CS 2316

Individual Homework 10 – XML Processing

Due: Wednesday November 19th, before 11:55 PM
Out of 100 points

Files to submit: **1. HW10.py**

This is an INDIVIDUAL assignment!

Collaboration at a reasonable level will not result in substantially similar code. Students may only collaborate with fellow students currently taking CS 2316, the TA's and the lecturer. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. You should not exchange code or write code for others.

For Help:

- TA Helpdesk – Schedule posted on class website.
- Email TA's or use T-Square Forums

Notes:

- **Don't forget to include the required comments and collaboration statement (as outlined on the course syllabus).**
- **Do not wait until the last minute** to do this assignment in case you run into problems.

Background:

Baseball. It's America's favorite pastime, and a statistician's dream. You've been inspired by Jonah Hill's character in Moneyball and have decided you want to put your 2316 skills to the test crunching baseball numbers. We've given you actual data on every MLB game played this season (freely available via gameday on mlb.com. You can check out gd2.mlb.com/components/game/mlb to take a look at the data used yourself. The important data is contained in game_events.xml under each game for a given year, month and day). Your task in this assignment will be to parse the given .xml file which contains detailed information of every game down to the pitches in each at-bat, and then process that data to focus mainly on pitching. Your goal will be to output an xml file containing *Pitchers* as the main element instead of g*ames*. The children of *Pitchers* are Pitcher elements. The children of each *Pitcher* will be a number of *PitchData* elements, each with a *pitch*T*ype* attribute representing the type of pitch, and three children, *NumPitched,* A*vgSpeed* and *StrikeToBallRatio* which all have just text elements corresponding to their stats.

Format of Output XML:

```
<Pitchers>
    .
    .
    .
    <Pitcher name=”Julio Teheran”>
        <PitchData pitchType=”FT”>
            <NumPitched>741</NumPitched>
            <AvgSpeed>88.5338731443995</AvgSpeed>
            <StrikeToBallRatio>2.0121951219512195</StrikeToBallRatio>
        </PitchData>
        <PitchData pitchType=”FF”>
            <NumPitched>1246</NumPitched>

            <AvgSpeed>91.33073836276088</AvgSpeed>

            <StrikeToBallRatio>2.1464646464646466</StrikeToBallRatio>

        </PitchData>
        .
        .
        .
    </Pitcher>
    .
    .
    .
</Pitchers>
```

Note: Pitchers should be saved to the XML file in alphabetical order! PitchData elements within each Pitcher element will not have a specified order.

Structure of games data:

You don't need to be familiar with all the rules of baseball to complete this homework. Pull up one of the smaller input files and look at the structure of the *games* xml tree. The g*ames* xml tree contains a lot of noisy (interesting) data, but we only really care about pitchers at the moment. The important element in the tree for our uses is the *atbat* element. There are a number of these elements which are children of the *top* and *bottom* of each *inning*. Fortunately, each *atbat* element contains a *pitcher* attribute which you will use to keep track of which pitcher pitched that *atbat*. Since we are grouping a Pitcher's pitches, we need to use the *pitch* elements inside each at bat to keep track of how many of a certain type of pitch that pitcher has thrown, the speeds at which he throws it, and how often he throws that pitch for strikes/balls. **IMPORTANT:** The *pitch_type attribute* corresponds to the type of pitch that you will need to group a pitcher's pitches (values such as “FF”, “FT”, “SL”, “CH”). The *type* attribute corresponds to whether or not the pitch is a ball or strike. When calculating the strike to ball ratio, consider 6 strikes to 0 balls as having a 6.0 strike to ball ratio. Any pitch thrown that is not a ball is a strike. (Pitches batted in play for example). **HINT:** It is highly recommended that you use dictionaries to do your logic in this stage. Remember that a pitcher can pitch in more than one game so you will have to be able to update his pitch stats accordingly. Don't focus on the games, but rather, the at-bats.

Bad Data:

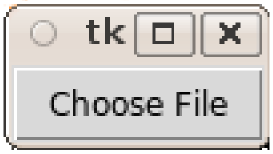
Due to the large scope of the data collected in the full xml file we've given you, there are some problems with the data. Some at-bats will not have values for some pitch speeds. You will need to handle these occurrences by ignoring them. If you cannot convert the pitch speed for a pitch into a float, you should **IGNORE** that pitch along with the ball or strike data associated with it.

Size of Input Data:

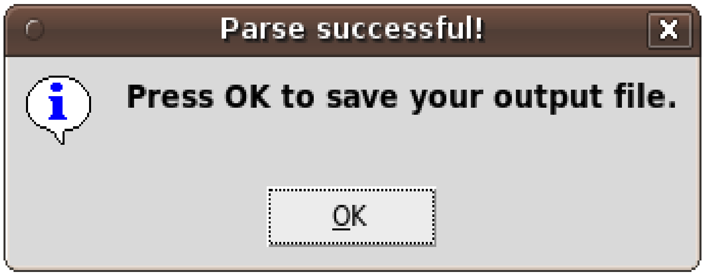
We are providing you some relatively large files to parse in this homework because these are real data sources. You probably will not want to open the “big_games.xml” file in a web browser like Chrome or Firefox. It will most likely crash your browser or take a very long time to load. It's suggested that you use the smaller files (“2innings.xml”, “sample_game.xml”) to observe the structure of the XML and to test and construct your code. When you are confident that your logic is correct, you can then run your code on the bigger files. “big_games.xml” is quite a large file to parse so you should make sure to think about efficiency in your code. Make sure you are using dictionaries and not parsing over the whole xml document more than you need to. Your code should not take an extremely long time to run. If it is taking more than a couple minutes to parse “big_games.xml,” you should look for ways to make your code more efficient. If you're stuck on this, come into the TA Help Desk preferably before the day this homework is due.

A note on checking your output: the sizes of your output files will probably not match the sizes of the output files given to you for checking. This is due to the fact that the output files were written to be readable in a text editor, whereas your xml files will most likely not be. You will have to check your output a different way.

GUI:

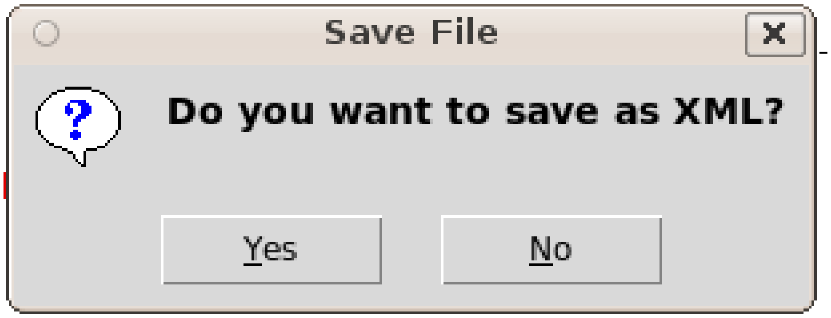


Your GUI should contain a Choose File button which will open up a file dialog for the user to choose the file containing the game data. After the user selects an input xml file, your program should attempt to parse the file given. If there is an error while parsing the file, a warning dialog should display to the user. If successful, the following message box should display to the user:



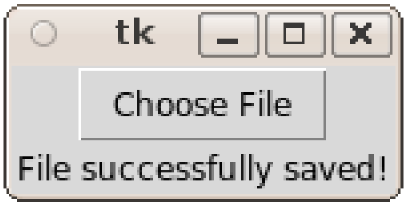
Press OK to save your output file.

Pressing the OK button will display a confirmation dialog which will look like the following.



Do you want to save as XML?

If the user presses 'Yes,' a *file save* dialog should display allowing the user to choose where to save the output file. If the user presses 'No,' the GUI should return to the original window with just the Choose File button. The user should be able to enter the name of a file (such as “big_output.xml”) and your program will save the results as an XML file. After the save is successful, you should update the GUI to say “File successfully saved!”.



File successfully saved!

Suggested Outline:

You may use any combination of class/functions/methods you wish to use to do this homework. The following is a *suggested* outline for your program:

__init__: set up the GUI along with any instance variables you might be using. You will want to use a dictionary to store data on each specific pitcher.

clicked: Ask user for their file. Ensure proper error handling. Call *parseGames* to parse the tree and then writePitchers to write to an output file.

parseGames: Loop through all the *atbat* elements in the xml tree and pass each one into *parseAtbat*. After all the elements have been processed successfully, you should display the specified *Data Processing Succeeded* dialog to the user.

parseAtbat: Takes in an *atbat* argument which should be an *xml.etree.ElementTree.Element* and creates/updates the entry for the corresponding pitcher in your dictionary of pitchers. You will want to consider all the pitches in the atbat and whether or not they are balls and strikes. Should be able to ignore invalid pitch speeds. **HINT:** You should not be attempting to calculate the average speed of pitches yet since that can only be done after all atbats are processed.

writePitchers: Use the pitchers dictionary, which should contain a list (OR dictionary) of pitches for each pitcher, to write an output xml file (see format given in introduction). Use the *processPitches* method to process this list of pitchers for a pitcher and return a dictionary of key *pitchType* and value being a tuple (# of this type of pitch thrown, avg speed, strikes:balls ratio) which you will use to make your *PitchData* elements.. Before you write the output file, you should display an askyesno dialog to the user asking whether he wants to save the file as XML. If the user answers yes, display a dialog asking where to save the file. If the user answers no, end the function.

processPitches: Takes in a list of pitches (e.g. (pitchType, pitchSpeed, ballOrStrike)) and populates a dictionary keyed by pitchType. At the end of the function, you should return a dictionary in which the values are (# of pitch thrown, avg speed, strikes:balls ratio) but you may want to use the dictionary as an intermediate step before you try to calculate those numbers. Keep in mind that you first need to separate the pitches out into groups of one pitchType before you can calculate stats on that pitchType. You will probably want to use a slightly different *processPitch* method if you had already put pitches into a dictionary during the *parseAtbat* method. Remember to handle the case when there have been 0 balls thrown on a pitch. The strike to ball ratio should treat this case as the same as if there had been 1 ball thrown. Note as well that the *type* attribute for a *pitch* element in the *games* xml tree can be “S”, “B”, or “X” corresponding to a strike, a ball, or a hit in play. A ball hit in play will also count as a strike for our purposes.

Grading:

You will earn points as follows for each piece of functionality in your code:

GUI:		**15**
-Allows specification of destination file location	5	
-Shows all warning messages correctly and provides success message to user when done	10	
Reading data source:		**10**
-Allows user to select input file	5	
-Proper error handling	5	
Data parsing:		**30**
-Atbats are found and pitcher identified for each	10	
-Separates Pitch elements by pitch_type	10	
-Records pitch speed and type (ball or strike) for each pitch	5	
-Ignores invalid pitch speeds	5	
Data analysis:		**25**
-Computes average speed of pitch correctly	10	
-Computes strike to ball ratio correctly	10	
-Computes number of times pitched correctly	5	
Data output:		**20**
-Data outputted in correct format	15	
-Pitchers written in alphabetical order	5	
Total possible:		**100**