**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. Dẫn nhập về định dạng XML: 3](#_Toc57897796)

[1. XML là gì? Tác dụng của file XML: 3](#_Toc57897797)

[2. Đặc điểm của XML: 4](#_Toc57897798)

[II. Cấu tạo thư mục & thư viện cần có: 4](#_Toc57897799)

[1. Cấu tạo thư mục: 4](#_Toc57897800)

[2. Các thư viện cần có: 5](#_Toc57897801)

[III. Diễn giải code: 5](#_Toc57897802)

[1. Cách thức đọc file excel: 5](#_Toc57897803)

[2. Cách thức tạo và validate các trường dữ liệu: 6](#_Toc57897804)

[3. Cách thức chuyển từ .xml về lại .xlsx: 10](#_Toc57897805)

[IV. References: 13](#_Toc57897806)

1. **Dẫn nhập về định dạng XML:**
2. **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.

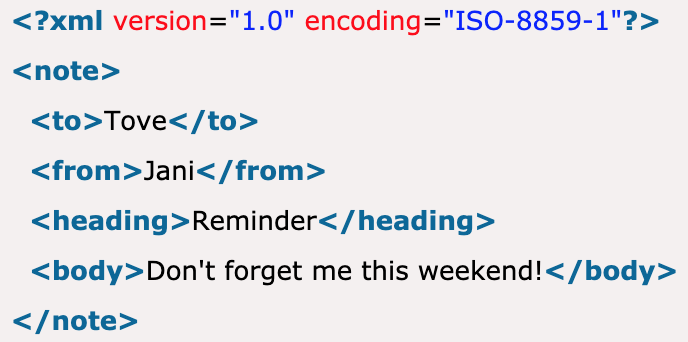
Mục đích chính của XML là đơn giản hoá việc chia sẻ dữ liệu giữa các platform và các hệ thống được kết nối với mạng Internet. Chính vì vậy, XML có tác dụng rất lớn trong việc chia sẻ, trao đổi dữ liệu giữa các hệ thống.

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…

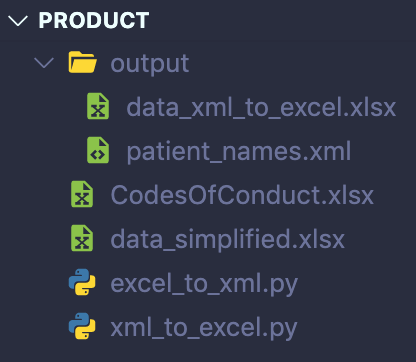
Đây là một ví dụ về cú pháp của XML:



1. **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.

1. **Cấu tạo thư mục & thư viện cần có:**
2. **Cấu tạo thư mục:**



Truy cập repo của project tại [link này](https://github.com/TQN-BACHKHOA/201_XLSTH_MR).

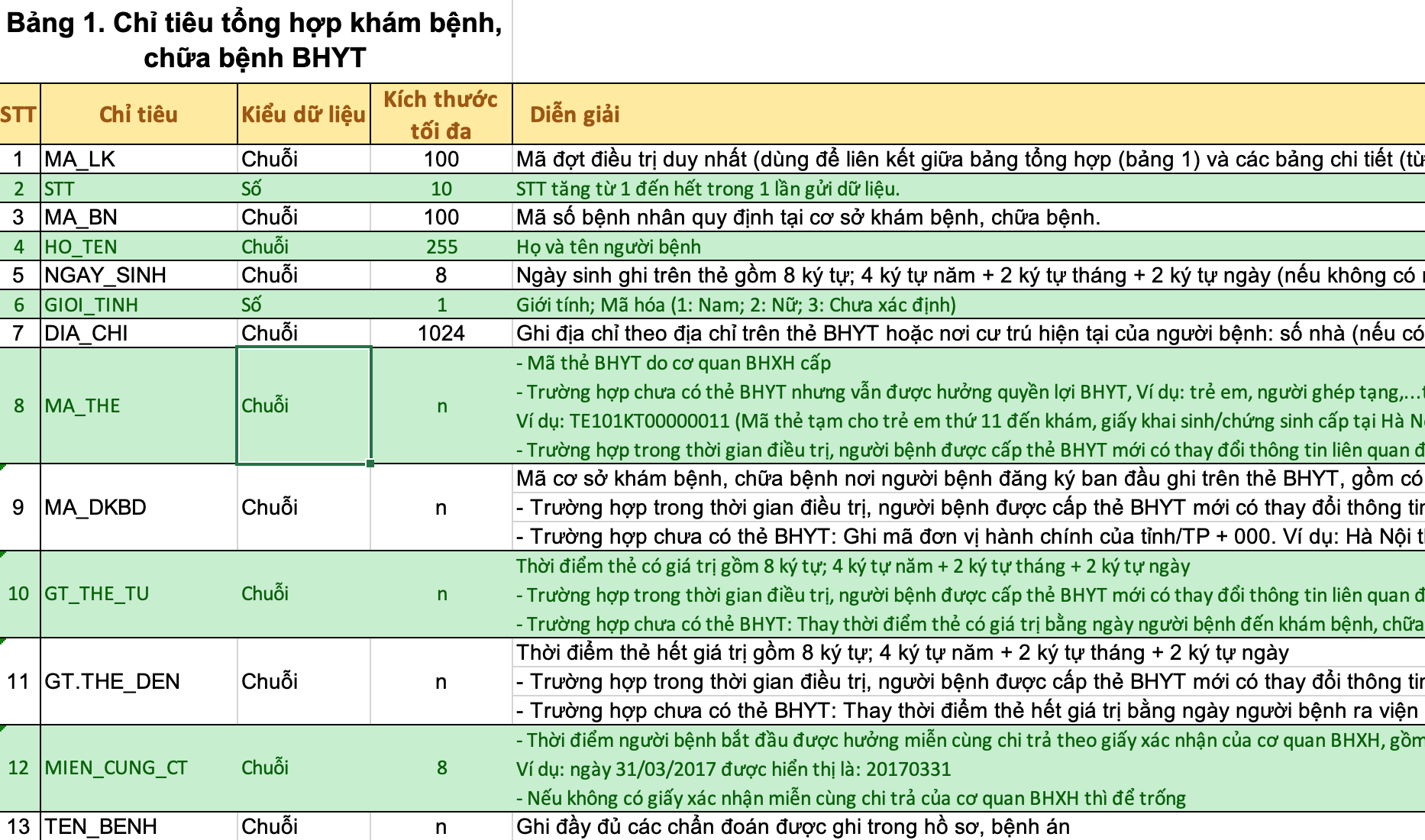
Dựa trên [trang web](https://thuvienphapluat.vn/van-ban/bao-hiem/Quyet-dinh-4210-QD-BYT-2017-du-lieu-dau-ra-trong-thanh-toan-chi-phi-kham-chua-benh-y-te-361955.aspx) chứa quy định chuẩn và định dạng dữ liệu trong quản lý y tế của *thuvienphapluat.vn*, tác giả đã soạn lại một số tiêu chuẩn trong file 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 foler 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*.

**

*Hình 1.2. Nội dung minh hoạ của file CodeOfConduct.xlsx*

1. **Các thư viện cần có:**

* [openpyxl](https://openpyxl.readthedocs.io/en/stable/): thư viện dùng để đọc/ghi file Excel theo những định dạng xlsx/xlsm/xltx/xltm.
* [yattag](https://www.yattag.org/): thư viện dùng để tạo ra file HTML hoặc XML bằng code Python.
* [datetime](https://docs.python.org/3/library/datetime.html): thư viện để đọc và tính toán ngày tháng trong Python.
* [BeautifulSoup](https://www.crummy.com/software/BeautifulSoup/bs4/doc/): dùng để đọc file xml.

1. **Diễn giải code:**
2. **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]

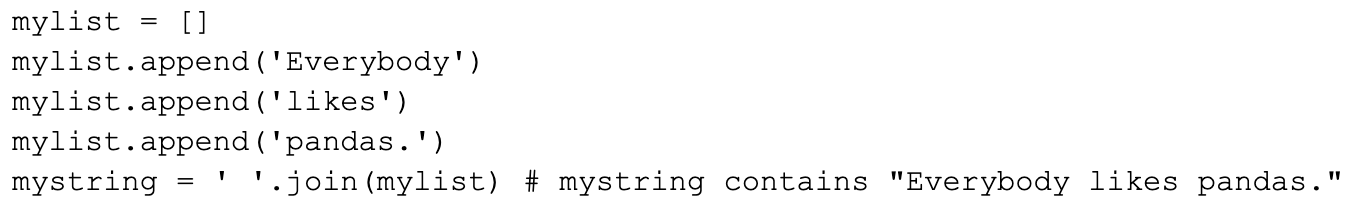
Tiếp đến ta import thư viện yattag để xuất định dạng xml:

*from* yattag *import* Doc, indent

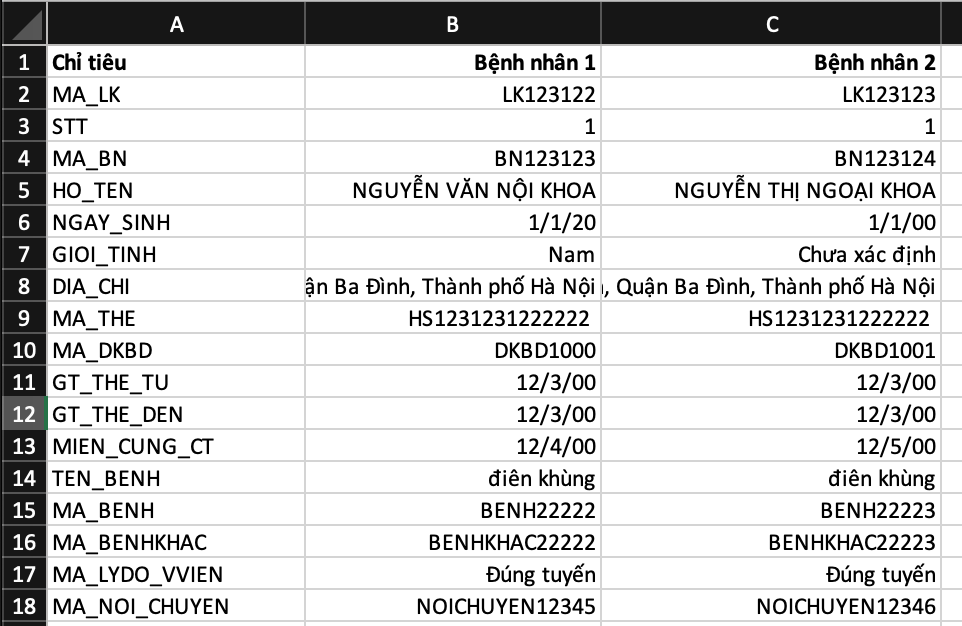
# *Create Yattag doc, tag and text objects*

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:



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



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:

*Hình 2.2. Nội dung file data\_simplified.xlsx*

1. **Cách thức tạo và validate các trường dữ liệu:**

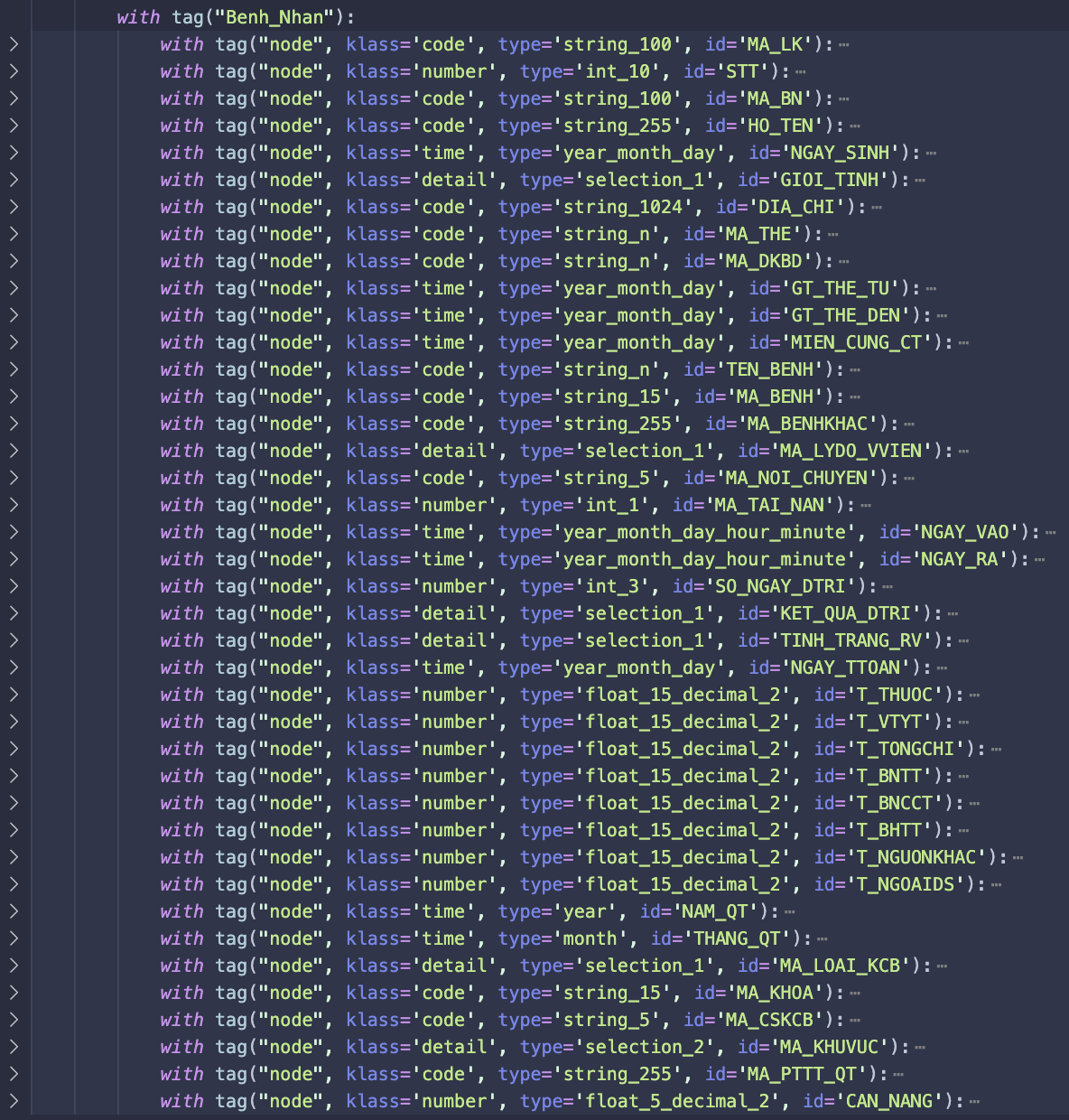
*with* tag('Cac\_Benh\_Nhan'):

# *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*.



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 thoả 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:

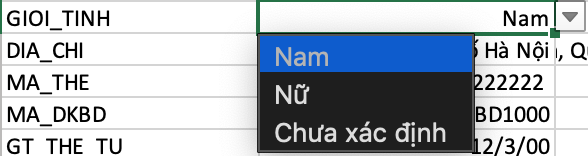
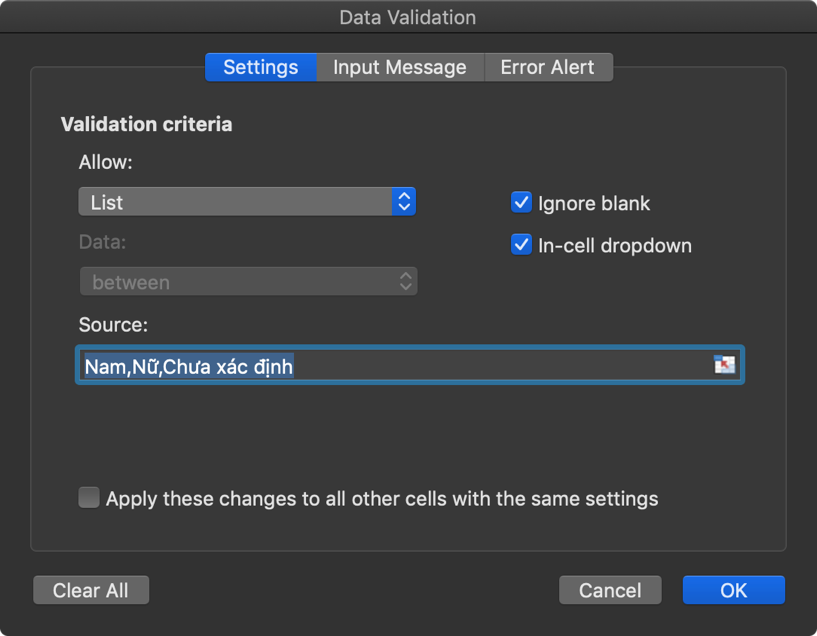
text("NODATA")

*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:



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

Nhờ đó, ta có thể rào được 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 *‘yyyymmdd’* 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]

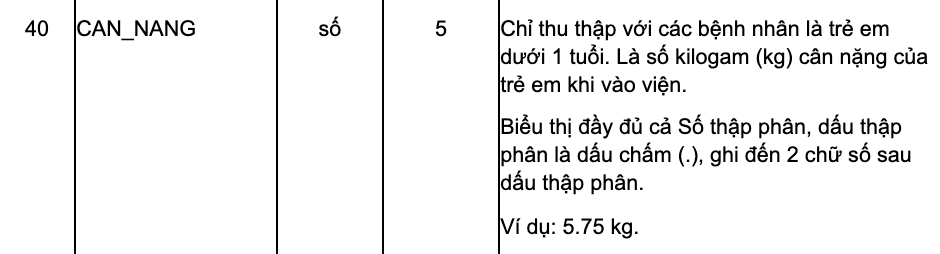
text(col[count].strftime("%Y%m%d"))

*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).



*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 thoả bé hơn hoặc bằng 365 thì thoả 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(),

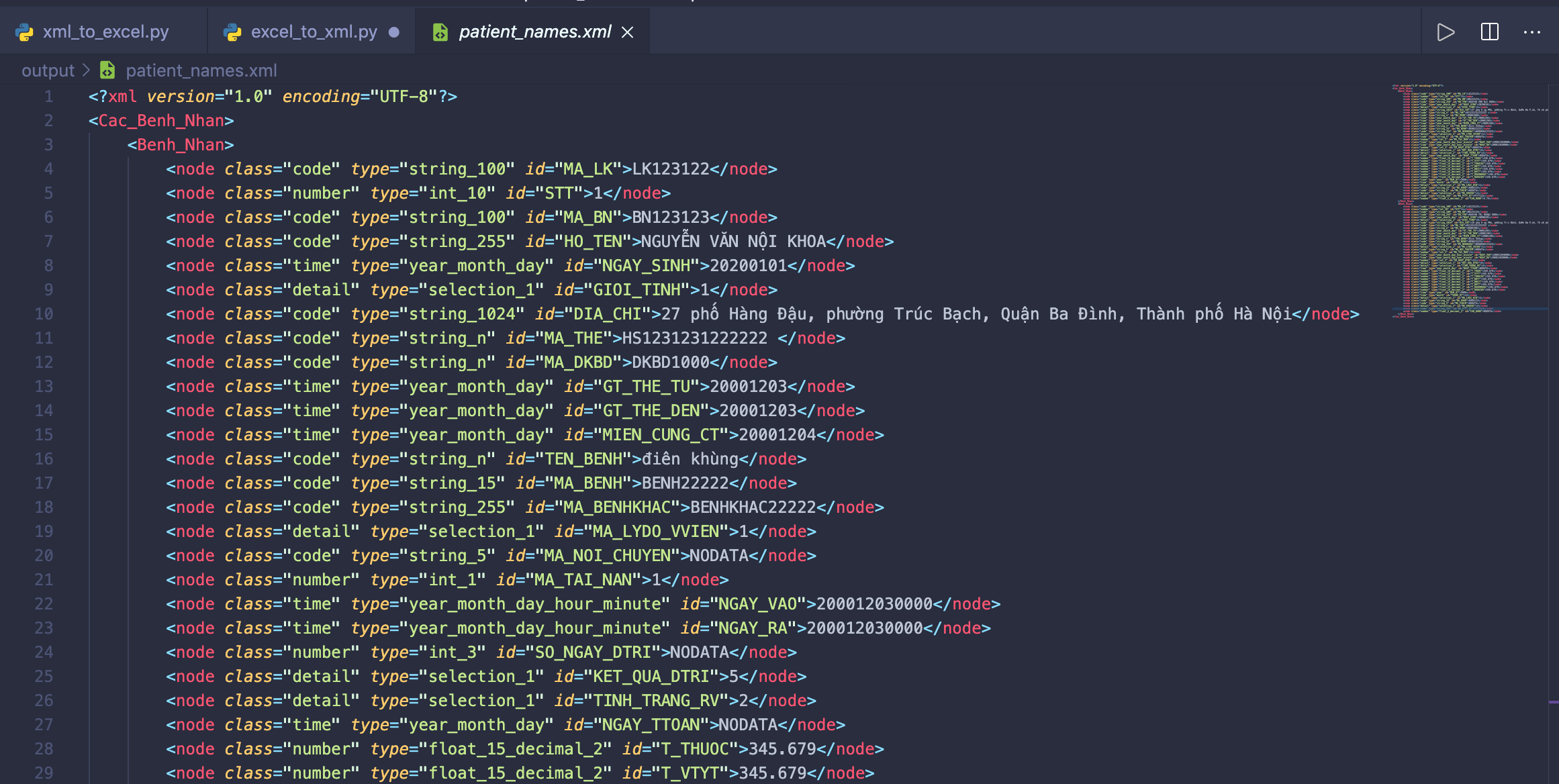
indentation = ' '

)

*with* open("output/patient\_names.xml", "w") *as* f:

f.write(result)

Kết quả thu được như hình dưới:



*Hình 2.6. File patient\_names.xml xuất ra được*

1. **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:

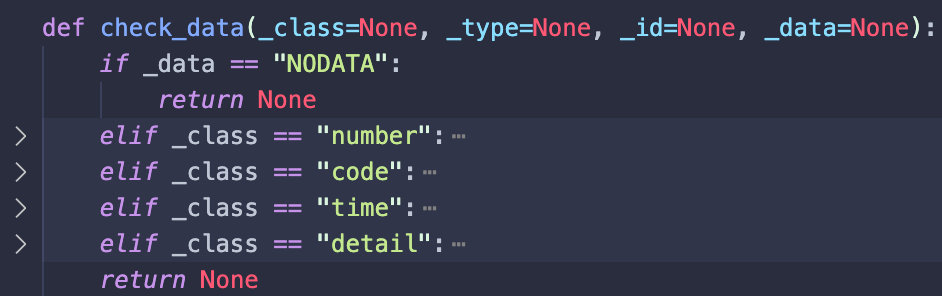
stt\_float = \_type.find("float\_")

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:



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):

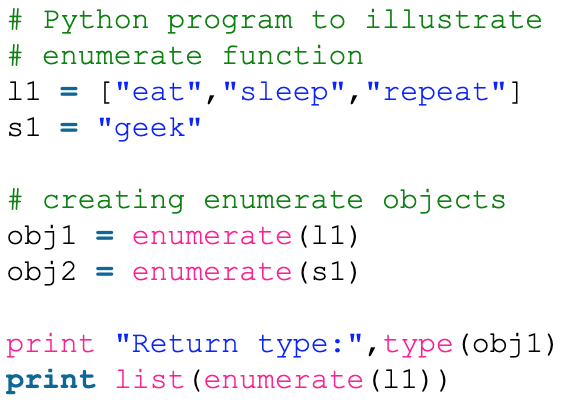
ws.cell(row=i+1, column=j+1).value = value

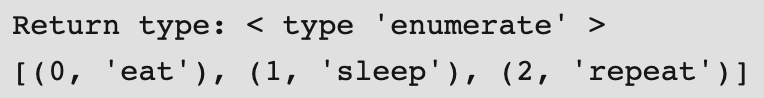
newfilename = os.path.abspath("./output/data\_xml\_to\_excel.xlsx")

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()*:



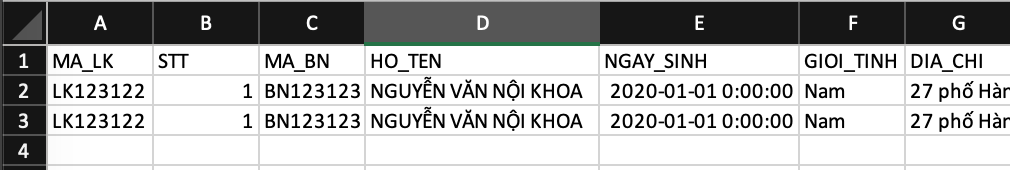
*Hình 3.1. Ví dụ về hàm enumerate()*

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)



*Hình 3.2. Kết quả file data\_xml\_to\_excel.xlsx*

1. **References:**

<https://stackoverflow.com/>

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

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