CSE 231 **Fall 2015**

Computer Project #10

Assignment Overview

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

It is worth 15 points (1.5% of course grade) and must be completed no later than 11:59 PM on Monday, November 30.

Assignment Deliverable

The deliverable for this assignment is the following file:

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

Be sure to use the specified file name and to submit it 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

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.

For this assignment, you will design and partially implement the **`times`** module, which contains **`class Time`**.

Assignment Specifications

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

2. You will supply stubs for the methods listed below. Recall that a stub is a function definition which is syntactically correct, but which is incomplete (it does no useful work).

Each stub will include a succinct docstring which states the purpose of the method.

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 body of the stub for this method will be a **pass** statement.

2. Methods **__str__** and **__repr__** return a **str** object which is a printable representation of a **Time** object.

The body of the stub for each of these methods will be a **return "time string"** statement.

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

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

The body of the stub for this method will be a **pass** statement.

4. Method **from_seconds** accepts an **int** object (the number of seconds from the start of a day) and changes the value of a **Time** object. For example:

```
T = times.Time()
T.from_seconds( 2300 )
```

The body of the stub for this method will be a **pass** statement.

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

```
T1 = times.Time( 6, 15, 30, 5 )
T2 = times.Time( 8, 9, 15, -4 )

T1 == T2
T1 != T2
T1 < T2
T1 <= T2
T1 > T2
T1 >= T2
```

The body of the stub for each of these methods will be a **return False** statement.

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( 6, 15, 30, 5 )  # 06:15:30+05
T2 = T1 + 300                    # 06:20:30+05
```

The body of the stub for this method will be a **`return self`** statement.

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
```

The body of the stub for this method will be a **`return 0`** statement.

Assignment Notes

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

2. It is critical that your stubs use the specified names and specified number of parameters.

3. Each stub should be a syntactically correct method (although it does not need to accomplish the work associated with the method). A method which returns a value must contain a **`return`** statement which returns a value of the correct type; a method which does not return a value must contain a **`pass`** statement.

4. The file named **`check.py`** in the **`Project10`** directory contains a simple program which calls the methods in class Time and can be used to identify some problems with your stubs, such as misspelled method names and incorrect numbers of arguments.