There are far more ways to solve these problems, than the few that I give below…

1. (5 pts) Write the Python expression (lie of code) to do the following:

$$a = \sqrt{12 + \frac{fred}{harry^3}}$$

```
a=math.sqrt(12+fred/harry**3)
```

You may also have worried about preventing integer division, and not having to import math…:

```
a=(12+float(fred)/harry**3)**0.5
```

2. (10 pts) Create the function *sumSquares(n)* which will be given a positive integer and will return the sum of all the squares of numbers from 1 until (and including) n. Example:

```
sumSquares(3) # returns 14 (= 1+4+9)

def sumSquares(n):
      total=0
      for i in range(1,n+1):
            total+=i*i
      return total

or, with list comprehensions:

def sumSquares(n):
      return sum([i*i for i in range(1,n+1)])
```

3. (10 pts) The values of *sumSquares(1), sumSquares(2), … sumSquares(n)* forms an interesting sequence. It looks like: 1, 5, 14, 30, 55, … If there are n numbers in this sequence (from *sumSquares(1)* … to … *sumSquares(n)*) , how many of those numbers are evenly divisible by 3 or evenly divisible by 5, but not by both. Create the function *sq35(n)* to return that answer. Assume that your answer for problem #2 above is correct.

```
def sq35(n):
      cnt=0
      for i in range(1,n+1):
            sm = sumSquares(i)
            if (sm%3==0 and sm%5!=0) or (sm%3!=0 and sm%5==0):
                  cnt+=1
      return cnt
```

or

```
def sq35(n):
      cnt=0
      for i in range(1,n+1):
            if sumSquares(i)%3==0 or sumSquares(i)%5==0:
                  cnt+=1
            if sumSquares(i)%15==0:
                  cnt-=1
      return cnt
```

4. (5 pts) Create the function *countPuncs(Elaine)* that will be given a string (which may be empty), and returns the number of punctuation characters in it. The punctuation characters we want to count are: period, comma, colon, semi-colon, exclamation mark and question mark. Example:

```
countPuncs('Stop! In the name of love, Before you break my heart.') # returns 3
```

```
def countPuncs(Elaine):
      puncs='.,:;!?'
      cnt=0
      for c in Elaine:
            if c in puncs:
                  cnt+=1
      return cnt
```

or

```
def countPuncs(Elaine):
      puncs='.,:;!?'
      cnt=0
      for p in puncs:
            cnt+=Elaine.count(p)
      return cnt
```

or

```
def countPuncs(Elaine):
      puncs='.,:;!?'
      return len([c for c in Elaine if c in puncs])
```

5. (10 pts) Create the function *replaceAll(astring,lookfor,replaceWith)* which duplicates the work of the built-in string method: *.replace()*. Given a string (in *astring*), it will find all (exact match) occurrences of the string in the variable *lookfor* inside *astring* and replace each of them with the string in *replaceWith*. All 3 parameters, *astring, lookfor* and *replaceWith* will be non-empty strings. Of course, you may not use the built-in string *.replace()* method. Example:

```
replaceAll('Now he said: no way, nohow, NO, no', 'no', 'yes')  # returns 'Now he said, yes way,
yeshow, NO, yes'
1. replaceAll('whatever, what', 'what', 'where')  # returns 'whereever, where'
2. replaceAll('he said, she said','he','she')  # returns: 'she said, sshe said'
3. replaceAll('he said whatever else', 'whatever', ' ') # returns 'he said  else' (replaceWith is
a single space)
```

This one is tricky because it's not really possible to solve it correctly using the looping construct "for i in range(len(astring))" or "for c in astring". That's because the list produced by range() or the sequence of characters produced by astring cannot be skipped when necessary. So, a different approach needs to be taken, using "while" or ".find()"

…and remember, it's possible that lookfor is not found inside astring…

```
def replaceAll(astring,lookfor,replaceWith):
    pos=astring.find(lookfor)
    answer=''
    while pos>=0:
        answer += astring[:pos]+replaceWith
        astring = astring[pos+len(lookfor):]
        pos=astring.find(lookfor)
    return answer+astring
```

Here's a fancy way based upon an idea from Ann Caplin:

```
def replaceAll(astring,lookfor,replaceWith):
    return replaceWith.join(astring.split(lookfor))
```

6. (15 pts) Create the function *primesUnder(n)* which will return a sentence informing us of all the prime numbers less than *n* (which is an integer). Note: 0 and 1 are not primes. Examples:

```
primesUnder(5) # returns 'The primes less than 5 are 2 and 3.'
primesUnder(3) # returns 'The prime less than 3 is 2.'
primesUnder(12) # returns 'The primes less than 12 are 2 and 3 and 5 and 7
and 11.'
primesUnder(2) # returns 'There are no primes less than 2.'
```

```
def isPrime(n):
    if n<2:
        return False
    if n==2 or n==3:
        return True
    for i in range(2,int(n**0.5)+1):
        if n%i==0:
            return False
    return True

def listPrimes(n):
    lst=[]
    for i in range(2,n):
        if isPrime(i):
            lst.append(i)
    return lst

def primesUnder(n):
    if n<=2:
        return 'There are no primes less than 2'
    if n==3:
        return 'The prime less than 3 is 2'
    lst=listPrimes(n)
    answer='The primes less than '+str(n)+' are '
    for prime in lst:
        answer+=str(prime)+' and '
    return answer[:-5]
```

7. Extra credit: (10 pts) Create the function countWords(s), which will be given a (possibly empty) string, and will return the number of words therein. For the sake of this function's definition, a word is a set of non-space characters surrounded by spaces, or located at the beginning or end of a string. So, one or more spaces separate words. Here are examples:

```
countWords('')  # returns 0
countWords('fred') # returns 1
countWords('    ')  # returns 0
countWords('a fred    ') # returns 2
countWords('  .!@#$%   dot.here') # returns 2
```

One way to do it is to step through the string, and figure out when you are transitioning between being outside a word and then inside…

```
def countWords(s):
      was_space=True    # start off as if the character before the first
                        #   letter in s is a space
      cnt=0
      for c in s:
            if c != ' ' and was_space:  # if we're starting a word…
                  cnt+=1
            if c == ' ':
                  was_space=True
            else:
                  was_space=False
      return cnt
```

or the fancy way, if you know the .split() method:

```
def countWords(s):
      return len(s.split())
```