
for page_number in range(len(trailer.pages)):
    trailer.pages[page_number].Rotate = 90

writer = PdfWriter('document-rotate.pdf')
writer.trailer = trailer
writer.write()
```

Fusionner deux PDF

```
from PyPDF2 import PdfFileReader, PdfFileWriter

pdf_writer = PdfFileWriter()
for path in ['document_1.pdf', 'document_2.pdf']:
    pdf_reader = PdfFileReader(path)
    for page in range(pdf_reader.getNumPages()):
        pdf_writer.addPage(pdf_reader.getPage(page))

with open('document_merge.pdf', 'wb') as fd:
    pdf_writer.write(fd)
```

Créer un PDF

Il est l’heure de réveiller l’artiste en moi !

On a beaucoup de cas d’usage différents

Créer un document nécessite l’utilisation d’outils très variés selon le but précis de la création et les contraintes que l’on a.

Il est important de se poser les bonnes questions et de bien réfléchir afin de déterminer comment procéder.

Créer un PDF manuellement

C’est adapté lorsque l’on connaît bien la spécification ou que l’on veut une interface minimaliste.

Attention : ça pique !

Dessiner un rectangle

```
import pydyf
from pathlib import Path

document = pydyf.PDF()
draw = pydyf.Stream()
draw.rectangle(100, 100, 50, 70)
draw.stroke()
document.add_page(pydyf.Dictionary({
    'Type': '/Page',
    'Parent': document.pages.reference,
    'Contents': draw.reference,
    'MediaBox': pydyf.Array([0, 0, 200, 200])}))

with open('document.pdf', 'wb') as fd:
    document.write(fd)
```

Créer un PDF avec une interface haut niveau

C’est adapté lorsque l’on veut avoir beaucoup de liberté, sans avoir à étudier la spécification.

On peut faire la logique de mise en page en pur Python, mais on est lié à une bibliothèque.

C’est une solution simple lorsque l’on a une mise en page simple.
Lorsque la mise en page est compliquée, c’est… compliqué.
Vraiment très compliqué.

Écrire du texte avec borb

```
from borb.pdf.canvas.layout.page_layout.multi_column_layout
from borb.pdf.canvas.layout.text.paragraph import Paragraph
from borb.pdf.document import Document
from borb.pdf.page.page import Page
from borb.pdf.pdf import PDF

pdf = Document()
page = Page()
pdf.append_page(page)
layout = SingleColumnLayout(page)
layout.add(Paragraph('Hello World!'))
with open('document.pdf'), 'wb') as fd:
    PDF.dumps(fd, pdf)
```

Écrire du texte avec ReportLab

```
from reportlab.pdfgen.canvas import Canvas

canvas = Canvas('document.pdf')
canvas.drawString(100, 100, 'Hello World!')
canvas.showPage()
canvas.save()
```

Créer un PDF à partir de HTML et CSS

C’est adapté lorsque l’on a de la mise en page complexe, et une bonne connaissance de HTML et CSS.

On n’est pas lié à un outil.

Le contenu est stockable dans un format stable, facile à mettre en ligne et à convertir.

Écrire du texte avec WeasyPrint

```
from weasyprint import HTML

html = HTML(string='''
    <style>h1 { color: red }</style>
    <h1>The title</h1>
    <p>Content goes here</p>''')
html.write_pdf('document.pdf')
```

C’est le moment de poser des questions !

On peut voir des exemples, on peut discuter des autres outils, on peut parler de vos problématiques…