



X-Rays Tracker App Notice

Introduction

This app aims at making measurement tracking way easier by providing automatically built tables. Some improvements can be added to this code, which is why this notice was created. It's what makes the code reusable and updatable: explaining the code to its' users enables it to be permanent and useful in a long-term.

I'll explain the different steps to how I constructed the App and what's important to have in mind regarding the different studies, when trying to modify the code.

Table of Contents

PRESENTATION.....	2
I. SCREENING THE DIRECTORY	2
1. OVERALL PRESENTATION	2
2. RUNNING THE SCREENING DIRECTORY CODE	3
II. CREATING THE APPLICATION	3
1. WHOLE APP BACKGROUND	3
2. OBJECT-ORIENTED PAGES	3
3. PANDASTABLE'S DATAEXPLORER	4
4. PROVIDING THE NEEDED INFORMATION	5
III. DATABASE HANDLING	5
1. OPENING THE DATABASE	5
2. SAVING THE DATABASE	5
3. REFRESHING THE DATABASE	6
IV. PLATFORM PERSONALIZATION	6
1. HOW TO CHANGE SOME APP'S FEATURES	6
2. ARCHITECTURE OF THE PLATFORM	6
V. FROM PYTHON TO .EXE	7
VI. MIGHT BE USEFUL	8
VII. IMPROVEMENTS IDEAS.....	8
ANNEXES	9
ANNEX 1 : CODE TUTORIALS USED :	9
ANNEX 3: UPDATES FROM V0 TO V4 (LAST VERSION)	9
ANNEX 4: LAST CHECK – AUGUST 16 TH , 2019	9
ANNEX 5 : USER GUIDELINES	10

Table of Figures

<i>Figure 1 : Studies coding differences</i>	<i>2</i>
<i>Figure 2 : Database creation</i>	<i>3</i>
<i>Figure 3 : X Rays to verify- screenshot.....</i>	<i>4</i>
<i>Figure 4 : Study page.....</i>	<i>5</i>
<i>Figure 5 : App architecture.....</i>	<i>7</i>
<i>Figure 6 : Error Sources</i>	<i>8</i>



Presentation

Context

The Spine Research Lab's work and data depends on a large part on the X-Rays. The X-Rays provide the data for the SPSS database, which enables the researchers to access and share data. Until now, everything is kept into a huge Excel file per Study and tracking X-Ray measurement can be really complicated because of the different follow-ups of the different types of X-Rays. The main aim is to know the status of each study and what the different priorities are.

The aim of this App is to provide a list of all available X-Rays, their status (to be checked, verified, completed or missing), and all information needed for the treatment, like the path, the site name and the date (could be the OR Date for operative patients, or the Date of 1st Entry for non-operative patients).

Each X-Rays have special characteristics: Study – Site – Patient ID – View – Time Index

Those characteristics can be extracted from each file name, yet each study has its' own way to 'code' that set of information.

I. Screening the directory

1. Overall presentation

For each Study, in order to know the available files depending on their format, the computer screens the whole folder to create the database.

This process is adapted to each study, as each of them has its' special folder organization and special way to name files. **Those differences had to be taken into account for every step of the App creation.** Figure 1 shows the main differences between the studies:

		PON	PEED	ScoliRisk	CTJ	NiH	MESA	PCD	Supine
	Directory	Study > Site> PON > O/NOp > Patients folders	Study > site >Patients folder	Study > site >Patients folder	Study > Patient s Folders	Study > site >Patients folder	Study > site >Patients folder	Study > site >Patie nts folder	Study > site >Patients folder
	File name example	WUNPONY0081C S (I-01) 6wks.lat1	BAV-001 (I-00) base.2014.01.22.ap	BAU-018 (I-07) 2YR.2014.01.23.lat 1-2	NYUCT J0001-RZ (I- 05) 1YR.cs.lat1	HS008 (I-00) baseline yyyy.mm.dd.ap	08-011 (I-00) base.2016.01.14.bendl.xls x	BSCPCDY0 002-DR (I- 00) base.fs.lat 1	HSSVS0302- ND (SUPINE) supine.2017.1 2.21.lat1
	View	.view or .typeview.view							
	Time index	(I-xx) and wk nbr							
Date	Where	Excel Imported	Name	Name	Excel Not Import ed	Name	Name	Excel Not Import ed	
	Consequence	<ul style="list-style-type: none"> If in name, the saved 'last update' doesn't contain the date as a column : it's in the file name If not (in excel), it saves the date every time it is entered in the table 							

Figure 1 : Studies coding differences



The process is then done by study, with a function per study, the name of the function goes this way: **Refresh_{StudyName}**

Steps

- 1) Browse the sites
- 2) Browse the patients' folders per site
- 3) Browse each patient folder to see how many formats exist per file (num, tif, xls, dcm for the same name)
- 4) Compare the file name with existing saved database with comments and validations
- 5) Merges the refreshed database with its comments and validation

Figure 2 : Database creation

2. Running the screening directory code

The python file runs the different screening directory files.

It creates a function to execute each of the studies' screening codes. This same function is run at the App's running, and when the table aims at being refreshed: **Refresh_{StudyName}**

II. Creating the Application

1. Whole App background

To create the App I chose the Tkinter module because it's easy to be use to and it provides everything needed for a classic app. Qt was an option but couldn't be downloaded on the computer due to a lack of capacity.

The apps' display parameters details are available in **Platform Personalization**.

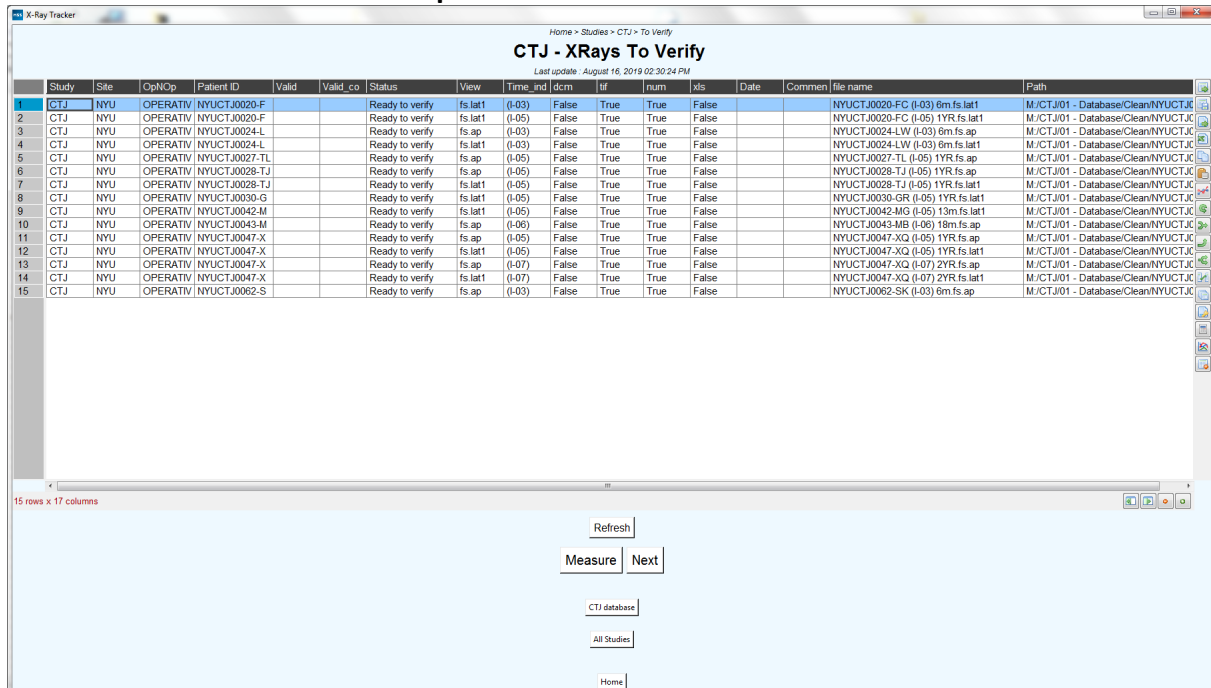
All the pages are created when the app runs, as well as the Tables. The tables have a refresh button

2. Object-Oriented Pages

The different pages are all created as classes, which makes it way quicker and easier to open, but also more complicated to store into lists.

Each class has its own variables and can display buttons and labels. Variables from other classes are not recognized unless they are marked as global before they are defined.

3. Pandastable's DataExplorer



The screenshot shows the Pandastable DataExplorer interface. The title bar indicates the current view is 'CTJ - X Rays To Verify'. The table below lists 15 rows of data, each representing an X-Ray study. The columns include Study, Site, OpNOp, Patient ID, Valid, Valid_co, Status, View, Time_ind, dcm, lat, num, xls, Date, Comment, file name, and Path. The 'Status' column for all rows is 'Ready to verify'. The 'View' column shows 'fs lat1' for most studies and 'fs ap' for others. The 'Time_ind' column shows values like (0-03), (0-05), (0-06), (0-07), and (0-08). The 'dcm' column shows 'False' for most studies and 'True' for others. The 'lat' column shows 'True' for most studies and 'False' for others. The 'num' column shows 'True' for most studies and 'False' for others. The 'xls' column shows 'False' for most studies and 'True' for others. The 'Date' column shows '1YR fs lat1' for most studies and '2YR fs lat1' for others. The 'Comment' column shows 'Ready to verify' for most studies and 'Ready to verify' for others. The 'file name' column shows the file name for each study. The 'Path' column shows the path for each study.

Study	Site	OpNOp	Patient ID	Valid	Valid_co	Status	View	Time_ind	dcm	lat	num	xls	Date	Comment	file name	Path
1	CTJ	NYU	OPERATIV	NYUCTJ0020-F		Ready to verify	fs lat1	(0-03)	False	True	True	False			NYUCTJ0020-FG (0-03) 6m fs lat1	M:/CTJ/01 - Database/Clean/NYUCTJ
2	CTJ	NYU	OPERATIV	NYUCTJ0020-F		Ready to verify	fs lat1	(0-05)	False	True	True	False			NYUCTJ0020-FG (0-05) 1YR fs lat1	M:/CTJ/01 - Database/Clean/NYUCTJ
3	CTJ	NYU	OPERATIV	NYUCTJ0024-L		Ready to verify	fs ap	(0-03)	False	True	True	False			NYUCTJ0024-LW (0-03) 6m fs ap	M:/CTJ/01 - Database/Clean/NYUCTJ
4	CTJ	NYU	OPERATIV	NYUCTJ0024-L		Ready to verify	fs lat1	(0-03)	False	True	True	False			NYUCTJ0024-LW (0-03) 6m fs lat1	M:/CTJ/01 - Database/Clean/NYUCTJ
5	CTJ	NYU	OPERATIV	NYUCTJ0027-TL		Ready to verify	fs ap	(0-05)	False	True	True	False			NYUCTJ0027-TL (0-05) 1YR fs ap	M:/CTJ/01 - Database/Clean/NYUCTJ
6	CTJ	NYU	OPERATIV	NYUCTJ0028-TJ		Ready to verify	fs ap	(0-05)	False	True	True	False			NYUCTJ0028-TJ (0-05) 1YR fs ap	M:/CTJ/01 - Database/Clean/NYUCTJ
7	CTJ	NYU	OPERATIV	NYUCTJ0028-TJ		Ready to verify	fs lat1	(0-05)	False	True	True	False			NYUCTJ0028-TJ (0-05) 1YR fs lat1	M:/CTJ/01 - Database/Clean/NYUCTJ
8	CTJ	NYU	OPERATIV	NYUCTJ0030-G		Ready to verify	fs lat1	(0-05)	False	True	True	False			NYUCTJ0030-GR (0-05) 1YR fs lat1	M:/CTJ/01 - Database/Clean/NYUCTJ
9	CTJ	NYU	OPERATIV	NYUCTJ0042-M		Ready to verify	fs lat1	(0-05)	False	True	True	False			NYUCTJ0042-MG (0-05) 13m fs lat1	M:/CTJ/01 - Database/Clean/NYUCTJ
10	CTJ	NYU	OPERATIV	NYUCTJ0043-M		Ready to verify	fs ap	(0-06)	False	True	True	False			NYUCTJ0043-MB (0-06) 18m fs ap	M:/CTJ/01 - Database/Clean/NYUCTJ
11	CTJ	NYU	OPERATIV	NYUCTJ0047-X		Ready to verify	fs ap	(0-05)	False	True	True	False			NYUCTJ0047-XQ (0-05) 1YR fs ap	M:/CTJ/01 - Database/Clean/NYUCTJ
12	CTJ	NYU	OPERATIV	NYUCTJ0047-X		Ready to verify	fs lat1	(0-05)	False	True	True	False			NYUCTJ0047-XQ (0-05) 1YR fs lat1	M:/CTJ/01 - Database/Clean/NYUCTJ
13	CTJ	NYU	OPERATIV	NYUCTJ0047-X		Ready to verify	fs ap	(0-07)	False	True	True	False			NYUCTJ0047-XQ (0-07) 2YR fs ap	M:/CTJ/01 - Database/Clean/NYUCTJ
14	CTJ	NYU	OPERATIV	NYUCTJ0047-X		Ready to verify	fs lat1	(0-07)	False	True	True	False			NYUCTJ0047-XQ (0-07) 2YR fs lat1	M:/CTJ/01 - Database/Clean/NYUCTJ
15	CTJ	NYU	OPERATIV	NYUCTJ0062-S		Ready to verify	fs ap	(0-03)	False	True	True	False			NYUCTJ0062-SK (0-03) 6m fs ap	M:/CTJ/01 - Database/Clean/NYUCTJ

Figure 3 : X Rays to verify- screenshot

The DataExplorer enables multiple choices as commenting, validating X-Rays, but also sorting and modifying the database.

Uploaded:

- From a dataframe

```
pt=Table(dataframe=dataframe)
pt.show()
```

- From a csv

```
pt.importcsv(file.csv)
```

Saved:

```
pt.doExport('targetedfile.csv')
```

4. Providing the needed information

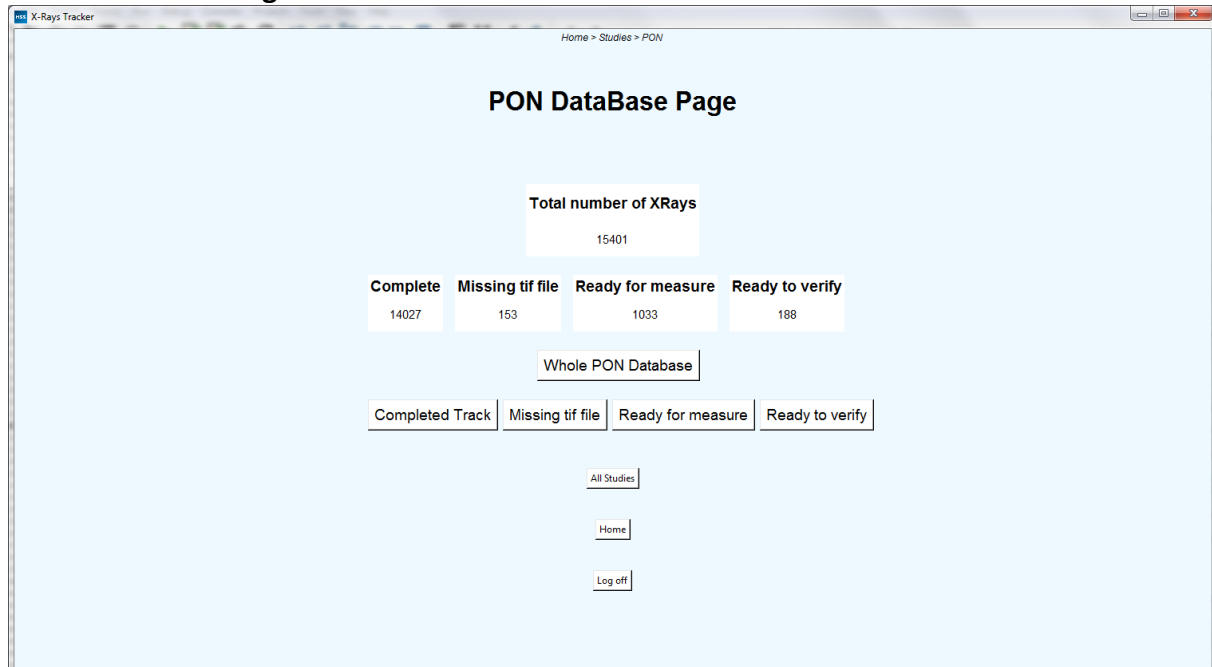


Figure 4 : Study page

Each study page provides:

- Overall number of X-Rays
- Number of completed track
- Number of missing file
- Number of X-Rays ready to be measured
- Number of X-Rays ready to be verified
- Number of Unvalid X-Rays
- Number of problematic X-Rays

III. Database handling

1. Opening the database

First opening:

Focuses on the last database created:

```
pt=Table(dataframe-dataframe)
pt.show()
```

Refresh:

```
pt.importCSV(file.csv)
```

2. Saving the database

Saving button for the raw database

- 1) Saving the table into an ephemere csv file: `pt.doExport('ephemere.csv')`



- 2) Save ephemere into a database
- 3) Filter and treat that database and save it into the CSV file
- 4) Open that same ephemere csv file and show it on the Table

The program uses **pandastable** Tables in order to display the different dataframes but when modifying the table, it modifies the table and not the dataframe. The Table cannot be filtered so that's why we need to go through all those steps in order to save the whole modified table and not the dataframe originally displayed.

3. Refreshing the database

Refresh the database, which means screening again the files: **see Figure 2**
The program uses **pandastable** Tables in order to display the different dataframes but when modifying the table, it modifies the table and not the dataframe. The Table cannot be filtered so that's why we need to go through all those steps in order to save the whole modified table and not the dataframe originally displayed.

IV. Platform Personalization

1. How to change some app's features

- All the text is in Arial font
- Background color : <https://htmlcolorcodes.com/fr/> check this website for more choices
- Buttons color :
 - Background
 - Text Color
- Time format used is the following :
 `strftime('%B %d, %Y %H:%M:%M:%S %p')` : August 08, 2019 02:59:22 PM
 <http://strftime.org> to modify the **time format**
- TITLE FONT = can be changed too

2. Architecture of the Platform

The platform creates all pages at the time it runs. Databases displayed and time of last update are the only refreshed objects.

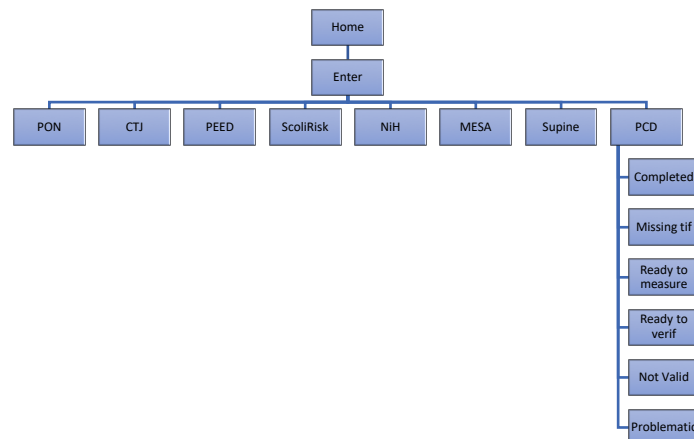


Figure 5 : App architecture

V. From python to .exe

In order to create an executable application, python and its' packages need to be on the environment path -which requires an admin access-. Pip is particularly needed. If not recognized, can be implemented by:

- Going to the Python official website > Downloading the executable installer > When opening it, choose the 'modify' option: it will reboot the packages and make sure their path is in the environment. [Click here for more details.](#)

Three ways can be found on the internet to transform a python file into an app:

- **Pyinstaller:**
 using `Pyinstaller -n XRayTracker -i icoco.ico xraytracker.py`
 This automatically creates the app. If the app created opens and closes itself, that means there's a code error. The solution is to run it and read the error raised.
 auto-py-to-exe can also run and create automatically the **Pyinstaller** code:
`pip install auto-py-to-exe`
`auto-py-to-exe`
- **Cx_freeze:**
 Using `python setup2.py build`
- **Py2exe:**
- only for python versions before 3.4: **this solution doesn't exist anymore** (tuple error)

Pyinstaller was the quicker and raised less errors than cx_freeze.

Python 3.7 has some errors with the last numpy version (`numpy.random.common` not found) : using **Python 3.6.8** can be a solution only if **TCL 8.6.9** is used, it won't work with the 8.6.6 version of TCL, which is the one downloaded with python 3.6.8. TCL is the package associated with Tkinter. This error is very recent and has been transmitted by Python's users to Python mid-July 2019

Uninstalling all python versions and reinstalling only the last one, then reinstalling each module is what worked in my case. Running the code depends on the previous downloaded packages and anything can create an error.

VI. Might be useful

Other codes were used in order to create a fully representative database:

- **df_displayer.py** : displays the database you want, useful to have a quick idea of how many columns, rows there are, save it into csv or excel... anything is possible with that table.

The databases were created on the current architecture: If it has come to change, some errors can happen with the App. This table sums up all the possible error that can occur

Study concerned	Source
CTJ	The CTJ database has been created on the folder directory available: the only site was the NYU and so then it was no 'site folders', so everything is built for NYU. Any further change of the directory will create an error
Studies databases in general	The main weakness of the code is that it creates the database from the existent directory but is not adaptable. It doesn't identify the difference between files and folders and just treats it as it was defined. If a folder is created that changes the order, everything could be blocked and the app won't run

Figure 6 : Error Sources

VII. Improvements ideas

Some more ideas to have a more complete and performant app.

Database features

- Add a filtering feature base on the Table given by Renaud : it will identify each study and set its' parameters : the app also gives the non-compulsory X-Rays information,
- Be able to say when an X-Ray is missing (depending on Study parameters and available files) and provide a list of missing X-Rays per Site

App features

- Find a way to display the *showstatusbar* and *showtoolbar* from **Pandastable** to **Tkinter**
- Rebooth the user interface but making it depend on user's authorizations, and a signature available – to know measured/verified what
- Measurement planning and measurement report for users



	To Measure	X	X	X	X	X	X	X	X	problematic but better if fixed
	To Verify	X	X	X	X	X	X	X	X	
	Problematic	X	X	X	X	X	X	X	X	
	Not Valid	X	X	X	X	X	X	X	X	
Code check	All uniformized, except for the saving process that adapts to each study.									

Annex 5 : User guidelines

User Guidelines has been saved in the same directory as the app. It provides useful information for anyone wanting to use the app.