CSE 231 **Fall 2015**

Computer Project #11

Assignment Overview

This assignment focuses on the design, implementation and testing of a Python class to manipulate times, as described below.

It is worth 50 points (5% of course grade) and must be completed no later than 11:59 PM on Monday, December 7.

Assignment Deliverables

The deliverables for this assignment are the following files:

> **`times.py`** – the source code for your implementation of **`class Time`**
> **`test.py`** – the source code for your test program

Be sure to use the specified file names and to submit them for grading via the **handin system** before the project deadline.

Assignment Background

A common representation of the time of day, referred to as UTC (Coordinated Universal Time) uses a 24-hour clock and the notation hh:mm:ss+zz or hh:mm:ss-zz, where:

> hh is a two-digit hour between 00 and 23
> mm is a two-digit minute between 00 and 59
> ss is a two-digit second between 00 and 59
> zz is a two-digit offset from Coordinated Universal Time between -12 and +12

For example, 08:30:00-03 represents 8 hours, 30 minutes and 0 seconds in the time zone which is 3 hours behind Coordinated Universal Time. Additional information can be found at:

> https://en.wikipedia.org/wiki/Coordinated_Universal_Time

Assignment Specifications

For this assignment, you will design, implement and test the **`times`** module, which contains **`class Time`**. To test the **`times`** module, you will develop a program which *thoroughly* tests each method in **`class Time`**.

1. Your implementation of **`class Time`** must be in the file named **`times.py`** (and it must be the only thing in that file).

2. Your implementation of the test program must be in the file named **`test.py`** (and it must be the only thing in that file).

The test program must import your **`times`** module and must demonstrate that each method in **`class Time`** is implemented correctly. The output produced by your test program must be clearly labeled and readable.

In addition, the test program must contain at least one function (besides function **`main`**), and that function must have at least one parameter which is a **`Time`** object and which it uses in producing some output. The test program must call your function.

Specifications for Class Time

1. Method **`__init__`** initializes a **`Time`** object. The following examples illustrate the method's behavior, where the desired time is shown as a comment:

```
A = times.Time( 6, 15, 30, 5 )   # 06:15:30+05
B = times.Time( 8, 9, 15, -4 )   # 08:09:15-04
C = times.Time( 14, 20, 45 )     # 14:20:45+00
D = times.Time( 23, 59 )         # 23:59:00+00
E = times.Time( 12 )             # 12:00:00+00
F = times.Time()                 # 00:00:00+00
```

The method will validate all parameters. If the method is unable to construct a valid **`Time`** object, it will raise a **`ValueError`** exception.

2. Methods **`__str__`** and **`__repr__`** return a **`str`** object which is a printable representation of a **`Time`** object. Assuming the Python statements above, the following examples illustrate the behavior of the methods, where the desired **`str`** object is shown as a comment:

```
str(A)     # "06:15:30+05"
repr(B)    # "08:09:15-04"
str(C)     # "14:20:45+00"
repr(D)    # "23:59:00+00"
str(E)     # "12:00:00+00"
repr(F)    # "00:00:00+00"
```

3. Method **`from_str`** accepts a **`str`** object (representing the desired time) and changes the value of the **`Time`** object. For example:

```
T = times.Time()
T.from_str( "06:15:30+05" )      # 06:15:30+05
```

The method will validate the parameter. If the parameter does not represent a valid **`Time`** object, it will raise a **`ValueError`** exception.

4. Method **`get_as_local`** returns a **`str`** object which is a printable representation of the time based on a 12-hour “local” clock (which leaves out the time zone). Assuming the Python statements above, the following examples illustrate the behavior of the method, where the desired **`str`** object is shown as a comment:

```
A.get_as_local()      # "06:15:30 AM"
B.get_as_local()      # "08:09:15 AM"
C.get_as_local()      # "02:20:45 PM"
D.get_as_local()      # "11:59:00 PM"
E.get_as_local()      # "Noon"
F.get_as_local()      # "Midnight"
```

5. The class will support six forms of comparison between two **`Time`** objects. For example:

```
T1 = times.Time( 7, 35, 15, -6 )
T2 = times.Time( 7, 21, 30, -5 )
T1 == T2   # False
T1 != T2   # True
T1 < T2    # False
T1 <= T2   # False
T1 > T2    # True
T1 >= T2   # True
```

The methods will raise a **`TypeError`** exception if the second parameter is not a **`Time`** object.

6. The class will support the addition of a **`Time`** object and an **`int`** object (which represents a number of seconds). For example:

```
T1 = times.Time( 23, 15, 0, 5 )  # 23:15:00+05
T2 = T1 + 300                    # 23:20:00+05
T3 = T1 + 3600                   # 00:15:00+05
T4 = T1 + -90000                 # 22:15:00+05
```

The method will raise a **`TypeError`** exception if the second parameter is not an **`int`** object.

7. The class will support the subtraction of two **`Time`** objects, where the result is the number of seconds by which the two times differ. For example:

```
T1 = times.Time( 14, 20, 45 )    # 14:20:45+00
T2 = times.Time( 14, 18, 15 )    # 14:18:15+00
T1 – T2                          # 150
T2 – T1                          # -150

T1 = times.Time( 7, 35, 15, -6 ) # 07:35:15-06
T2 = times.Time( 7, 21, 30, -5 ) # 07:21:30-05
T1 – T2                          # 4425
T2 – T1                          # -4425
```

The method will raise a **`TypeError`** exception if the second parameter is not a **`Time`** object.

Assignment Notes

1. Items 1-10 of the Coding Standard will be enforced for this project.

2. It is critical that your methods use the specified names, number and type of parameters.

3. The quality of your test program is important. It must be thorough in showing that each method operates correctly for a set of parameter values which cause the method to execute different paths (sequences of statements). Output must be clearly labeled indicating the expression that was evaluated and the outcome.

4. The keyword **`raise`** is used to raise an exception. For example:

```
raise TypeError
```

See subsection 14.6.1 of the Punch and Enbody textbook.

5. For this assignment, midnight is defined to be 00:00:00 (rather than 24:00:00).