Exercises

Level 5: Advanced Python Syntax

5.1: Date/Time

1) Create a program that does the following:

- **a.** Asks the user to input year, month, day, hour, minute, second, microsecond (one after another).
- **b.** Create a **datetime** variable with the entered info.
- **c.** Extract the **datetime** into year, month, day, hour, minutes, second, and microsecond. Display the following result (where … is the extracted value):

  Year: …
  Month: …
  Day: …
  Hour: …
  Minute: …
  Second: …
  Microsecond: …
- **d.** Display the entered **datetime** with the following format: **2016-09-25 18:23:14:12342**
- **e.** Display the entered **datetime** with the following format: **2016 September 25 06:24:14:12342 PM**
- **f.** Do parts d-e with the current local time.
- **g.** Do parts d-e with the current UTC time.

2) Modify the above program to request the user enter the date in the following format (for example):

2016-09-25 18:23:14:12342

3) Modify the above program to request the user enter the date in the following format (for example):

2016 September 25 06:24:14:12342 PM

4) Create a 'date calculator' program. It should do the following:

- **a.** Prompt the user to enter any date and time.
- **b.** Prompt the user to enter a *delta* time that is used to add or subtract from the original. For example, if the user enters **-**00:25:13:0 then subtract 25 minutes and 13 seconds (and zero microseconds). Another example: 72:12:00:154 means to add 72 hours, 12 minutes, and 154 microseconds.
- **c.** Display the calculated resulting date and time, in an easily-readable format.

5) Create a 'date differential' program. It should do the following:

- **a.** Prompt the user to enter any date and time.
- **b.** Prompt the user to enter another date and time.
- **c.** Subtract the two **datetime**s and display the result to the user. It should be displayed in several separate ways:
  - **i.** Total number of days (including fractions of days).
  - **ii.** Total number of hours (including fractions).
  - **iii.** Total number of minutes (including fractions).
  - **iv.** Total number of seconds (including fractions).
  - **v.** Total number of microseconds (including fractions).
  - **vi.** A complete (grammatically-correct) sentence that breaks everything down to the correct units and excluding any 0s. For example:

    The difference is 5 days, 3 hours, 16 minutes, 10 seconds, and 150 microseconds.

6) Modify your **Loan** classes to take a loan start date and loan end (maturity) date instead of a *term* parameter. Create a **term** method that calculates and returns the loan term (in months) from the two dates.

5.2: Decorators

Decorators

1) Modify the Timer class to work as a decorator (feel free to use the provided sample code). Its usage should look like this:

```
@Timer
def myFunc(arg1, arg2):
    print "Do Work Here"
```

An example output would look like: **<function myFunc at 0x34173DF0>: 1.5467 seconds**

How does this compare to the previous approach to using the context manager? When is this more useful and when are context managers more useful?

Memoization

2) Create a decorator that memoize's the result of a function. This decorator should be flexible enough that it can work with a function with any number of parameters. Note that memoizing should happen on a per-parameter basis; meaning, cache the result for every unique set of parameter values. **Hint**: Use a *dict*.

Be sure to test this decorator on different functions to ensure it works properly. You should also use in conjunction with the **timer** decorator, to demonstrate that subsequent calls to the memoized function are quicker than the initial call for a given parameter set.

3) Use your memoization decorator from the previous exercise to memoize the recursive versions of the Loan waterfall functions. Time the functions before and after; do you see a difference?

In practice, there are quite a few considerations when implementing a memoization decorator. For example, there are memory considerations (what happens to the cache if/when an object or function gets destroyed?). However, all of this is beyond the scope of this course. Note that many firms will have internal libraries that already have a memoization decorator or class implemented, for you to use out-of-the-box.