# PYTHON PROBLEM 1

## RULES

This exercise can be performed completely within `ipython` with out the pylab extension. *At no point can you store what Bill says! You cannot edit Bill either; that would be cruel.* All interactions must be through talking with him. The following are invalid.

```
a = talkToBill()
talkToBill().some_function() # Bill's output is temporarily stored in a chain.
talkToBill().some_property   # See above.
```

All of the code problems (except the RPN calculator) can be solved in one legible line of code (ignoring import statements). This means `def` is not required, but also not an invalid solution.

Hints: `http://goo.gl/7JRUpU`, `http://goo.gl/frBLtZ`, `http://goo.gl/QGjZ7k`.

## PROBLEM

Bill is a travelling salesman with a briefcase filled with goodies that he is attempting to sell to you. However, Bill is a strange character. He only seems to accept programming statements, and only one line expressions at that. What can I say, CS people are kinda *weird*.

Your only interface to Bill is the `talkToBill` function, and you forgot a notepad so you can't write down what he says (see above).

When you send him no input (google for these answers),

1. What type of data structure does he return? Write a new function that returns the same data (create a Bill simulator).
2. What is the access cost for this type of data structure? Why?
3. What is the insert cost for this type of data structure? Why?
4. What are the properties that index must hold when inserting an element into this data structure? Why?
5. What other scenarios would this data structure be useful for?

You begin to wonder some profound mathematics questions to yourself. You decide to ask Bill, keeping in mind that he only accepts one statement inputs (an example is given in the help of `talkToBill`).

6. "What is Bill's default input?"

7. "What items can I get from Bill?" (in sorted order)
8. "If I chose one of each item from Bill, how much would that cost?"
9. "Can I get Bill to return a list of tuples, where the tuple contains the item and the cost?"

Bill is so predictable. In fact, this makes him very vulnerable. He can do all sorts of things, if you just ask in the right way.

10. How can you get him to write out your name?
11. Make him tell you "NAME is pretty" for any input name.
12. How can you get him to print out the date?

You want to ask him more complicated questions but sadly forgot your pad at the office. But, being quite the clever programmer, you decide there is an alternative way to get him to remember things for you. This opens a world of questions that couldn't be asked before (or not easily).

13. For all the items Bill blabs on about, reverse the name of the item. Additionally, remove all items that are cheaper than $2. The resulting data type should be the same as when Bill is called with no input.
14. Make Bill output both the price and the current date he sold you the item.
15. Add five hilarious items to his list.
16. For each item, choose a random number between 0 and 10 of that item. How much would that cost? (Additionally, how could you do this in one statement like in the second section?)

Finally, out of boredom, you decide to turn Bill into a calculator! He supports the following operations: `+`, `-`, `*`, `/`. You set him up to work like a Reverse Polish Notation (RPN) calculator, where he accept a list of numbers and operators and returns a result.

## APPENDIX

Below is the function declaration/help for the `talkToBill` function.

```
def talkToBill(f):
    ''' Bill, the weird salesman

    Bill only responds to single statement inputs.

    Parameters
    ----------
    f : statement to ask.
        What do you want to ask Bill?

    Returns
    -------
    Depends!
    '''
```