Python Time Processor

Summary

Create a Python program (*hw2.py*) that processes times.

All times in the program are represented by (hour, minute, AM/PM) tuples. Hours and minutes are represented using integers and AM/PM is represented by strings (either AM or PM). Example: The time 4:12 PM is represented by the tuple (4, 12, 'PM').

Directions

You must create the following functions and top-level functionality in the file *hw2.py*.

create_time_list

Parameters: Name of the input file (string)
Returns: A list of time tuples. Each time tuple has three elements in this order: (hours–integer, minutes–integer, AM/PM–string).
Raises: The function is also capable of throwing exceptions, `EmptyFileError`, `ImproperTimeError`, or `FileNotFoundError`.
Description: Reads a file (the name of the file is stored in the parameter, `filename`) consisting of times and returns a list of time tuples. Each line in the file represents a single time and the order of the items in the list must match the order in the file.

A valid line representing a time consists of these three fields, in this order, separated by spaces:

1. hour (integer from 1-12)
2. minute (integer from 0-59, with leading zeros such as 09)
3. AM/PM (string, either AM or PM in uppercase letters only)

The line is invalid if any of the following are true:

- incorrect number of fields
- a field has the wrong type
- a field has the correct type but an unacceptable value (such as 0 for hours)

This function must also perform error checking by throwing the following programmer-defined exceptions:

- `EmptyFileError`: The file is empty.
- `ImproperTimeError`: This exception is raised in any of the times are invalid.

The function also can throw the system-defined exception `FileNotFoundError` if the file does not exist. This exception is automatically raised by `open` so no additional work by the programmer is necessary.

Other exceptions may be possible but they are not relevant for this assignment.

IMPORTANT: This function only throws the exceptions. It does not catch them. The exceptions are caught in the top-level functionality.

time_compare_gen

Parameters: *time_list* –list of time tuples from *create_time_list*; *target* –a single time tuple
Returns: This function is a ***generator function*** that will ***yield***, for each time tuple in *time_list*, a tuple that indicates how far in the future it is from *target*.
Description: The yielded tuples will contain two integers in this order: (hours, minutes).
Examples: Assume the target is: `(4, 12, 'PM')`:

Time Tuple	**Difference**	
(4, 13, 'PM')	(0, 1)	1 minute in the future
(6, 20, 'PM')	(2, 8)	2 hours and 8 minutes in the future
(4, 12, 'AM')	(12, 0)	12 hours in the future
(4, 11, 'PM')	(23, 59)	23 hours and 59 minutes in the future
(4, 12, 'PM')	(0, 0)	now

Top-Level Functionality

This program has the following top-level functionality:

1. Get the name of an input file from the command line (using *sys.argv*). WARNING: Do not prompt the user for a file name.
2. Call *create_time_list* using the file and storing its result into *time_list*.
3. Create exception handlers for the *EmptyFileError*, *ImproperTimeError*, and *FileNotFoundError* exceptions. Each exception must have its own exception handler that prints an error message specific to the type of exception and exits the program.
4. Use a single list comprehension over *time_list* that includes each time as a string in the format "4:09 PM". The minutes field must be exactly two digits and there is a single space before AM or PM. Print this list.
5. Using a single call to *max*, find the time that occurs latest in the day. Print the result.
   - To accomplish this, you will need to set the parameters appropriately–see the documentation for `max`.
   - You can write other code and functions to assist but you must use *max* (only once) to determine the latest time.
6. Using a single call to *sorted*, print the times in ascending order (earliest in the day to latest in the day).

- To accomplish this, you will need to set the parameters appropriately–see the documentation for `sorted`.
- You can write other code and functions to assist but you must use *sorted* (only once) to sort the list.

7. Create a variable *target* that is the first entry in *time_list*.
8. Use a single list comprehension that creates a list with the yielded values by calling *time_compare_gen* with *time_list* and *target*. Print this list.

Notes

- You may NOT use the *datetime* (or similar) library for this assignment.
- You may create other functions beyond what is listed here.

Sample Run

Assuming there is a sample time file as shown:

```
$ cat sample1.txt
4 12 PM
8 23 PM
4 03 AM
1 34 AM
12 48 PM
4 13 AM
11 09 AM
3 12 PM
4 10 PM
$ python3 hw2.py sample1.txt
['4:12 PM', '8:23 PM', '4:03 AM', '1:34 AM', '12:48 PM', '4:13 AM', '11:09 AM', '3:12 PM', '4:10 P
M']
(8, 23, 'PM')
[(1, 34, 'AM'), (4, 3, 'AM'), (4, 13, 'AM'), (11, 9, 'AM'), (12, 48, 'PM'), (3, 12, 'PM'), (4, 10,
 'PM'), (4, 12, 'PM'), (8, 23, 'PM')]
[(0, 0), (4, 11), (11, 51), (9, 22), (20, 36), (12, 1), (18, 57), (23, 0), (23, 58)]
```

IMPORTANT! Do not assume that testing is complete if your program produces the correct output for this provided input file. During grading, your assignment will be tested with other input files.

Grading Notes

- A test that crashes due to a run-time exception will be considered a failing test.
- Programs that contain syntax errors will receive a zero.

Version

- Last updated 18-Jan-2022; refined rubric

HW2 Rubric

Criteria	Ratings					Pts
createTimeList	**9 pts Full Marks**	**7 pts Passes most tests**	**4 pts Passes some tests**	**2 pts Attempted**	**0 pts No Marks**	9 pts
Exception handling	**12 pts Full Marks**	**12 to >0.0 pts Some tests fail and/or improper implementation**			**0 pts No credit**	12 pts
timeCompareGen	**9 pts Full Marks**	**7 pts Passes most tests**	**4 pts Passes some tests**	**2 pts Attempted**	**0 pts No Marks**	9 pts
List comprehension - list of time strings (Step 4)	**5 pts Full Marks**	**4 pts Passes most tests**	**2 pts Passes some tests**	**1 pts Attempted**	**0 pts No Marks**	5 pts
Latest time using single call to max (Step 5)	**5 pts Full Marks**	**4 pts Passes most tests**	**2 pts Passes some tests**	**1 pts Attempted**	**0 pts No Marks**	5 pts
Sorting times in ascending order using single call to sorted (Step 6)	**5 pts Full Marks**	**4 pts Passes most tests**	**2 pts Passes some tests**	**1 pts Attempted**	**0 pts No Marks**	5 pts
List comprehension - list of time difference tuples (Step 8)	**5 pts Full Marks**	**4 pts Passes most tests**	**2 pts Passes some tests**	**1 pts Attempted**	**0 pts No Marks**	5 pts

Criteria	Ratings		Pts
Additional Deductions	**0 pts** **Additional Deductions** Deductions will be represented using negative points.	**0 pts** **No Additional Deductions**	0 pts
			Total Points: 50