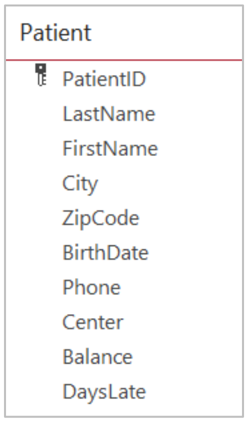
Program Specification

- *Main objective: Victorino Health* operates multiple clinics in central Texas: North Waco Clinic –#10, West Waco Clinic –#20, and Hewitt Clinic –#30. They are requesting that you write a program that allows their employees to enter new patient data, update and delete patient data, as well as write a report that filters and sorts patient information. The application must be menu-driven as specified in the requirements below. A CSV file containing Patient data as well as a backup file has been provided for downloading in Canvas. The Patient entity/class (shown on the right) contains the attributes (i.e. fields) in this file.



Read before starting

- I recommend that you read the entire assignment before beginning.
- The expectation is that students have read the *entire* chapter and attended *all* lectures. Students must demonstrate the ability to code using statements and functions as taught in class and the textbook.
- **You must use the coding concepts as taught in this course**, and follow the programming standards as documented in Canvas > Modules > Help> **"Help - Class Coding Standards"**.
- Periodically review the ***Assignment Tips & Updates*** page in our module looking for tips as well as any ***required*** updates to this assignment.
- **All of your code for this assignment must be written in functions.**
- **Globals** - global constants are permitted but global variables are not.
- You will be graded based on the quality of the application including the code, data entry, output, and usability. You will also be graded on whether you used all of the major concepts we have covered – see chapter names for guidance (e.g. loops, lists, etc.)
- This code must be free of syntax errors; otherwise, it will not be graded. Thus, comment any code that crashes, or for a better grade, improve the code and submit it one day late with a 10% deduction.

Instructions

1. *Download the files*
   - Download your Canvas assignment instructions and data files to the same folder on your computer/cloud**:** patients.csv, and patients-BKUP.csv. You must write & execute your code from this same folder.
     - The backup file should never be updated as it serves as your original backup data.

2. *At the top of your code*
   - Add the usual program documentation header at the top of your code.
   - Add code to import required modules.
   - Add 2 global constants at the top of your code for the two CSV files provided: patients.csv and patients-BKUP.csv. Since we are using global constants for the file names, file names should not appear in the rest of your code as you will be using these constants. Please note that this BKUP file is not the same as the TEMP file you will be creating when updating/deleting records.
   - You will have a main() function and 5 other functions to handle the main 5 menu options. These functions will not have parameters.

3. *Menu*
   - Your menu will consist of the following options: 1 – View Patient Details, 2 – Add New Patient, 3 – Update Patient, 4 – Delete Patient, 5 – Display Patient Listing, 9 – Restore Patient Data - for testing only, X – Exit. Put some type of separator between options 5 and 9 so the user doesn't accidentally pick the last 2 options (i.e. 9 and X).
   - Once an option is selected, process the selection as explained in the following steps, and re-display the menu.

4. *Option 9 - Restore Patient Data*
   - **Code this option before the other options!** Option 9 will allow you to restore the patient file from the backup file in the event that you corrupt the file when testing option 2 (add a new patient).
   - For option 9, copy the following two lines of code within the menu code, and add a message telling the user that the data was restored from the backup file. The following code means this:
     - Line 1: from the python module called *shutil*, import only the *copyfile()* function.
     - Line 2: copy from a *source* file (e.g. bkup) to a *destination* file (e.g. dest) - i.e. restore our data. Replace these 2 sample variable names (in red) below with your global constant names defined at the top of your code. The backup file should never be updated as it serves as your original data backup.
     - Note: this is the only code needed for option 9!

```
from shutil import copyfile
copyfile(bkup, dest)
```

5. *Fn1 - View Patient Details*
   - This feature should allow a user to lookup up all data for a *single* patient. First, display a banner with an appropriately labeled title for this option. Allow the user to specify a patient ID. List the patient's data vertically (i.e. on different lines), appropriately labeled and values aligned & formatted for readability. For the center, display both the center number & center name on one line (as described on page 1).
   - Halt execution so the user can view the data before returning to main() to re-display the menu.

6. *Fn2 - Add New Patient*

   Tip: see lecture notes on how to do *add a new record in a file.*

   - This feature should allow a user to enter all data for a new patient (and add it to the bottom of the existing file). First, display a banner with an appropriately labeled title for this option.
   - Allow the user to enter all data for the patient with the exception of patient id, balance, and days late as the latter two will be pre-initialized to 0. Make sure to validate all data entry as all values are required. This includes testing for valid numeric input, dates, etc. Zip code is 5 digits and can have leading zeroes which must be preserved. Due to the nature of this interface, the user will type hyphens with the phone number: 254-555-5555. For birthdate, you may separate the data entry into 3 separate fields but you must produce a result as mm/dd/yyyy before saving the record; however, a better approach is to use Python's split() string method – i.e. a_list = any_str.**split**(delimiter). This splits the text using the specified delimiter string character and splits any_str into different elements in a 1D list. Remove any leading zeros the user may have entered on month & day. If the center number is invalid, display a list of choices properly labeled. The patient ID must be auto-generated and set to the next number after the highest patient ID in the file – note: you must not hardcode this as we will use another dataset to test this with different IDs. Make sure to save any text case data entry (lower/upper/mixed) appropriately.
   - Notify the user that the record was successfully saved and include the new patient ID in the message (so a user can look them up later in option 1 if desired).
   - Halt execution so the user can view the message before returning to main() to re-display the menu.

7. *Fn3 - Update Patient*

   Tip: see lecture notes on how to do *update a record in a file.*

   - This feature should allow a user to update any or all data values for an existing patient (except the primary key – i.e. Patient ID). An important requirement is to not require the user to update every value. First, display a banner with an appropriately labeled title for this option.
   - Prompt the user for the patient record that is to be updated (i.e. ask for the patient ID).
   - Notify the user if the record was not found as well as notify the user if the record was successfully updated.
   - Validation is not required on this functionality.
   - Halt execution so the user can view the message before returning to main() to re-display the menu.

8. *Fn4 - Delete Patient*

   Tip: see lecture notes on how to do *delete a record from a file.*

- This feature should allow a user to delete any existing patient.
- Prompt the user for the patient record that is to be deleted (i.e. ask for the patient ID).
- If the record is found, display the patient's data and confirm that the user wants to delete the record.
- Notify the user if the record was deleted or if the record was not found.
- Validation is not required on this functionality.
- Halt execution so the user can view the message before returning to main() to re-display the menu.

9. *Fn5 - Display Patient Report*

- The ***report*** should display *all* patient data filtered & sorted as specified by the user. First, display a banner with an appropriately labeled title for this option.

- **Add Multi-Field Filter Capability**

  Tip: see lecture notes on how to do *multi-field filtering (with optional filters)*.

  - Provide a shorter banner labeled "FILTER OPTIONS"
  - Provide the following *multi-field **filter options***. Allow the user to enter values for as few or as many filter options as they desire. Make sure to handle any text case data entry.
    - Last name – pattern match – e.g. "san" displays all names that start with those 3 characters – see out Ch8 slides
    - First name – pattern match
    - City – exact match
    - Birth date – exact match
    - Center number – exact match
    - Balance (min) – range match – e.g. all balances that meet this *minimum* requirement
    - Balance (max) – range match – e.g. all balances that meet this *maximum* requirement
    - Days late – exact match
  - Note: you are not required to validate any of the filter input

  - Display the *total number of records* that met the filter criteria; however, do not display any patient records yet. If the total is 0, display an appropriate, descriptive yet brief message stating that there was no match.
    - Tip: while testing, you will want to display the records at this point to make sure your filtering is working before proceeding. Then make sure to remove or comment out the testing code.

- Allow the user to quit the processing if the filtered result is large
    - Add a constant at the top of this function called FILTER_PROMPT_MAX and set it equal to 50.
    - Let's now allow a user to stop the process if there are a lot of records (if they so choose). If the number of filtered records is more than the FILTER_PROMPT_MAX, ask the user if they would like to proceed to see the patient report. If they say yes (*or* the filtered records is less than or equal to this value – i.e. there are not a lot of records to display), show them the sort menu and then display the Patient Report (as described below). If they say no, then do not proceed with displaying the sort menu nor the report.

- Add Sort Capability

    Tip: see lecture notes which cover both *single-field* & *multi-field sorting (asc/desc/mixed)*.

    - Display the following ***sort menu options***. Let the user know to select a sort option or to press any key to continue (i.e. no sorting). Display only the text shown in bold below – i.e. do not display the words to the right in blue (but you must achieve the part in blue):
        - **1 – Patient ID** → sort by patient ID ascending
        - **2 – Patient Name** → sort by last name and first name, both ascending
        - **3 – Center** → sort by center followed by last name & first name, all ascending
        - **4 – Days Late** → sort by days late descending then last name asc, then first name asc

    - Sort the filtered data appropriately based on the user's sort option choice making sure to sort by the more detailed specifications to the right in blue above – i.e. If the user selects "2 – Patient Name", you must sort the data by last name and first name ascending.

- Display the Patient Report
    - Add something to clearly separate the filter & sort choices from the actual report. Also, display an appropriate title for the report.
    - Display the final results (i.e. filters & sorted) in tabular form (rows & columns). All columns should be properly labeled with field names, followed by a dashed line under each separate field, and all values perfectly aligned and formatted. Your output must not exceed a max width of 120.
    - Pause execution every **20** records so the user can view the data. When they press enter, it should continue to display the next 20 records.

- At the end of this function, prompt the user if they want to enter new filter criteria. If they respond yes, then re-loop through the code in this function to start over. Otherwise, return to main() to re-display the menu.

Submitting the Assignment

- First, review the ***Assignment Tips & Updates*** page to see if there are any additional requirements to this assignment – denoted with the tag: **[required].**
- To submit the assignment, go to Canvas and open the assignment link.
- Upload your *.py file**.** You may upload more than once as long as it is before the due date & time. Only the final submission will be graded. Assignments will be accepted one day late albeit with a 10% deduction.
- **Note:** You will only submit your program; do not upload any csv files.