

CSE 391

Homework #1

Python

September 19, 2007

1. Explain why Python is both *strongly* and *dynamically* typed.
2. Suppose Bob has a two-dimensional 9x9 array (a list of lists) representing a Sudoku board (www.sudoku.com). It contains a number if a position has been filled in already and `None` otherwise.
Suppose for a given square (x, y) Bob has a list of possible values `possibilities`, and he wants to know which is the right one. Suppose further he has a function `isSolvable` which takes a board representation as input and returns whether or not it is possible to solve it, but which may alter the input board as it decides this.

Bob codes the following function to do that job:

```
def tryOutPossibilities(board, x, y, possibilities):
    for z in possibilities:
        board[x][y]=z
        if isSolvable(board):
            print z + " is the correct choice!"
```

There's a serious problem with Bob's solution as coded. What is it?

3. Given a list `a`, write expressions in slice notation for the following portions of the list:
 - (a) the third through sixth, inclusive, elements
 - (b) all but the last element
 - (c) the whole list
4. Suppose Fred has a program which frequently needs to get the results of some function `f` which takes as input a tuple of immutable objects and returns an output computed entirely from the input tuple. Unfortunately, this function is extremely expensive to compute, so he would like to cache his results so that if he wants the value of `f` for a tuple for this which has been computed before, he can get it nearly instantly.

Write a class, `CacheListFunction`, to help him. Your class should offer the following two methods:

- a constructor which takes one argument, the function `f` to cache. `f` can itself be assumed to take a single argument, a tuple of immutable items.
- `invoke`, which should take an immutable tuple as an argument. This function should see if we've calculated `f` on that tuple before; if so, it should return our memorized value. Otherwise, it should compute the value, memorize it, and return it. `f` should never be called twice with the same input.

5. Sally wants a dictionary which maps lists of strings to the first other string that list makes her think of. She tries this:

```
thoughtOfString={}
sounds=['bark!','meow!','neigh!']
thoughtOfString[sounds]='quack!'
```

but it results in the error:

```
TypeError: list objects are unhashable
```

What is the problem? How would you fix it?

6. Sam Sloppycoder wrote the following function which given a two-dimensional array `m`, a function `g`, and the array's dimensions returns a copy of `m` where each element has been transformed by the function `g`:

```
def TDapply(g, m, d1, d2):
    a=[]
    for i in range(0, d1):
        r=[]
        for j in range(0, d2):
            r.append(g(m[i][j]))
        a.append(r)
    return a
```

Sam's function can be rewritten as two-argument function `TDapply(g,m)` consisting of a single statement using list comprehensions (which also works more generally on lists of lists of heterogeneous lengths). Do this.

7. Alice has a function `getNextList` which will give her the next in an extremely long sequence of small lists of numbers (less than 20 numbers in each list). She wants to concatenate the first ten million of these lists.

Alice can think of two ways to do this.

```
def approachA():
    l=[]
    for x in range(1000000):
        l.extend(getNextList())
```

```
def approachB():
    l=[]
    for x in range(1000000):
        l=l+getNextList()
```

Which will be faster and why? (**hint:** if it's not clear which is faster, try doing this with a dummy `getNextString` function that always returns `[1,2,3,4]`, reducing the number of lists concatenated to 10,000.)

8. A secret organization has sent you a message using a very simple code: within a text, every letter which follows a V, F, or D (upper or lower case, ignoring word boundaries) should be extracted and concatenated, and everything else should be discarded.

Write a program which takes one argument, the filename of the ciphertext to decode, and writes the decoded message to a file with the same name as the ciphertext except with a `.decoded` suffix.

You can find a practice ciphertext at <http://www.cis.upenn.edu/~cse391/vfd.txt> . Please include both your code and its output on that example.