

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA



BÁO CÁO TỔNG KẾT
MỞ RỘNG XỬ LÝ SỐ TÍN HIỆU

Đề tài:

Chuyển đổi từ EXCEL sang định dạng XML
và ngược lại theo quy chuẩn của bộ y tế.

Giáo viên hướng dẫn: Thầy Nguyễn Thanh Tuấn

Sinh viên thực hiện:

Thái Quang Nguyên 1813294

Thành phố Hồ Chí Minh-Tháng 11-Năm 2020

Mục lục

<i>I.</i>	<i>Dẫn nhập về định dạng XML:</i>	<i>3</i>
1.	XML là gì? Tác dụng của file XML:	3
2.	Đặc điểm của XML:	4
<i>II.</i>	<i>Cấu tạo thư mục & thư viện cần có:</i>	<i>4</i>
1.	Cấu tạo thư mục:	4
2.	Các thư viện cần có:	5
<i>III.</i>	<i>Diễn giải code:</i>	<i>5</i>
1.	Cách thức đọc file excel:	5
2.	Cách thức tạo và validate các trường dữ liệu:	6
3.	Cách thức chuyển từ .xml về lại .xlsx:	10
<i>IV.</i>	<i>References:</i>	<i>13</i>

I. Dẫn nhập về định dạng XML:

1. XML là gì? Tác dụng của file XML:

XML (Extensible Markup Language) là ngôn ngữ đánh dấu được tạo ra bởi World Wide Web Consortium (W3C) để xác định cú pháp mã hoá tài liệu để con người và máy có thể đọc được. Đây là một dạng ngôn ngữ đánh dấu, có chức năng truyền dữ liệu và mô tả nhiều loại dữ liệu khác nhau. Nó thực hiện điều này thông qua việc sử dụng thẻ xác định cấu trúc tài liệu cũng như cách tài liệu được lưu trữ và vận chuyển.

Mục đích chính của XML là đơn giản hoá việc chia sẻ dữ liệu giữa các platform và các hệ thống được kết nối với mạng Internet. Chính vì vậy, XML có tác dụng rất lớn trong việc chia sẻ, trao đổi dữ liệu giữa các hệ thống.

Có thể dễ dàng so sánh XML với một ngôn ngữ đánh dấu khác là Hypertext Markup Language (Ngôn ngữ đánh dấu siêu văn bản - HTML) được sử dụng để mã hoá các trang web. HTML sử dụng một tập hợp các ký hiệu đánh dấu được xác định trước mô tả định dạng nội dung trên một trang web.

Tuy nhiên, điểm khác nhau là XML có thể mở rộng được, nó không có ngôn ngữ đánh dấu được xác định trước như HTML. Thay vào đó, XML cho phép người dùng tạo biểu tượng đánh dấu riêng để mô tả nội dung, tạo một biểu tượng không giới hạn và tự định nghĩa. Đặc biệt, HTML là ngôn ngữ tập trung vào việc trình bày nội dung, trong khi XML là ngôn ngữ mô tả dữ liệu được sử dụng để lưu trữ dữ liệu.

XML thường được sử dụng làm cơ sở cho các định dạng tài liệu khác như: RSS, ATOM, Microsoft .NET...

Đây là một ví dụ về cú pháp của XML:

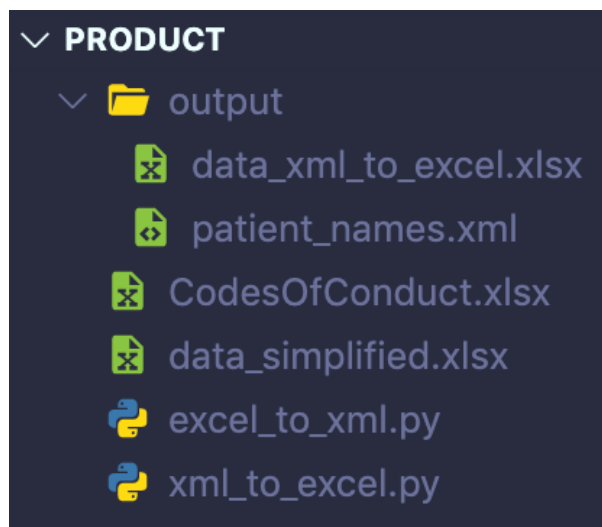
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

2. Đặc điểm của XML:

- XML được dùng cho dữ liệu có cấu trúc.
- Về trực quan, XML khá giống với HTML.
- Tuy là văn bản, nhưng XML không phải dùng để đọc.
- Cấu trúc file XML thường rất dài.
- XML được coi là cầu nối đưa HTML tới XHTML và là nền tảng cho RDF và Web mã hoá.
- XML là một module.
- XML miễn phí bản quyền, platform độc lập và được hỗ trợ rất tốt.

II. Cấu tạo thư mục & thư viện cần có:

1. Cấu tạo thư mục:



Truy cập repo của project tại [link này](#).
Dựa trên [trang web](#) chứa quy định chuẩn và định dạng dữ liệu trong quản lý y tế của *thuvienphapluat.vn*, tác giả đã soạn lại một số tiêu chuẩn trong file excel *CodesOfConduct.xlsx* ở thư mục ngoài *product*. File này chứa những tiêu chuẩn của các trường của dữ liệu, chia làm 4 sheets, minh hoạ như hình 1.2.

Ở những file trong thư mục con *product*, chứa file *data_simplified.xlsx* là file excel để trích xuất dữ liệu.

Hình 1.1. Cấu trúc thư mục

File *excel_to_xml.py* là file code python để đọc file .xlsx và xuất dữ liệu ra định dạng .xml, data đó cuối cùng được lưu trong file *patient_names.xml* trong folder con *output*.

File *xml_to_excel.py* làm tác vụ ngược lại với file trên là đọc file *patient_names.xml* và xuất ra lại file *data_xml_to_excel.xlsx*.

Bảng 1. Chỉ tiêu tổng hợp khám bệnh, chữa bệnh BHYT

STT	Chỉ tiêu	Kiểu dữ liệu	Kích thước tối đa	Diễn giải
1	MA_LK	Chuỗi	100	Mã đợt điều trị duy nhất (dùng để liên kết giữa bảng tổng hợp (bảng 1) và các bảng chi tiết (từ
2	STT	Số	10	STT tăng từ 1 đến hết trong 1 lần gửi dữ liệu.
3	MA_BN	Chuỗi	100	Mã số bệnh nhân quy định tại cơ sở khám bệnh, chữa bệnh.
4	HO_TEN	Chuỗi	255	Họ và tên người bệnh
5	NGAY_SINH	Chuỗi	8	Ngày sinh ghi trên thẻ gồm 8 ký tự; 4 ký tự năm + 2 ký tự tháng + 2 ký tự ngày (nếu không có)
6	GIOI_TINH	Số	1	Giới tính; Mã hóa (1: Nam; 2: Nữ; 3: Chưa xác định)
7	DIA_CHI	Chuỗi	1024	Ghi địa chỉ theo địa chỉ trên thẻ BHYT hoặc nơi cư trú hiện tại của người bệnh: số nhà (nếu có
8	MA_THE	Chuỗi	n	- Mã thẻ BHYT do cơ quan BHXH cấp - Trường hợp chưa có thẻ BHYT nhưng vẫn được hưởng quyền lợi BHYT, Ví dụ: trẻ em, người ghép tạng,... Ví dụ: TE101KT00000011 (Mã thẻ tạm cho trẻ em thứ 11 đến khám, giấy khai sinh/chứng sinh cấp tại Hà N - Trường hợp trong thời gian điều trị, người bệnh được cấp thẻ BHYT mới có thay đổi thông tin liên quan đ
9	MA_DKBD	Chuỗi	n	Mã cơ sở khám bệnh, chữa bệnh nơi người bệnh đăng ký ban đầu ghi trên thẻ BHYT, gồm có - Trường hợp trong thời gian điều trị, người bệnh được cấp thẻ BHYT mới có thay đổi thông tin - Trường hợp chưa có thẻ BHYT: Ghi mã đơn vị hành chính của tỉnh/TP + 000. Ví dụ: Hà Nội t
10	GT_THE_TU	Chuỗi	n	Thời điểm thẻ có giá trị gồm 8 ký tự; 4 ký tự năm + 2 ký tự tháng + 2 ký tự ngày - Trường hợp trong thời gian điều trị, người bệnh được cấp thẻ BHYT mới có thay đổi thông tin liên quan đ - Trường hợp chưa có thẻ BHYT: Thay thời điểm thẻ có giá trị bằng ngày người bệnh đến khám bệnh, chữa
11	GT.THE_DEN	Chuỗi	n	Thời điểm thẻ hết giá trị gồm 8 ký tự; 4 ký tự năm + 2 ký tự tháng + 2 ký tự ngày - Trường hợp trong thời gian điều trị, người bệnh được cấp thẻ BHYT mới có thay đổi thông tin - Trường hợp chưa có thẻ BHYT: Thay thời điểm thẻ hết giá trị bằng ngày người bệnh ra viện
12	MIEN_CUNG_CT	Chuỗi	8	- Thời điểm người bệnh bắt đầu được hưởng miễn cùng chi trả theo giấy xác nhận của cơ quan BHXH, gồm Ví dụ: ngày 31/03/2017 được hiển thị là: 20170331 - Nếu không có giấy xác nhận miễn cùng chi trả của cơ quan BHXH thì để trống
13	TEN_BENH	Chuỗi	n	Ghi đầy đủ các chẩn đoán được ghi trong hồ sơ, bệnh án

Hình 1.2. Nội dung minh họa của file CodeOfConduct.xlsx

2. Các thư viện cần có:

- [openpyxl](#): thư viện dùng để đọc/ghi file Excel theo những định dạng xlsx/xlsm/xltx/xltm.
- [yattag](#): thư viện dùng để tạo ra file HTML hoặc XML bằng code Python.
- [datetime](#): thư viện để đọc và tính toán ngày tháng trong Python.
- [BeautifulSoup](#): dùng để đọc file xml.

III. Diễn giải code:

1. Cách thức đọc file excel:

Trước tiên ta import hàm `load_workbook()` từ thư viện `openpyxl` và gọi hàm với đối số là tên của file excel chứa dữ liệu:

```
from openpyxl import load_workbook
wb = load_workbook("data_simplified.xlsx")
ws = wb.worksheets[0]
```

Tiếp đến ta import thư viện `yattag` để xuất định dạng xml:

```
from yattag import Doc, indent
# Create Yattag doc, tag and text objects
doc, tag, text = Doc().tagtext()
```

Class `yattag.Doc` hoạt động như cách ta liên kết các chuỗi lại với nhau, ví dụ đơn giản như hình dưới:

```
mylist = []
mylist.append('Everybody')
mylist.append('likes')
mylist.append('pandas.')
mystring = ' '.join(mylist) # mystring contains "Everybody likes pandas."
```

Hình 2.1. Cách thức hoạt động của class `yattag.Doc`

	A	B	C
1	Chỉ tiêu	Bệnh nhân 1	Bệnh nhân 2
2	MA_LK	LK123122	LK123123
3	STT	1	1
4	MA_BN	BN123123	BN123124
5	HO_TEN	NGUYỄN VĂN NỘI KHOA	NGUYỄN THỊ NGOẠI KHOA
6	NGAY_SINH	1/1/20	1/1/00
7	GIOI_TINH	Nam	Chưa xác định
8	DIA_CHI	ận Ba Đình, Thành phố Hà Nội	Quận Ba Đình, Thành phố Hà Nội
9	MA_THE	HS123123122222	HS123123122222
10	MA_DKBD	DKBD1000	DKBD1001
11	GT_THE_TU	12/3/00	12/3/00
12	GT_THE_DEN	12/3/00	12/3/00
13	MIEN_CUNG_CT	12/4/00	12/5/00
14	TEN_BENH	điên khùng	điên khùng
15	MA_BENH	BENH22222	BENH22223
16	MA_BENHKHAC	BENHKHAC22222	BENHKHAC22223
17	MA_LYDO_VVIEN	Đúng tuyến	Đúng tuyến
18	MA_NOI_CHUYEN	NOICHUYEN12345	NOICHUYEN12346

Hình 2.2. Nội dung file `data_simplified.xlsx`

Tạo mẫu file

`data_simplified.xlsx` chứa nội dung là các trường (cột A) và các thông số tương ứng của từng bệnh nhân (từ cột B trở đi).

Bây giờ ta sẽ đọc lần lượt từng bệnh nhân (từng cột B, C), trong mỗi bệnh nhân ta đọc từng hàng chính là thông số của các trường liên quan đến bệnh nhân. Các dòng code dưới giúp ta làm việc này:

2. Cách thức tạo và validate các trường dữ liệu:

```
with tag('Cac_Benh_Nhan'):
    # Use ws.max_row for all rows
    for col in ws.iter_cols(min_col=2, max_col=3, min_row=2,
max_row=41):
        col = [cell.value for cell in col]
```

Tạo một tag `<Cac_Benh_Nhan></Cac_Benh_Nhan>` để lưu trữ thông tin từ file `data_simplified.xlsx` trong một lần đọc dữ liệu. Dữ liệu của chúng ta bắt đầu từ hàng 2 cho đến hàng 41, và từ cột B cho đến C (tức là cột 2 và 3), ta khai báo `min_row`, `max_row`, `min_col`, `max_col` như trên là đối số của hàm `iter_cols()`. Ta lưu tất cả các giá trị của từng cell của một cột vào mảng một chiều `col`, sau này ta có thể truy xuất từng giá trị đó thông qua index của mảng `col`.

```

with tag("Benh_Nhan"):
> with tag("node", klass='code', type='string_100', id='MA_LK'):--
> with tag("node", klass='number', type='int_10', id='STT'):--
> with tag("node", klass='code', type='string_100', id='MA_BN'):--
> with tag("node", klass='code', type='string_255', id='HO_TEN'):--
> with tag("node", klass='time', type='year_month_day', id='NGAY_SINH'):--
> with tag("node", klass='detail', type='selection_1', id='GIOI_TINH'):--
> with tag("node", klass='code', type='string_1024', id='DIA_CHI'):--
> with tag("node", klass='code', type='string_n', id='MA_THE'):--
> with tag("node", klass='code', type='string_n', id='MA_DKBD'):--
> with tag("node", klass='time', type='year_month_day', id='GT_THE_TU'):--
> with tag("node", klass='time', type='year_month_day', id='GT_THE_DEN'):--
> with tag("node", klass='time', type='year_month_day', id='MIEN_CUNG_CT'):--
> with tag("node", klass='code', type='string_n', id='TEN_BENH'):--
> with tag("node", klass='code', type='string_15', id='MA_BENH'):--
> with tag("node", klass='code', type='string_255', id='MA_BENHKHAC'):--
> with tag("node", klass='detail', type='selection_1', id='MA_LYDO_VVIEN'):--
> with tag("node", klass='code', type='string_5', id='MA_NOI_CHUYEN'):--
> with tag("node", klass='number', type='int_1', id='MA_TAI_NAN'):--
> with tag("node", klass='time', type='year_month_day_hour_minute', id='NGAY_VAO'):--
> with tag("node", klass='time', type='year_month_day_hour_minute', id='NGAY_RA'):--
> with tag("node", klass='number', type='int_3', id='SO_NGAY_DTRI'):--
> with tag("node", klass='detail', type='selection_1', id='KET_QUA_DTRI'):--
> with tag("node", klass='detail', type='selection_1', id='TINH_TRANG_RV'):--
> with tag("node", klass='time', type='year_month_day', id='NGAY_TTOAN'):--
> with tag("node", klass='number', type='float_15_decimal_2', id='T_THUOC'):--
> with tag("node", klass='number', type='float_15_decimal_2', id='T_VTYT'):--
> with tag("node", klass='number', type='float_15_decimal_2', id='T_TONGCHI'):--
> with tag("node", klass='number', type='float_15_decimal_2', id='T_BNTH'):--
> with tag("node", klass='number', type='float_15_decimal_2', id='T_BNCT'):--
> with tag("node", klass='number', type='float_15_decimal_2', id='T_BHTT'):--
> with tag("node", klass='number', type='float_15_decimal_2', id='T_NGUONKHAC'):--
> with tag("node", klass='number', type='float_15_decimal_2', id='T_NGOAIDS'):--
> with tag("node", klass='time', type='year', id='NAM_QT'):--
> with tag("node", klass='time', type='month', id='THANG_QT'):--
> with tag("node", klass='detail', type='selection_1', id='MA_LOAI_KCB'):--
> with tag("node", klass='code', type='string_15', id='MA_KHOA'):--
> with tag("node", klass='code', type='string_5', id='MA_CSKCB'):--
> with tag("node", klass='detail', type='selection_2', id='MA_KHUVUC'):--
> with tag("node", klass='code', type='string_255', id='MA_PTTT_QT'):--
> with tag("node", klass='number', type='float_5_decimal_2', id='CAN_NANG'):--

```

Với mỗi một trường dữ liệu, ta sẽ tạo một tag với tên tương ứng, cú pháp như hình bên. Ta có thể chia các trường dữ liệu theo *class* và *type*, ví dụ như class ‘time’ để chỉ thời gian và type ‘year_month_day’ để chỉ quy định của nội dung tag đó.

Type ‘int_5’ là số integer khi chuyển thành chuỗi bằng lệnh `str()` thì `len(str())` sẽ bé hơn hoặc bằng 5, tương ứng với yêu cầu trong file *CodeOfConduct.xlsx*.

Hình 2.3. Các tag và class, type tương ứng

Ở mỗi trường giá trị khi đọc vào, ta check xem có giá trị hay không hàm `if col[count] == None`, biến `count` được dùng để đọc lần lượt các hàng trong một cột, khi đọc xong một hàng thì sẽ được cộng lên 1. Điều kiện tiếp theo chính là chiều dài của dữ liệu có thỏa mãn yêu cầu không, ta có ví dụ như trường dữ liệu *MA_LK*:

```

with tag("node", klass='code', type='string_100', id='MA_LK'):
    if col[count] == None or len(str(col[count])) > 100:
        text("NODATA")
    else:
        text(col[count])
    count += 1

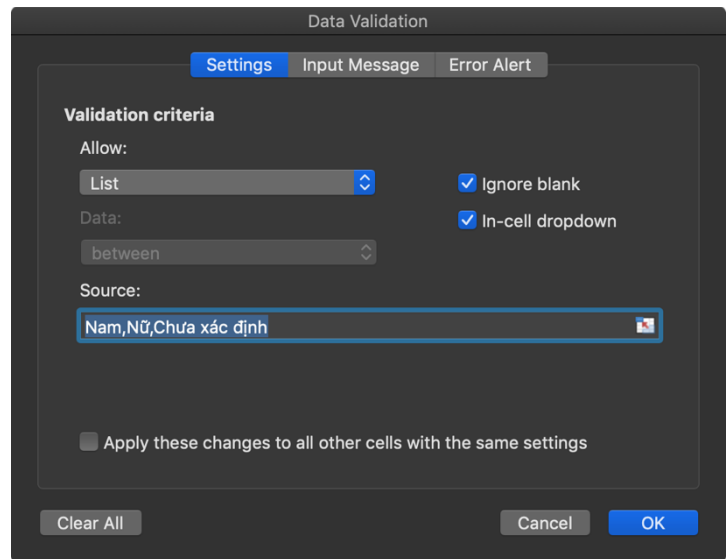
```

Ở những trường như là ‘GIOI_TINH’, giá trị nhập vào excel được phân thành những options là ‘Nam’, ‘Nữ’ hoặc ‘Chưa xác định’. Điều này được thực hiện trong file *data_simplified.xlsx* bằng chức năng Data Validation như sau:

GIOI_TINH		Nam	
DIA_CHI		Hà Nội, Q	
MA_THE		222222	
MA_DKBD		BD1000	
GT THE TU		12/3/00	

Hình 2.4. Chức năng Data Validation trong EXCEL để tạo option box

Nhờ đó, ta có thể ràng buộc khoảng giá trị nhập vào của người nhập. Tiếp đến ta đọc cell này ở code python như sau:



```
with tag("node", klass='detail', type='selection_1', id='GIOI_TINH'):
    if col[count] == None:
        text("NODATA")
    elif col[count] == "Nam":
        text('1')
    elif col[count] == "Nữ":
        text('2')
    elif col[count] == "Chưa xác định":
        text('3')
    count += 1
```

Tương tự như hàm switch case của C, ta xuất ký tự '1', '2', '3' tương tự với các giá trị đầu vào là 'Nam', 'Nữ' và 'Chưa xác định'.

Ở trường dữ liệu là thời gian, nhờ vào thư viện datetime, ta check xem đó có phải là biến datetime hay không bằng hàm `type()`, sau đó xuất ra theo định dạng mong muốn là 'yyymmdd' bằng hàm `strftime("%Y%m%d")`.

```
with tag("node", klass='time', type='year_month_day', id='NGAY_SINH'):
    if type(col[count]) is datetime.datetime:
        temp_date = col[count]
        text(col[count].strftime("%Y%m%d"))
    else:
        text("NODATA")
    count += 1
```

Ở đây, vì ở trường dữ liệu CAN_NANG ở cuối, ta chỉ thu thập dữ liệu đối với các trẻ em dưới 1 tuổi, nên ta sẽ dùng thư viện datetime để lấy giá trị của thời điểm hiện tại, trừ đi cho giá trị ngày sinh nhập vào xem có bé hơn 365 ngày không (ở đây chưa xét đến yếu tố năm nhuận hay múi giờ).

40	CAN_NANG	số	5	Chỉ thu thập với các bệnh nhân là trẻ em dưới 1 tuổi. Là số kilogam (kg) cân nặng của trẻ em khi vào viện. Biểu thị đầy đủ cả Số thập phân, dấu thập phân là dấu chấm (.), ghi đến 2 chữ số sau dấu thập phân. Ví dụ: 5.75 kg.
----	----------	----	---	--

Hình 2.5. Quy định của trường dữ liệu CAN_NANG

```
current_date = datetime.datetime.now()
temp_date = datetime.datetime(2020, 11, 22)
#...
with tag("node", klass='number', type='float_5_decimal_2',
id='CAN_NANG'):
    if (current_date - temp_date).days <= 365:
        if type(col[count]) == float and len(str(col[count])) <= 5:
            text(col[count])
            age = (current_date - temp_date).days
            print(current_date)
            print(temp_date)
            print(age)
        else:
            text("NODATA")
```

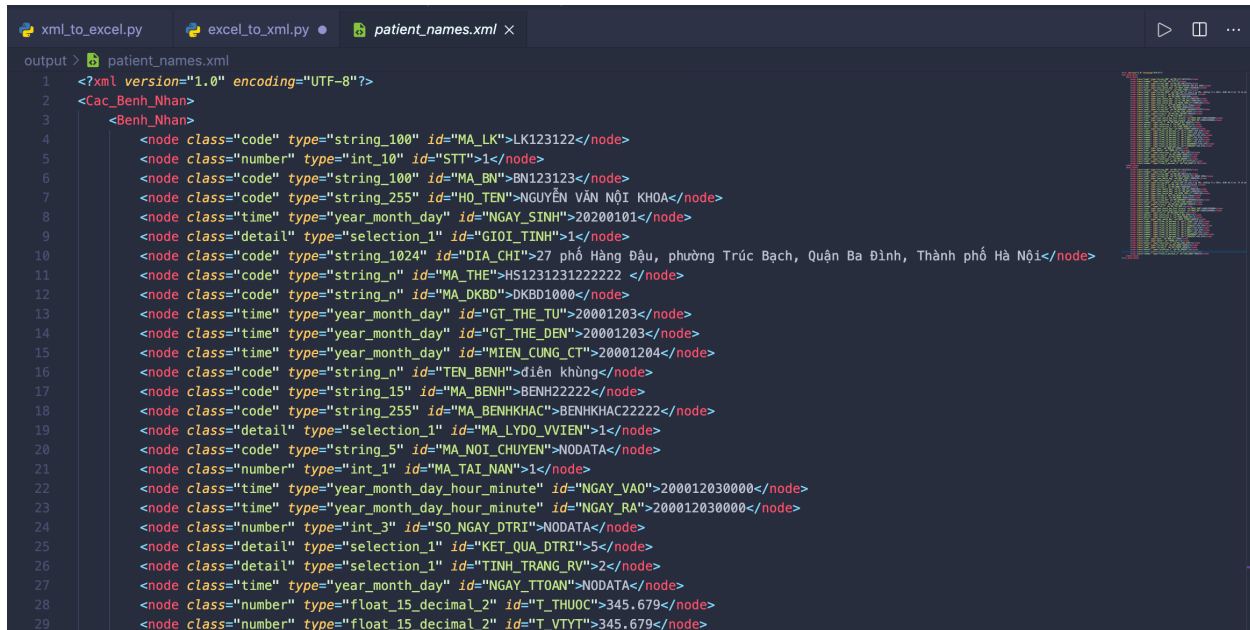
Biến *current_date* được gán cho thời điểm đọc file excel, biến *temp_date* được khởi tạo và về sau được gán bằng giá trị ngày sinh của bệnh nhân. Hiệu của hai giá trị này tính ra số ngày nếu thỏa bé hơn hoặc bằng 365 thì thỏa mãn, tính được bằng hàm *(current_date - temp_date).days*.

Cuối cùng, ta có thể lưu lại các giá trị gồm các tag, text vào file *patient_names.xml*:

```
result = indent(
    doc.getvalue(),
    indentation = '    '
)

with open("output/patient_names.xml", "w") as f:
    f.write(result)
```

Kết quả thu được như hình dưới:



```
output > patient_names.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <Cac_Benh_Nhan>
3   <Benh_Nhan>
4     <node class="code" type="string_100" id="MA_LK">LK123122</node>
5     <node class="number" type="int_10" id="STT">1</node>
6     <node class="code" type="string_100" id="MA_BN">BN123123</node>
7     <node class="code" type="string_255" id="HO_TEN">NGUYỄN VĂN NỘI KHOA</node>
8     <node class="time" type="year_month_day" id="NGAY_SINH">20200101</node>
9     <node class="detail" type="selection_1" id="GIOI_TINH">1</node>
10    <node class="code" type="string_1024" id="DIA_CHI">27 phố Hàng Đậu, phường Trúc Bạch, Quận Ba Đình, Thành phố Hà Nội</node>
11    <node class="code" type="string_n" id="MA_THE">HS1231231222222 </node>
12    <node class="code" type="string_n" id="MA_DKBD">DKBD1000</node>
13    <node class="time" type="year_month_day" id="GT_THE_TU">20001203</node>
14    <node class="time" type="year_month_day" id="GT_THE_DEN">20001203</node>
15    <node class="time" type="year_month_day" id="MIEN_CUNG_CT">20001204</node>
16    <node class="code" type="string_n" id="TEN_BENH">diễn khùng</node>
17    <node class="code" type="string_15" id="MA_BENH">BENH22222</node>
18    <node class="code" type="string_255" id="MA_BENHKHAC">BENHKKHAC22222</node>
19    <node class="detail" type="selection_1" id="MA_LYDO_VVIEN">1</node>
20    <node class="code" type="string_5" id="MA_NOI_CHUYEN">NODATA</node>
21    <node class="number" type="int_1" id="MA_TAI_NAN">1</node>
22    <node class="time" type="year_month_day_hour_minute" id="NGAY_VAO">200012030000</node>
23    <node class="time" type="year_month_day_hour_minute" id="NGAY_RA">200012030000</node>
24    <node class="number" type="int_3" id="SO_NGAY_DTRI">NODATA</node>
25    <node class="detail" type="selection_1" id="KET_QUA_DTRI">5</node>
26    <node class="detail" type="selection_1" id="TINH_TRANG_RV">2</node>
27    <node class="time" type="year_month_day" id="NGAY_TTOAN">NODATA</node>
28    <node class="number" type="float_15_decimal_2" id="T_THUOC">345.679</node>
29    <node class="number" type="float_15_decimal_2" id="T_VTYT">345.679</node>
```

Hình 2.6. File `patient_names.xml` xuất ra được

3. Cách thức chuyển từ .xml về lại .xlsx:

Với file `output/patient_names.xml` có được từ việc chạy file `excel_to_xml.py`, ta có thể tiếp đến đọc ngược lại file .xml về lại .xlsx bằng thư viện BeautifulSoup.

Ta định nghĩa một hàm `readFile()` như sau để đọc dữ liệu vào nhờ BeautifulSoup:

```
def readFile(filename):
    if not os.path.exists(filename):
        print("Cannot find .xml file!")
        os._exit(0)
        return
    with open(filename, 'r') as f:
        data = f.read()
    Bs_data = BeautifulSoup(data, "xml")
```

Với trường dữ liệu của ta, các thông tin của bệnh nhân được lưu dưới label 'node' và label tổng (tức là mỗi bệnh nhân) là 'Benh_Nhan', ta sử dụng hàm `find_all()` để tìm tất cả các trường này, phục vụ cho việc quét dữ liệu tiếp theo:

```
bn_details = Bs_data.find_all('node')
benh_nhan = Bs_data.find_all('Benh_Nhan')
```

Tiếp đến ta sẽ quét 2 mảng này để thực hiện validate dữ liệu và xuất ra file excel, code như sau:

```
mdlist = []
temp = []
for _node in bn_details:
    temp.append(_node.get('id'))
mdlist.append(temp)
```

```

for _benh_nhan in benh_nhan:
    temp = []
    for _node in bn_details:
        node_text = check_data(_class=_node.get('class') ,
                                _type=_node.get('type') , _id=_node.get('id') ,
                                _data=_node.get_text())
        temp.append(node_text)
    mdlist.append(temp)
return mdlist

```

Vector 2 chiều *mdlist[]* sẽ là đối số trả về của hàm *readFile()*, với hàng đầu tiên là tên của các trường dữ liệu (chính là *id* của chúng), những hàng tiếp theo là giá trị của các trường dữ liệu (mỗi hàng là thông tin của từng bệnh nhân).

Khi quét dữ liệu của từng bệnh nhân, ta sẽ dùng hàm *check_data()* được định nghĩa phía dưới để validate dữ liệu, với các đối số vào là *class*, *type*, *id* và *data* của từng node, các thông tin này quy định cấu trúc dữ liệu mà ta đã tạo nên ở file python trước *excel_to_xml.py*.

Ở hàm *check_data()* ta validate dữ liệu dùng switch case với quy định có sẵn trong đối số *_type*, ví dụ như sau:

```

def check_data(_class=None, _type=None, _id=None, _data=None):
    if _data == "NODATA":
        return None
    elif _class == "number":
        if ("int" in _type) is True:
            if len(_data) <= int(_type[4:]):
                return int(_data)
            else:
                return None
        elif ("float" in _type) is True:
            stt_float = _type.find("float_")
            stt_decimal = _type.find("_decimal_")
            if len(_data) <= int(_type[6:stt_decimal]):
                return round(float(_data), int(_type[stt_decimal+9:]))

```

Tương tự với các class còn lại:

```
def check_data(_class=None, _type=None, _id=None, _data=None):
    if _data == "NODATA":
        return None
    > elif _class == "number": ...
    > elif _class == "code": ...
    > elif _class == "time": ...
    > elif _class == "detail": ...
    return None
```

Khi đã có một vector hai chiều là *mdlist[]*, tiếp đến ta bỏ nó vào file excel mới trong thư mục *output* bằng hàm *to_Excel()*:

```
def to_Excel(mdlist):
    wb = Workbook()
    ws = wb.active
    for i,row in enumerate(mdlist):
        for j,value in enumerate(row):
            ws.cell(row=i+1, column=j+1).value = value
    newfilename = os.path.abspath("./output/data_xml_to_excel.xlsx")
    wb.save(newfilename)
    print("Process completed")
    return
```

Dùng hàm *enumerate()* để đánh số từng dòng của *mdlist[]* và quét các số đó thông qua biến *i*, quét dữ liệu thông qua biến *row*. Dùng hàm *enumerate()* tương tự với mảng *row* (tức là từng hàng của *mdlist[]*), lưu các số để đánh thứ tự vào biến *j*. Sau là một ví dụ của hàm *enumerate()*:

```
# Python program to illustrate
# enumerate function
l1 = ["eat", "sleep", "repeat"]
s1 = "geek"
```

```
Return type: < type 'enumerate' >
[(0, 'eat'), (1, 'sleep'), (2, 'repeat')]
```

Hình 3.1. Ví dụ về hàm *enumerate()*

```
# creating enumerate objects
obj1 = enumerate(l1)
obj2 = enumerate(s1)

print "Return type:", type(obj1)
print list(enumerate(l1))
```

Qua đó, ta có biến *i* chỉ số hàng và biến *j* chỉ số cột trong file excel, trong file excel thì số đếm bắt đầu từ 1 nên ta cộng *i* và *j* với 1. Gán giá trị của từng cell bằng hàm *cell().value*. Cuối cùng ta save lại file excel mới trong thư mục *output*:

```
result = readFile("output/patient_names.xml")
if result:
    to_Excel(result)
```

	A	B	C	D	E	F	G
1	MA_LK	STT	MA_BN	HO_TEN	NGAY_SINH	GIOI_TINH	DIA_CHI
2	LK123122	1	BN123123	NGUYỄN VĂN NỘI KHOA	2020-01-01 0:00:00	Nam	27 phố Hà
3	LK123122	1	BN123123	NGUYỄN VĂN NỘI KHOA	2020-01-01 0:00:00	Nam	27 phố Hà
4							

Hình 3.2. Kết quả file data_xml_to_excel.xlsx

IV. References:

<https://stackoverflow.com/>

<https://www.javatpoint.com/xml-tutorial>

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>