# ICD10CM Parser

*Application in the real world:*

ICD10CM (International Classification of Diseases – Clinical Modification) is a clinical vocabulary that has over 95,000 medical diagnosis codes, and is widely used in health IT. It is used in every healthcare setting ranging from EHRs, Health Insurance companies to IT companies managing healthcare data. ICD10CM vocabulary enables semantic interoperability between two disparate systems for effective and efficient sharing of disease information between health systems. Every October, the latest version of the vocabulary is released which includes the addition of new codes (2022 was all about COVID-19 related codes), and the deprecation of codes that are no longer relevant.

*Downloading the ICD10CM current release:*

Refer to this link - <https://www.cms.gov/medicare/icd-10/2022-icd-10-cm>. Download this zip file - 2022 Code Descriptions in Tabular Order - Updated 02/01/2022 (ZIP). Use this file as the input to the program - icd10cm\_order\_2022.txt. The file has no headers. The maximum length of a line is 400 characters.

*What ICD10CM Parser does:*

Diagram

Description automatically generated

The ICD10CM release is in the form of a large positional flat file containing positional records. The file has 95,000+ rows and is of the size of about 14 MB. We need to parse the data in the file, clean up and validate it before it is loaded into the database. Here are the potential problems that the data can have, and how the program handles them:

|  |  |
| --- | --- |
| **Potential problem** | **How ICD10CM Parser handles the problem** |
| ICD10CM codes not conforming to the standard format | Using Regular Expressions, such rows are caught as bad data |
| Fields having leading, trailing, or embedded whitespace | They are caught and removed as part of clean-up |
| Some fields enclosed in quotes | These are caught and removed as part of clean-up |
| Positional file is hard to import into database | Converts positional to comma-delimited file |
| Rows having non-printable characters | Using Regular Expressions, such rows are caught as bad data |
| Some rows not conforming to the field positions | Regular expressions catch these as well |

In addition to the above, ICD10CM Parser also has the capability to generate the Release notes for the release being parsed. This requires that the previous release (already validated) exists in the directory. The release notes compare the validated current release data with the previous release’s data and publish the list of new codes and deprecated codes of the current release.

*Instructions to run the program:*

The program, program.py imports the module ICD10CM\_Parser.py. Program.py has two unit tests under name==main. The program does not need any user input and is self-sufficient to run and generate output.

*Output:*

* Scenario 1 - When there are no validation errors: The output will have two text files (a) Validated, comma-delimited rows in icd10cm\_validated.txt (b) ReleaseNotes.txt
* Scenario 2 – When there is a bad ICD10CM in the current release: The output will have only one text file with the rows having the bad data - icd10cm\_error\_rows.txt

*Course requirements for the project code:*

4 container types (list, tuple, set AND dictionary): Done

* List - validated\_output\_list
* Tuple - list\_of\_tuples
* Set - validated\_code\_set
* Dictionary - \_\_ICD10CM\_col\_pos\_dict

1 iteration type (for, while) – program.py – line 32 - Done

1 conditional (if) – program.py – line 145 – Done

1 try block with an else condition – ICD10CM\_Parser.py – line 135 - Done

1 user-defined function that accepts parameters/arguments and/or returns a value – Program.py – line 42 – Done

1 input and/or output file (include input data with your project) – two input files - icd10cm\_order\_2022 and icd10cm\_order\_2021 included in the folder - Done

1 user-defined class – ICD10CM\_Parser.py has 2 classes - Done

The class must be imported by your main program from a separate file – Program.py import ICD10CM\_Parser – Done

The class must have the following required structures (ICD10CM\_Parser.py):

* at least 1 private attribute - \_\_ICD10CM\_headers\_tuple. Has get/set methods to access - Done
* 2 public self class attributes - self.file, self.record\_list - Done
* at least 1 private method - \_\_record\_cleaner() - Done
* 2 public class methods that take arguments, return values and are used by your program – remove\_text, append\_text - Done
* init() class method that takes at least 1 argument – Done
* a repr() or str() class method – \_\_repr\_\_ implemented - Done
* a magic class method (not one of the methods listed above) - \_\_del\_\_() - Done
* Provide 2 unit tests that prove two of your public class methods work as expected. The tests should evaluate results using assert statements – Unit tests call user-defined functions in Program.py which call the public class methods from ICD10CM\_Parser.py - Done