Report スクリプティング（Python、ReportLab）

目次

Report スクリプティング（Python、ReportLab）

はじめに

スクリプトで Web、DB にアクセスができるようになり、スマートホンやタブレットからでも手軽に情報をとりだすことができるようになりました。これらの場合、表示するフォントや改行等の体裁はブラウザやツールがローカル環境に合わせて編集するので乱れなく最適な状態で表示されます。

しかし、量が多かったりネットワークに制約がある場所に情報を送るときはレポートの方が向いています。問題はレポートを作成した環境（OS やツール）と表示したい環境が異なる場合で、OS が異なるだけでフォントや改行コードの組合せが異なるため作成した通りには表示されません。この環境によって表示が変わったり文字化けするといった問題は Adobe 社が作った PDF 仕様（現在は ISO 32000）と AJ1 フォント（ファイルに埋め込まない、商用可[1]）を使うと日本語は正しく再現されます。

1. PDF を生成するライブラリ

Python から利用できる PDF 用のライブラリにはいくつかあります。

ReportLab PDF Toolkit (open source)が BSD ライセンス（免責条項を明記すれば再利用、再配布可）で公開しており、企業が有償版のコア部分を無償で開示したものです。Windows のフォントファイルがそのまま使えるので、以降、このライブラリの使用例を紹介します。

https://docs.reportlab.com/

【参考】fpdf2 は ReportLab よりも機能が少ないながら手軽に使えます。Python 用のライブラリは以下のサイトで開発し LGPL-3.0 ライセンス（変更、再配布可）で公開しています[2]。こちらは現状(ver.2.5.6)Windows 10 標準の日本語フォントのファイル形式 ttc（複数 ttf 同梱）が使えないため、ttf 形式の日本語フォントを別途準備する必要があります。

https://github.com/PyFPDF/fpdf2

2. ReportLab のインストール

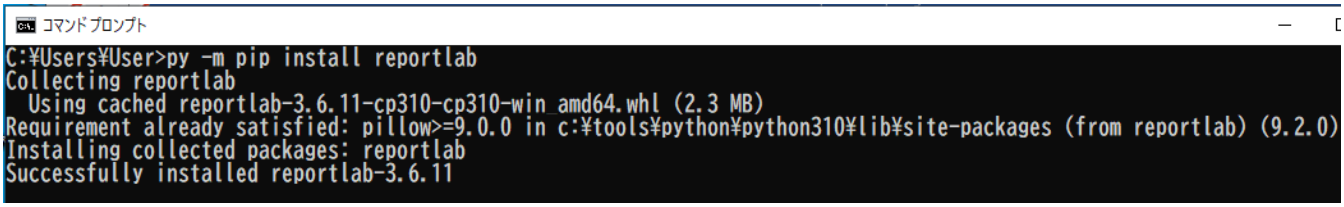
以下 Windows 上で Python 3.10.5 を使った環境で説明します。

Python には pip というパッケージャー（packager）が標準モジュールとして添付されているので、ライブラリの追加はこれを使って行います。コマンドは以下のとおりです。

```
> py -m pip install reportlab
```

※ py は Windows の Python 起動コマンドで、Linux の場合に使う python3 と同義になります

問題なくインストールが終わると以下のように表示されます。



[1] フォントライセンス https://helpx.adobe.com/jp/fonts/using/font-licensing.html#act-client

[2] fpdf2 は PHP 用の fpdf⇒PyFPDF から分岐しています。基になった fpdf はより柔軟なライセンス条項で今でも開発・公開され（ http://www.fpdf.org/ ）日本語化 ver もあります

Report スクリプティング（Python、ReportLab）

3. データベースから PDF を編集するコード例

RDB の 1 テーブルを読みダンプに近い形で全件を PDF に出力します。

（1）入力

SQLite に格納されている以下の形式のデータを使います。

	名前	1月	2月	3月	4月	5月	6月
1	太郎	100	200	300	400	500	600
2	次郎	110	220	330	440	550	660
3	三郎	121	242	363	484	605	726
4	四郎	410	420	330	440	450	460
5	五郎	510	520	330	540	550	560
6	六郎	610	620	330	640	650	660
7	七郎	710	720	330	740	750	760

（2）出力イメージ

以下のコードを実行すると環境変数の TEMP に設定されているディレクトリに出力します。

ファイル名は、“**Report***yyyymmddHHMMSS 一意な文字列*.pdf”（下線部が作成日時）

PAGE: 1

名前	1月	2月	3月	4月	5月	6月
太郎	100	200	300	400	500	600
次郎	110	220	330	440	550	660
三郎	121	242	363	484	605	726
四郎	410	420	330	440	450	460
五郎	510	520	330	540	550	560
六郎	610	620	330	640	650	660
七郎	710	720	330	740	750	760
八郎	810	820	330	840	850	860
A太郎	100	200	300	400	500	600
B作	110	220	330	440	550	660
C蔵	120	230	340	450	560	670
D吾	123,456,789.01	240	350	460	570	680

コンソール出力> 作成ファイル... C:¥Users¥User¥AppData¥Local¥Temp¥Report20220907130520k70c66qg.pdf 出力ページ：2

（3）コード

pdf_simple.py

```
1   # 日付
2   import datetime
3   # ログ
4   import logging
5   logging.basicConfig(level=logging.DEBUG) # CRITICAL,ERROR,WARNING,INFO,DEBUG,
6
7   # データ取得先 DB
8   import sqlite3
9   dbfile = 'C:/Users/User/Desktop/Python-Codes/dbdata.db'
10  sql = 'SELECT * FROM collect_tbl'
11
12  # PDF 作成ライブラリ < ReportLab PDF Toolkit (open source) >
13  from reportlab.pdfgen           import canvas
14  from reportlab.lib.pagesizes    import A4, landscape, portrait
15  from reportlab.lib.units        import cm, mm, inch
16  from reportlab.lib              import colors
17  from reportlab.pdfbase.pdfmetrics import registerFont
```



```
18
19  # ローカルフォント(ttc も可)を使うときこれで登録
20  from reportlab.pdfbase.ttfonts  import TTFont
21
22  # UnicodeCIDFont で'HeiseiMin-W3', 'HeiseiKakuGo-W5'が使えます
23  from reportlab.pdfbase.cidfonts import UnicodeCIDFont
24
25  # 一時ファイルとして PDF を作ります
26  # mkstemp はファイルのハンドルとファイルのパス名をタプルで返してくる
27  #  (ハンドルは使わないので"_"で受け取って以降スルー)
28  import tempfile
29  _, pdffile = tempfile.mkstemp(suffix='.pdf'
30                                , prefix='Report'+datetime.datetime.now()
31                                 .strftime('%Y%m%d%H%M%S')
32                                , dir=tempfile.gettempdir()
33                                ) # gettempdir()は環境変数 TEMP,TMP の値取得
34
35  # PDF で使うフォントの登録
36  registerFont(UnicodeCIDFont('HeiseiMin-W3'))
37  registerFont(UnicodeCIDFont('HeiseiKakuGo-W5'))
38  # 用紙の設定 (x,y 等の指定を含め、単位は 1/72 インチ)
39  # reportlab.lib.units の cm, mm, inchi を使った計算式で指定できます
40  pWidth, pHeight = landscape(A4)
41  c = canvas.Canvas(pdffile, pagesize=(pWidth, pHeight))
42  textobject = None
43  page = 0
44  colLen = 11 # 出力編集の最小文字数
45
46  def pageset():
47      ''' ページ設定 '''
48      global textobject
49      c.setFont('HeiseiMin-W3', 11)
50      textobject = c.beginText()
51      # テキストの開始位置は用紙の左下を 0,0 とした x 軸、y 軸で指定します
52      # ここで指定した値が左余白と上余白（頁高さ-余白）になります
53      textobject.setTextOrigin(1.5*cm, pHeight-2*cm)
54
55  def list2Line(rlist):
56      ''' pdf 1 行編集 '''
57      cur = 0; move =3.5*cm
58      for item in rlist:
59          textobject.textOut(item)
60          cur += move
61          textobject.moveCursor(move, 0)
62
63      # textLine で改行し、moveCoursor で加算した分を減算し
64      # カーソル位置を行頭に戻します
65      textobject.textLine()
66      textobject.moveCursor(-cur, 0)
67
68  def midashi():
69      ''' 見出しの出力 '''
70      pageset()
71      global page
```



```
72        page += 1
73        textobject.textLine(text=f'PAGE: {page}' )
74        list2Line(col_names)
75        textobject.textLine(text='—'*62)
76
77    def editItem(s, align='<'):
78        ''' 属性を判定して出力形式に編集する '''
79       #try: return '{:{align}{colLen},}'.format(int(s)) # これだと[,]編集されない...
80        try: return '{:{a}{l},}'.format(int(s), a=align, l=colLen)
81        except Exception: pass
82        try: return '{:{a}{l},}'.format(float(s), a=align, l=colLen)
83        except Exception: pass
84
85        return '{:{a}{l}}'.format(str(s), a=align, l=colLen)
86
87    # データベースに接続し、グローバル変数で定義した sql を実行します
88    con = sqlite3.connect(dbfile)
89    cur = con.cursor()
90    cur.execute(sql)
91
92    # 検索結果から列名（description）を取り出して見出しにします
93    # description は列名等 7 つの属性×列数 のリストになっています(DB-API 仕様)
94    # 1 つ目の列名はそのまま（左詰め）、2 つめからの列名は右左詰めにします
95    col_names = [cur.description[0][0]]
96    col_names += [f'{cn[0]:<{colLen}}' for cn in cur.description[1:]]
97    midashi()
98
99    # 検索結果から列のデータを取り出して編集します
100   rows = cur.fetchall()
101   for no, row in enumerate(rows):
102       xrow = [row[0]] + [editItem(x) for x in row[1:]] # 2 列目以降編集
103       list2Line(xrow)
104       # ライン出力で Y 軸の数値が小さくなるのでページ替えの判定をします
105       if textobject.getY() > 1*cm: # Y 軸が頁下余白（1CM）に達した？
106           pass
107       else:
108           c.drawText(textobject)
109           c.showPage()   # 次ページ用意
110           pageset()
111           midashi()
112
113   c.drawText(textobject) # PDF 生成
114   c.save()               # PDF ファイル保存
115   con.close()
116
117   print('作成ファイル...', pdffile, f'出力ページ：{page}')
```

ｉ． PDF の編集に使うオブジェクト

RportLab を使って PDF にテキストを出力する場合、主に以下のオブジェクトを使います。

① reportlab.pdfgen.canvas.Canvas …PDF の土台になります

② textobject …テキストの出力場所になります

ii．PDF の編集手順

① Canvas（PDF 土台）の生成

出力ファイル名とページサイズを指定して Canvas のインスタンスを生成し(41 行め)、setFont()でデフォルトのフォント種類とフォントサイズを設定します（49 行目)。このフォントはテキスト出力時にフォントを指定しなかったときに使われます

※ 使用するフォントは予め reportlab.pdfbase.pdfmetrics.registerFont を使った登録が（36 行目）必要です。日本語（マルチバイト）フォントは、Windows の場合は C:¥Windows¥Fonts に格納されているものか reportlab.pdfbase.cidfonts.UnicodeCIDFont の'HeiseiMin-W3'か'HeiseiKakuGo-W5'を使うことができます。UnicodeCIDFont を使うと Acrobat Reader が PDF の表示を行うときに使用環境に合わせたフォントを選択して表示してくれます

② テキスト出力領域の確保

Canvas から beginText()を使って textobject を受け取り（50 行目)、setTextOrigin()メソッドで余白（Canvas 上に textobject を配置する起点の x 軸と y 軸）を設定します

※ Canvas 上に表示するテキストや図形の配置は、Canvas の左下端を起点（x=0,y=0）として 1/72 インチ単位で指定します。指定には reportlab.lib.units の cm,mm,inch が使えます

③ 行の出力

textobject は主に以下のメソッドで扱います。

- moveCursor(x 軸, y 軸）…出力位置である仮想のカーソルを移動（加算）します（61 行め）
- textOut(*出力文字列*) …カーソル位置に文字列を設定します
- textLine(text=出力文字列) …text=を省略すると、改行（行ピッチ設定値だけ y 軸を移動する）だけ行われます（61 行目、66 行目）

※ moveCursor で移動させた y 軸は改行しても自動では行頭に戻りません。行頭に戻したい場合は移動させた分だけマイナスの moveCursor を行います

※ textLine で行全体を書きだすことができますが、ReportLab がフォント幅等の計算をしながら行を分割・コード化する際に、入力した文字列と PDF 上の配置が異なってきます。右端揃えで項目を出力したい場合は Tables /TableStyles や drawRightString(x, y, '*出力*テキスト')を使う必要があります。

④ ページの出力と改ページ

textobject に textOut()や textLine()でテキストを設定しても PDF ファイルには反映されません。textLine()で y 軸が起点（=0）に近づくので、余白分を残して Canvas へ textobject を反映させた後改ページを行って新しい textobject を作ります。各処理には以下のメソッドを使います。

- 余白の残り確認 …textobject.getY()（105 行目）
- Canvas への反映 …drawText(textobject)（108, 113 行目）
- Canvas の改ページ …showPage()（109 行目）
- 新しい textobject ⇒ 「② テキスト出力領域の確保」に戻る
- PDF ドキュメント生成／ファイル保存 …save()（114 行目）で PDF として参照できるようになります（実行後の IDLE Shell が開いていると Acrobat Reader で開けない場合があります）

Report スクリプティング（Python、ReportLab）

4. Python の出力編集（format、f 文字列、日本語）

PDF 編集用のライブラリでは全体の配置や配色を設定できますが、項目単位の編集は Python で行う必要があります。

項目の出力編集を行う方法は①文字列型の format メソッドと②f 文字列、③format 組み込み関数があります。前項のコード例で使っているのが①と②です。

① format メソッド

編集対象文字列.format(*値*, …)

```
colLen = 11 # 出力編集の最小文字数
def editItem(s, align='<'):
    ''' 属性を判定して出力形式に編集する '''
    try: return '{:{a}{l},}'.format(int(s), a=align, l=colLen)
    except Exception: pass
    try: return '{:{a}{l},}'.format(float(s), a=align, l=colLen)
    except Exception: pass

    return '{:{a}{l}}'.format(str(s), a=align, l=colLen)
```

この関数の内容は、引数の s に対して整数(int)、小数点付き(float)の順に変換を試し例外が発生しない型の編集を行うというものです。float でも例外が発生した場合は文字列として編集します。

編集対象文字列の'{:{a}{l},}'は、以下の意味があります。

{} …置換対象になります。{}の外の文字列はそのままの状態で置換後の文字列と合成されます

{{}} …波括弧の入れ子は内部のものから置換した結果が外側の波括弧に渡されていきます

{:書式指定} …この例では最初に{:<11,}に置換され、{:<11,} 左詰め、{:<11,} 最小文字数が 11、{:<11,}3 桁のカンマ編集になります。

最小文字数は当該文字数よりも文字数が少なかったときに文字数分のスペースを確保します。より多かった場合はそのままで編集されます。

文字数で切り詰めたい場合は編集後の文字列を *文字列*[:*文字数*]で部分参照します

※ 実際には aline が左詰め（上記のコード例）の場合、最小文字数の指定は意味を持ちません。PDF 以外のプレーンテキストで出力し、かつ項目の右端を揃えたいケースでのみ効果があります

② f 文字列

f'*編集対象文字列*'

```
colLen = 11 # 出力編集の最小文字数
col_names += [f'{cn[0]:<{colLen}}' for cn in cur.description[1:]]
```

編集対象文字列の書式指定は format メソッドと同様で、f'{cn[0]:<{colLen}}'の内容は、リスト cn の一つ目の要素に対して、左詰め、最小文字数 11 文字の編集を行うように指定しています。

※ ここで扱っている cur.description は [[列名 1, , , , , ,], [列名 2, , , , , ,], …×テーブル列数]の 2 次元リストになっていて、cur.description[1:] で列名 2 以降を処理しています

③ 日本語（マルチバイト文字）

Python で文字列を操作する場合の基準は文字数です[3]。日本語混じりの文字列を揃えて出力するためには含まれている全角の文字数をカウントしなければならず、その基準が以下の資料です。

Unicode の標準付属書 #11[4]に全角／半角の説明があり、標準付属書 #44[5] の 5.11 Validation には unicodedata.east_asian_width(*文字*) が以下の応答をすると規定しています（関係部分だけ和訳済）。

East_Asian_Width (ea)
ea ; A ; 曖昧
ea ; F ; 全角
ea ; H ; 半角
ea ; N ; Neutral
ea ; Na ; 半角
ea ; W ; 全角

上記の事項を利用し、日本語（全角）の文字数は以下の方法でカウントできます。

```
import unicodedata
fw = 0
for c in 文字列:
    if unicodedata.east_asian_width(c) in 'FWA': # 全角文字と曖昧
        fw += 1
```

文字列の表示幅を揃えるには等幅／固定ピッチフォント[6]（全角 1 文字が半角 2 文字分の幅）を使い全角の文字数 × 2 ＋ 半角の文字数 を揃えればよいので、全角文字数を減算すれば半角文字列の表示幅と一致します（例えば、半角 20 文字の表示幅に全角 5 文字を含む文字列を揃えるには 15 文字で設定すればよい）。 以下、実行例。

```
list = [ ['abcdefghij1234567890', 1,  11,  111]
        ,['abcdefghij12345',     11, 111, 1111]
        ,['あいうえお１２３４５', 1,  11,  111]
        ,['あいうえお',          11, 111, 1111]
       ]
for data in list:
    print(','.join( str(c) for c in data ))
```

＜実行結果＞

```
abcdefghij1234567890,1,11,111
abcdefghij12345,11,111,1111
あいうえお１２３４５,1,11,111
あいうえお,11,111,1111
```

---

[3] Python 3 からは文字列を Unicode で扱い、デフォルトエンコーディング（encode/decode）は UTF-8 で 1 文字を 1 ～ 4 バイトで表します

[4] Unicode® 標準付属書 #11 EAST ASIAN WIDTH https://unicode.org/reports/tr11/

[5] Unicode® 標準付属書 #44 UNICODE CHARACTER DATABASE
http://www.unicode.org/reports/tr44/tr44-6.html

[6] 等幅フォントでも常に正確に全角と半角が２：１になる固定ピッチとは限りません。Windows10 で試した範囲では「MS ゴシック」か「BIZ UD ゴシックのサイズ 9」でより正確に表示できます。

Report スクリプティング（Python、ReportLab）

ｉ. 日本語を考慮せず表示幅を書式設定した例

```
list = [ ['abcdefghij1234567890', 1,  11,  111]
        ,['abcdefghij12345',     11, 111, 1111]
        ,['あいうえお１２３４５', 1,  11,  111]
        ,['あいうえお',          11, 111, 1111]
       ]

for data in list:
    print( ('{:<20}|' + '{:>5}|'*3).format(*data) )
```

＜実行結果＞

```
abcdefghij1234567890|    1|   11|  111|
abcdefghij12345     |   11|  111| 1111|
あいうえお１２３４５          |    1|   11|  111|
あいうえお                |   11|  111| 1111|
```

ⅱ. 日本語を考慮した表示幅を書式設定した例

```
import unicodedata
def fw_count(w):
    ''' 引数に含まれる全角(fullwide) の文字数をカウントします '''
    fw = 0
    for c in w:
        if unicodedata.east_asian_width(c) in 'FWA': # 全角文字と曖昧
            fw += 1

    return fw

for data in list:
    fwr = fw_count(data[0])
    print( ('{:<{fwc}}|' + '{:>5}|'*3).format(fwc=20-fwr, *data) ) # 最小文字数から減
```

＜実行結果＞

```
abcdefghij1234567890|    1|   11|  111|
abcdefghij12345     |   11|  111| 1111|
あいうえお１２３４５|    1|   11|  111|
あいうえお          |   11|  111| 1111|
```


Report スクリプティング（Python、ReportLab）

5. 作成したレポートのダウンロード

作成した PDF をローカルで見る場合は作成した一時ファイルを開けばよいのですが、チームで情報共有するような場合等に CGI を使って簡単にダウンロードできるようにすることもできます。

（1）CGI のファイル構成

＜ダウンロードサービスディレクトリ＞

```
  cgisv.py          …CGI サーバ
  index.html        …ダウンロードのリンクを表示する html
  pdf_simple.py     …PDF を作成するモジュール
  |
  py-cgi            …サーバが CGI モジュールを探すディレクトリ（cgi_directories に指定）
      pdf_cgimod.py …CGI モジュール（上位ディレクトリの pdf_simple.py を import）
```

（2）CGI サーバ

CGI の機能を持っているサーバが使えるのであれば改めて用意する必要はありませんが、Python が提供するライブラリを利用する例を以下にあげます。

＜cgisv.py＞起動するとポート 8001 でサービスを開始します

```
from http.server import CGIHTTPRequestHandler, HTTPServer

class Handler(CGIHTTPRequestHandler):
    cgi_directories = ["/py-cgi"]

PORT = 8001
httpd = HTTPServer(("", PORT), Handler)
print(f"Serving on port {PORT}...")
httpd.serve_forever()
```

（3）index.html

ドメインだけを指定されたときに使われる index でダウンロードの get リクエストを出すようにしましたが、同様のリクエストが出せればファイル名その他の内容は変更しても問題ありません。

```
<!DOCTYPE html>
<html>
<head>
<body>
    <h3>File name: PDF-Report</h3>
    <h3><a href='py-cgi/pdf_cgimod.py?fname=PDF-Report.pdf' download>Download File</a></h3>
</body>
</html>
```

ブラウザからアクセスするとこの画面がでます

（4）CGI モジュール

起動すると環境変数経由でリクエストパラメータを受け取り、imort で PDF 作成モジュールを実行します。

< pdf_cgimod.py > index.html の href= に記述しておくモジュール

```
#!/usr/bin/env python

# ログ
import logging
logging.basicConfig(level=logging.DEBUG) # CRITICAL,ERROR,WARNING,INFO,DEBUG,

# データ取得先 DB
import sqlite3

# PDF 作成モジュール実行
import os
import sys
sys.path.append(os.path.join(os.path.dirname(__file__), '..'))
import pdf_simple # pdf_simple のスクリプト（関数、クラスに含まれていないコード）が実行されます
dfName = pdf_simple.pdffile # 出来上がった PDF ファイル名

# CGI ダウンロード処理
# CGI 用に使うライブラリ
import cgitb
cgitb.enable()

que_para = os.environ.get('QUERY_STRING') # リクエストパラメータを環境変数から取得
logging.debug('QUERY_STRING %s', que_para)
if que_para == None or que_para.find('=') == -1:
    file = 'temp.pdf'
else:
    paras = que_para.split('&') # リクエストパラメータ(fname=xxxx&para2=…)を個々のパラメータに分割
    form = {key.lower() : val for key, val in (para.split('=') for para in paras)}
    file = form['fname'] # fname という名前のリクエストパラメータを取り出します

print('Content-Type:application/pdf; name="%s"' % file)
print('Content-Disposition: attachment; filename="%s"' % file)
print('')

logging.debug(__name__ + dfName)

with open(dfName, 'rb') as f:
    # バイナリ出力が必要なので Python 3 以降は sys.stdout.buffer を使う
    sys.stdout.flush()
    sys.stdout.buffer.write(f.read())

    # Python ver.2 は print でバイナリ出力が可能でした
#    print(f.read())
```


Report スクリプティング（Python、ReportLab）

（５）PDF 作成モジュールの変更点

CGI モジュールは実行結果を print()で標準出力に送り、サーバがこの内容を http レスポンスとして要求元に送り返します。PDF 作成モジュールで行っている print()も CGI の環境で実行している場合は行わないように変更します。

「データベースから PDF を編集するコード例」pdf_simple.py の 117 行め

```
print('作成ファイル...', pdffile, f'出力ページ：{page}')
```

を、以下の条件を満たしたときだけ実行するにうに変更します

```
if __name__ == '__main__':
    print('作成ファイル...', pdffile, f'出力ページ：{page}')
```

※ この条件で判定している変数 __name__ の値は Python から直接実行されたときだけ '__main__' が設定され import で実行した場合はモジュール名（この例では pdf_simple ）が設定されます

以上

