PYTHON PROBLEM 2

RULES

- Use Python 3 for this exercise (it is not possible within Python 2).
- Certain ways of writing operations are equivalent, and hence count as only one implementation method. For example, list comprehensions are equivalent to for loops, and temporary value creation is the same as chaining.

```
## The following two are equivalent implementations
a1 = [x**2 for x in range(5)] ## This is the python 3 range
a2 = []
for x in range(5):
    a2.append(x**2)

## The following two are equivalent implementations
temp = 'Hello World!'.lower()
b1 = temp.split()
b2 = 'Hello World!'.lower().split()
```

- Globals are not allowed.
- Reviewing ideas and definitions on the internet is encouraged, however, please attempt in all earnestness to develop the source code on your own. Using solutions or even similar ideas on the internet can rob you of the learning process
- Write up the solutions as if it were a real homework assignment.

PROBLEMS

There once was a girl named Clare, who was the most talented engineer of them all. She specialised in Data Acquisition Systems (DAQ) cards, which she could use to slay even the most difficult digital acquisition dragons.

Ryan is a bit tired

One day, she decided she wanted to create a battle droid for the FIRST robotics competition, complete with treads and a cool flame decal (for +2 damage points). She started out by writing her normal DAQ routine for creating a waveform, which went along the following lines.

```
def waveform(frequency, phase):
    ''' Return discrete samples of a waveform, at an offset of phase.  '''
    # Code code code
```

```
4
5       ## return a new phase, which points to the place we stopped taking
6       ## waveform samples at.
7       return waveform_points, phase
8
9   phase = 0
10  while True:
11      waveform_points, phase = waveform(frequency, phase)
12      sendToDaq(waveform_points)
```

However, she was tired of this method, as it required her to carry around a variable all through her code. Having just learned Python, she read about an incredible capability she didn't have access to in C: first class functions! She pondered to herself:

1. What is a first class function? Write down three unique examples of functions being used as first class objects.
2. A similar concept named higher order functions came up in her search. What do those do? How do they relate to first class functions?
3. She had met Bill from problem set 1. Was `talkToBill` a first class function or a higher order function?
4. Two similar concepts are nested functions and closures. Describe these two programming concepts, and what is different between them.
5. (Bonus) What is the performance cost (qualitatively) for using first class functions in Python? When should a developer worry about these problems in practice?

``Hmm, this seems pretty neat!'' she thought. She decided to try a few crazy things.

6. Create a function named `gThenf`, which takes in two functions f, g as input (where f and g both only take in one argument) and returns a new function. This new function takes in one input x and returns the result of first applying g to x, and then applying f to the result from g.
7. This function is widely known by another name. What is it?
8. Why are the arguments f, g in that order?
9. Write a new function(s) that does the same as `gThenf`, but performs this on any number of inputs.

These examples were cute, but she was hungry to use the computer for something that she couldn't do easily on paper.

10. Make a new function `walkAndApply`, that takes in a function f and applies it to every item in an iterable x. This should work for any type that is iterable. For example, x could be a list, a tuple, a set, a dictionary, a priority heap, a tree, etc. The returned value should be a list.
11. Use this function to apply a variety of functions (of your choosing) to a single value.

12. What is the common name of this function?

Given what she has learned, Clare decides to turn her attention back to *OskiBot 5000*, her lovable bear robot of destruction. Part of the *first* robotics competition has the judges try to break the robot by sending it arbitrary waveforms that are outside of the operational voltage range of the motors.

13. The waveforms given by the judges come as functions. Devise a way to monitor these waveform functions to print ``*OskiBot 5000* is shutting down to protect itself'' when the waveform exceeds a certain value defined at runtime (values cannot be hard coded). The result should have the same calling structure as the original waveforms given by the judges.

14. Devise a waveform function that returns a random value any time it is called. The random value must be allowed to exceed some runtime defined error threshold, as above.

15. Using decorator notation apply your protection function to this new waveform function in the prior question.

With her robot now safe from harm, she looks back at her original implementation of `waveform` and figures out a way to simplify her system so she does not have to pass around phase all the time. In this new system, each analog output channel on her DAQ card analog is given a function `f(number_samples)`, where `number_samples` is the number of samples that the DAQ wants to take from the waveform function `f`.

16. Devise a waveform generator function that fits in Clare's new DAQ architecture, where the phase does not have to be passed around explicitly. Additionally, this method should be able to make waveforms of different parameters. For example, a sine waveform generator should be able to keep track of the phase and work for any frequency. Additionally the same call should be able to attach to an arbitrary number of channels, each of which may require a different parameter set (for example, 4 channels running sine waves, each with some random initial phase offset and frequency). Choose two types of waveforms to generate (sine wave, ramp, triangle wave, etc).

17. Decorate these functions will two different types of protection checks. One should be for when the waveform exceeds some bounded region (for example, $\pm 3.3\,\mathrm{V}$), while the other should check for when the rate of change between waveform samples exceeds some value (the slew rate is higher than the system can handle).

With these modifications complete, *OskiBot 5000* is ready to obliterate *ArborTreeSissyBot*.

For those in the Conolly lab, answer the following extra questions.

18. How do the fourth set of questions (those about protection) relate to the real time system?

19. How to the fifth set of questions (those about analog output) relate to the real time system?