CS101-Ames Final Exam **KEY**, 70 points possible
12/18/2013

Part I. Function writing. [5 points each]

1.
```
def tens(n):
    if n==0:
        return 1
    return 10*tens(n-1)
```

2.
```
def reverse3Digits(n):
    ones     = n % 10
    tens     = (n / 10) % 10
    hundreds = n/100
    return ones*100 + tens*10 + hundreds
```

3.
```
def reverseDigits(n, reversed=0):
    if n==0:
        return reversed
    return reverseDigits(n/10, reversed*10+n%10)
```

4.
```
sum=0
for n in range(1,101):
    # next line: NOT int div!
    sum += 1./2**n
print sum
```

3.
```
# a possible alternative solution to problem 3,
# but only if you're comfortable with logarithms.
def reverseDigits(n):
    if n<10:
        return n
    digits = int(math.log(n,10))+1  # number of digits in n
    msd = n/10**(digits-1)
    return reverseDigits(n-msd*10**(digits-1))*10 + msd
```

Part II. Reading.

`11 5 2 1 0`
`1 2 4 8 16 32`
`dlrow olleH`

Part III. Writing. [10 points each]

1.
```
def encrypt(message, n):
    result = ""
    for i in range(len(message)):
        c = message[i];
        cn = ord(c)+n
        if cn>126:
            cn -= 95 # 95 is 127-32
        result += chr(cn)
    return result
```

```
# Better
def encrypt(message, n):
    result = ""
    for c in message:
        cn = ord(c)+n
        if cn>126:
            cn -= 95 # 95 is 127-32
        result += chr(cn)
    return result
```

2.
```
def removeSilence():
    left  = BCAudio.getLeft()
    right = BCAudio.getRight()
    newLeft = []
    newRight = []
    for i in range(len(left)):
        if left[i]==0 and right[i]==0:
            pass
        else:
            newLeft.append(left[i])
            newRight.append(right[i])
    left[:] = newLeft
    right[:] = newRight
```

Part IV. Writing a class. [15 points]

```
class FaceMetrics:
    def __init__(self, leftEye, rightEye, mouthLeft, mouthRight, mouthCenter):
        eyeDist    = self.distance(leftEye, rightEye)
        mouthWidth = self.distance(mouthLeft, mouthRight)
        self.eyeToMouthRatio = float(eyeDist)/mouthWidth
        self.mouthAsymetry   = self.distance(mouthCenter, mouthLeft) / \
                               self.distance(mouthCenter, mouthRight)

    def distance(self, p1, p2):
        return math.sqrt((p1.x-p2.x)**2 + (p1.y-p2.y)**2)

    def compareTo(self,face2):
        difference = abs(self.mouthAsymetry   - face2.mouthAsymetry  )/2. + \
                     abs(self.eyeToMouthRatio - face2.eyeToMouthRatio)/2.
        return difference
```