Document contains instructions on how to use executable and edit the python code.

Contents of folder:

- Executable

- Python Code Folder

- This Document

Using the Executable:

The executable is a self-contained application and doesn't require any accompanying files or folders. However, the desktop image for the exe file is not compiled into the program and is considered an external dependency (the image is located in the python code folder). It's worth noting that this image is not necessary for the program to run; the executable remains fully functional on its own. (Currently supports conversion of .mat and .img files)

The .img files require a txt file be present in the same directory acting as the image header info. At minimum this file must contain x\_dim, y\_dim, z\_dim (columns, rows and number of slices)

Note: The DICOM UIDs generated for the converted files are random and serve no specific functional purpose other than meeting DICOM format requirements. These DICOM files are primarily intended for viewing in a DICOM viewer.

1. Initiating the Program:

* To start the program, simply double-click the executable. Please note that it may take a moment to initialize.

2. Using the User Interface:

* A user interface (UI) will appear on the screen. Click the "Select Files" button to choose the files you wish to convert. If you would like to specify the directory to save then then click the save location button. Then, when you're ready to proceed, click the "Convert" button.

3. Upon Completion:

* After the program has finished its task, you will find a folder in the same directory as the selected files, unless you have specified the directory to save the files, the you will instead find them in that directory. This folder will contain DICOM files converted from the ones you selected. Additionally, there will be a "logs.txt" file that provides information about any files that were skipped and the reasons for this.

4. Converting More Files:

* If you need to convert more files, simply select a new set of files, and the conversion process can be initiated again.

Adding/Editing the Code (Python 3.9):

The source code used to create the executable is contained in the "Python Code" folder. It is important to note that any changes made to the code won't automatically affect the executable; recompilation is required. If you intend to edit or run the Python code, follow these steps:

1. Ensure that you have the Python runtime installed on your system and are using python 3.9 (some modules are not supported yet in newer python versions).

**For best practice:** Copy the code to a different directory. It's recommended to create a Python virtual environment, though it's not mandatory. When editing the code, ensure you work within the new folder where the virtual environment is active. After testing any edits, replace the old code with the newly edited version.

3. Activate the virtual environment through the command prompt and execute "pip install -r requirements.txt" to install the required dependencies. Or simply run "pip install -r requirements.txt" in the regular command prompt if you are not using a virtual environment

Adding to the Code Base:

The core script within this codebase is "main.py," and you'll use "convert2exe.py" for program compilation. During compilation, the "hook-pydicom.py" file plays a crucial role (for detailed information, consult the pyinstaller documentation).

Additionally, there's a "DICOM modules" folder, which contains classes with valuable methods for working with DICOM files. The "DicomProcessing" object is currently the sole object in use. It can be found in "DicomModules\DICOM\_Objects\Base\_Class" and is a subclass of the pydicom.FileDataset object.

You'll also find a "UI" folder. The graphical user interface (GUI) design adheres to the MVC (Model-View-Controller) object-oriented pattern. Consequently, there are separate folders for Views, Controllers, and Models. This organization greatly enhances the program's structure and expandability.

To Add a New View:

1. Create a new Python file in the "Views" folder.

2. Define a class that inherits from ViewABC (check existing views for examples).

3. Define the method "\_place\_widgets(self)" in this class. Place all code to arrange custom tkinter widgets here. You can add other methods for displaying error messages, but keep the view devoid of logic.

To Add a New Controller:

1. Create a new Python file in the "Controller" folder.

2. Define a class that inherits from "ControllerABC" (refer to existing controller files for examples).

3. Define the method "\_bind(self)" in this class to bind controller methods to view widgets. This method can simply contain return/pass if there is no desired binding.

4. Create additional methods to handle any necessary preprocessing logic before sending data to the model for the primary task.

To Add a New Model:

* Create a new Python file in the "Model" folder and define a class.
  + In this class, you should implement the core functionality and processing required for the desired feature.
* Note: If your goal is to add a new type of conversion and not a completely new model, you can do so without creating a brand-new model. Follow these steps instead:
  + Create a new file within the "UI\Conversions" path.
  + In this new file, create a class that inherits from the existing Converter class located in "UI\Conversions\ABC."
  + Define a method named "Convert(self)" in your new class, which should contain the logic for generating a pixel array as a 2D numpy array.
  + Use the "WritePixelArray2Dicom" method from the Converter class to write the pixel array to a DICOM file.
  + In the Conversion Model file, add the file extension of the new conversion type to the "convertableFiles" class variable.
  + Also, add the file extension and a lambda function that instantiates the object as a key-value pair in the "converterObjDict" dictionary, respectively.

Finally, Update "main.py":

- Add your new view, controller, and model (without instantiation, i.e., just the class reference) to the Views dictionary.

- To connect your view, call the "SwitchView" method from one of the previously created controllers (not the one you just created). This will link the view, controller, and model you've created into the program's structure.