Ling 5801: Lecture Notes 19
Functional Programming

19.1 Use functions as arguments to other functions:

In 'quants.py', define some useful boolean functions taking list and comparator function...

```
# 'all_of' quantifier
def all_of(L,f):
    for l in L:
        if not f(l): return False
    return True

# 'none_of' quantifier
def none_of(L,f):
    for l in L:
        if f(l): return False
    return True

# 'some_of' quantifier
def some_of(L,f):
    for l in L:
        if f(l): return True
    return False
```

now, define a comparator function:

```
def is_A(c):
    return c=='A'
```

now see if all the elements of a list satisfy that comparator:

```
quants.all_of(['A','A','B'],is_A)
```

should return:

```
False
```

19.2 Define anonymous (un-named) functions:

'Lambda' function:

1. ⟨α-to-β-expr⟩ → `lambda` ⟨α-var⟩ `:` ⟨β-expr⟩

   define function of type ⟨α-to-β-expr⟩ for any input ⟨α-var⟩ returning ⟨β-expr⟩

Use 'lambda' notation in-line, to save defining comparator function every time:

```
quants.all_of ( ['A','A','B'], lambda c: c=='A' )
```