CPSC 3400 Languages and Computation
Winter 2024

Homework 2
Due: Wednesday, January 24 at 10:00pm

In this assignment, you will 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"). You must create the following functions and top-level functionality:

`createTimeList`

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). The function is also capable of throwing exceptions (see description).

Description: Reads a text 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, can have 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:

- `EmptyFile`: The file is empty.
- `ImproperTime`: This exception is raised if 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 thrown 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.

`timeCompareGen`

<u>Parameters:</u> List of time tuples returned from `createTimeList`, and a single "target" time tuple

<u>Description:</u> This function is a *<u>generator function</u>* that will *<u>yield</u>* for each time tuple in the list of time tuples, a tuple that indicates how far in the future it is from target. The yielded tuples will contain two integers in this order: (hours, minutes)

Example usage – 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

In addition to these functions, you must do the following steps to create a simple test driver. This code should reside outside the functions:

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 `createTimeList` with the input from the command line and storing its result into `timeList`.
3. Create exception handlers for the `EmptyFile`, `ImproperTime`, and `FileNotFoundError` exceptions. Each exception must have their 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 `timeList` that displays 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 `timeList`.
8. Use a single list comprehension that creates a list with the yielded values by calling `timeCompareGen` with `timeList` 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 Input File and Output

A sample input file is provided on `cs1` at:

```
/home/fac/bdiazcosta/cpsc3400/hw2/sample.txt
```

Here are the contents of that file:

```
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
```

If your program is working correctly, it should have the following output:

```
['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']
(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

Grading will be based on the following rubric:

`createTimeList` (correct functionality, no exceptions)	*9 points*
• All tests pass	(no penalty)
• One test fails	-4
• Two tests fail	-6
• Three or more tests fail	-9
Additional restrictions:	
• Does not return a list of tuples	-6
• Tuples do not contain the proper types	-4

Exception handling (`createTimeList` and Step 3) — *12 points*

• Nine exceptional test cases	-1 for each failing test
• Does not print error messages specific to that type of exception	-2
• Does not exit the program after an exception	-1
Additional restrictions:	
• Does not implement exception handling as directed	-4
• Does not use exception handling mechanism to catch errors	-10

`timeCompareGen` (correct functionality) — *9 points*

• All tests pass	(no penalty)
• One test fails	-4
• Two tests fail	-6
• Three or more tests fail	-9
Additional restrictions:	
• Not implemented as a generator function	-6
• Does not yield tuples	-5
• Yielded tuples do not contain the proper types	-3

List comprehension – list of time strings (Step 4) — *5 points*

• All tests pass	(no penalty)
• One test fails	-3
• Two or more tests fail	-5
Additional restrictions:	
• Minor formatting issues	-1 (per issue)
• Not putting a leading zero for minutes	-2
• Not implemented using a single list comprehension	-4
• Times are not converted to strings	-5

Latest time using single call to `max` (Step 5) — *5 points*

• All tests pass	(no penalty)
• One test fails	-3
• Two or more tests fail	-5
Additional restrictions:	
• Not implemented using a single call to `max`	-4

Sorting times in ascending order using single call to `sorted` (Step 6) — *5 points*

• All tests pass	(no penalty)
• One test fails	-3
• Two or more tests fail	-5
Additional restrictions:	
• Not implemented using a single call to `sorted`	-4

List comprehension – list of time difference tuples (Step 8)	*5 points*
• All tests pass	(no penalty)
• One test fails	-3
• Two or more tests fail	-5
Additional restrictions:	
• Not implemented using a single list comprehension	-4

In addition, programs may lose additional points as follows:

- Up to 5 points may be deducted if the program is very unreadable.
- Up to 40 points may be deducted if your program uses the `datetime` (or similar) library for this assignment.
- 6 point deduction if the file name is not obtained from the command line for part 1.
- Programs that contain syntax errors will receive a zero.

Submitting your Assignment

On `cs1`, run the following script in the directory with your program:

```
/home/fac/bdiazacosta/submit/cpsc3400/hw2_submit
```

This will copy the files `hw2.py` to a directory that can be accessed by the instructor. Please be sure to keep the same file names or the submission program will not work. Only the last assignment submitted before the due date and time will be graded. ***Late submissions are not accepted and result in a zero.***