Problems to review for the exam on Wed:

1. Consider the following definition:

```
def fun(n, m):
    return m - n
```

Evaluate the following expressions (perhaps using stack diagrams:

a) `fun(fun(1, 2), 3)`
b) `fun(fun(1, 2), fun(3, fun(fun(4, fun(5, 6)), 7)))`
c) `fun(fun(1, 2), fun(3, fun(fun(4, fun(5, 6)), fun(7, 8))))`

What happens if in the definition of `fun` above we replace `return` by `print`?

First off replacing return by print would be a disaster, since print does not return a value and the expressions above all build on the values calculated previously. The answers are:

```
>>>
>>> fun(fun(1, 2), 3)
2
>>> fun(fun(1, 2), fun(3, fun(fun(4, fun(5, 6)), 7)))
6
>>> fun(fun(1, 2), fun(3, fun(fun(4, fun(5, 6)), fun(7, 8))))
0
>>>
```

One does not need to use stack diagrams per se, as long as one communicates the actual situation correctly. For example `fun(fun(1, 2), 3) = 3 - fun(1, 2) = 3 - (2 - 1) = 2`. So, using parens would also work. On paper, with a pencil, diagrams would be easier to draw, in some cases at least. If print does not return a value (and it doesn't) then one can see why replacing return with it would fail:

```
>>> 1 + print 3
SyntaxError: invalid syntax
>>>
```

2. Simplify the following expressions:

- `a and not a` this is the same as `False`
- `a or not a` this is the same as `True`
- `x >= 6 and x < 7` if `x` is an integer this is the same as `x == 6`
- `a == False` same as `not a`
- `a == True` same as just `a`
- `a or True` this is `True` always
- `a and True` this is `a` always
- `a and False` this is `False`
- `a or False` this is `a` all the time

- `False and False or True` this is `True`
- `False and (False or True)` this is `False`

Is `not (a and b)` the same as `not a or not b`? Justify your answer.

It is and this is one of De Morgan's laws. You can prove it with a truth table, as follows:

a	b	a and b	not (a and b)	not a	not b	not a or not b
True	True	True	False	False	False	False
True	False	False	True	False	True	True
False	True	False	True	True	False	True
False	False	False	True	True	True	True

3. Define `chained conditionals` and write a program using a chained conditional to solve the following problem: Write a program (called `Eight`) that translates a number between `0` and `4` into the closest[1] letter grade. For example, the number `2.8` (which might have been the average of several grades) would be converted to `B-`. Break ties in favor of the better grade; for example `2.85` should be a `B`.

Chained conditionals are sequences of if-elif-elif-elif-else very much like a switch. This is a tricky program (for some reason) because people always confuse the cutoffs:

```
def grade(num):
    if num > 4:
        print "The upper limit is 4,", num, "not a valid grade."
    elif num >= 3.85:
        print "The grade is A"
    elif num >= 3.5:
        print "The grade is A-"
    elif num >= 3.15:
        print "The grade is B+"
    elif num >= 2.85:
        print "The grade is B"
    elif num >= 2.5:
        print "The grade is B-"
    elif num >= 2.15:
        print "The grade is C+"
    elif num >= 1.85:
        print "The grade is C"
    elif num >= 1.5:
        print "The grade is C-"
    elif num >= 1.15:
        print "The grade is D+"
    elif num >= 0.85:
        print "The grade is D"
    elif num >= 0.5:
        print "The grade is D-"
    elif num >= 0:
```

[1] A, B, C, D have the values 4, 3, 2, 1. A plus adds 0.3, a minus takes 0.3 away. F is 0. A+ is a 4.

```
            print "The grade is F"
        else:
            print "The lower limit is 0,", num, "not a valid grade."
```

Why 3.85 the cutoff for A? Because 3.85 is the midpoint between 3.7 and 4, so all the numbers bigger than 3.85 are closer to 4 (which is an A) than to 3.7 (which is A-).

4. Consider the following two fragments:

```
if x == 5:
  x = x + 1
else:
  x = 8
```

```
if x == 5:
  x = x + 1
if x != 5:
  x = 8
```

Are the two fragments logically equivalent? Why or why not?

Notice that the second test in the second fragment might be working on a changed x. Thus, if x is 5 x becomes 6 in that fragment. Since it's no longer 5 it becomes 8. If it had not been x it would have been turned into 8 anyway. So the second fragment turns x in to 8 no matter what. The first one though makes x 6 if it was 5 to start with, so since at least one such value exists for which the two fragments work differently the two fragments are not identical (they are not logical equivalent).

5. What is recursion? Write a recursive function that plays "guess the number" with the user. Use `raw_input` for usr input and keep playing until the user guesses the number. Can you keep track of and report the total number of guesses at the end? Can you program the game where you have a total of ten tries at most?

The saying goes that to understand recursion you first need to understand recursion. But this is just a humorous saying. A recursive computation is oen that is explicitly described in terms of itself. For example, the game function that we need to write:

```
def game(secret, left):
    if left == 0:
        print "Sorry, you lost. The number was:", secret
    else:
        guess = int(raw_input("Guess: "))
        if guess > secret:
            print "Try lower..."
            game(secret, left-1)
        elif secret > guess:
            print "Try higher..."
            game(secret, left-1)
        else:
            print "You got it! Secret number was:", secret
```

6. How do you find if a number divides another? What is an odd number? Write a function that takes one argument (a positive integer) and reports if the argument is odd or

not.What is a prime number? Write a function that takes one argument (a positive integer) and reports if the argument is prime or not.

Just looking at the remainder when we divide by 2 is enough to determine if a number is even or odd. A prime number is one that has no other divisors except 1 and itself. Here's how we could determine if a number is prime or not (we're talking integers, of course):

```
def prime(num):
    for val in range(num-1, 0, -1):
        if num % val == 0:
            break
    return val == 1
```

7. Write functions to print scalable letters Z, M, E, L, C, F, W and the digit 4 as square patterns of size `n` (where `n` is the only argument of the function, the size).

```
def z(i, j, size):
    return i == 0 or i == size-1 or
           i+j == size-1

def m(i, j, size):
    return j == 0 or j == size-1 or \
           i+j == size-1 and i <= (size-1)/2 or \
           i-j == 0      and i <= (size-1)/2

def e(i, j, size):
    return i == 0 or i == size-1 or \
           j == 0 or (j < size/2 and i == size/2)

def l(i, j, size):
    return j == 0 or (i == size-1 and j < size/2)

def c(i, j, size):
    return i == 0 and j < 2 * size/3 or \
           i == size-1 and j < 2 * size/3 \
           or j == 0

def f(i, j, size):
    return  i == 0 or \
           (i == (size-1)/2 and j < size/2) or \
            j == 0

def w(i, j, size):
    return  j == 0 or j == size-1 or \
           (i+j == size-1 and i>=size/2) or \
           (i-j == 0      and i>=size/2)

def four(i, j, size):
    return i+j==(size-1)/2 or \
           i==size/2 and j <3*size/4 \
           or j==size/2 and i >size/4
```

```
dict = {}

dict['z'] = z
dict['m'] = m
dict['e'] = e
dict['l'] = l
dict['c'] = c
dict['f'] = f
dict['w'] = w
dict['4'] = four


def pattern(fun, size):
    for i in range(size):
        for j in  range(size):
            if fun(i, j, size):
                print "*",
            else:
                print " ",
        print

for p in dict.keys():
    print "The pattern for: ", p
    pattern(dict[p], 11)
    print '-' * 40
```

8. (See problem 6.) Write a program that asks the user for two integer numbers and stores them in variables `n` and `m` and with `n <= m` and prints all the prime numbers between `n` and `m`. Are you using encapsulation and generalization in your coding[2]?

There's more than one way to do this, but here's one way:

```
def prime(num):
    for val in range(num-1, 0, -1):
        if num % val == 0:
            break
    return val == 1

def search():
    n = int(raw_input("left: "))
    m = int(raw_input("right: "))
    for i in range(n, m+1, 1):
        print i,":",
        if prime(i):
            print "prime"
        else:
            print
```

[2] The circumstances are very suitable, in this case.

9. Write a program that generates random temperatures (between 0 and 32 degrees) to correspond with the days between `n` and `m` (given) and then prints a histogram, like this:

```
>>> histogram(5, 14)
14 ****************** (17)
13 ********* (9)
12 ********* (9)
11 ************** (14)
10 ****** (6)
 9 ******************* (19)
 8 *** (3)
 7 ***************** (16)
 6 ******** (8)
>>>
```

Here's one way, using a dictionary to store the histogram:

```
import random

def histogram():
    n = int(raw_input("start: "))
    m = int(raw_input("end: "))
    days = {}
    for i in range(n, m+1):
        days[i] = random.randrange(33)

    for i in range(m, n-1, -1):
        print str(i).rjust(2) + ":",
        print '*' * days[i], "(" + str(days[i]) + ")"
```

10. Write a function that receives a word (as a string, of course) and prints back the word without the vowels, like this:

```
>>> steno("nectarine")
'nctrn'
>>> steno("blueberry")
'blbrry'
>>>
```

Here's a function that does two things at once:

```
def steno(word):
    result = ""
    other = ""
    for i in word:
        if i in "aeiou":
            other += "(" + i + ")"
        else:
            result += i
            other += i
    print result
    print other
```

Change then the code to surround the vowels with parens instead of not showing them.

```
>>> steno("nectarine")
'n(e)ct(a)r(i)n(e)'
>>> steno("blueberry")
'bl(u)(e)b(e)rry'
>>>
```

This is already part of the function we wrote above.

11. Write a function `count` which gets two arguments, a string and a character, and reports the number of times (perhaps zero) that the character appears in the string. Write another function `contains` which takes a string and a character and determines if the character appears in the string or not. Can you define `contains` in terms of `count`?

```
def count(word, c):
    num = 0
    for i in word:
        if i == c:
            num += 1
    return num

def contains(word, c):
    return count(word, c) > 0
```

12. Write a program that plays Hangman with the user.

```
def hangman(secret, limit):
    mask = '-' * len(secret)
    attempts = 0
    while mask != secret:
        letter = raw_input(mask + ": ")
        newmask = ""
        for i in range(len(secret)):
            if secret[i] == letter:
                newmask += letter
            else:
                newmask += mask[i]
        mask = newmask
        if letter not in secret:
            attempts += 1
            if attempts == limit:
                break
    if mask == secret:
        print "Great job."
    else:
        print "Better luck next time."
```

13. Write a program that shuffles[3] the letters in a word (word jumble)

```
import random

def shuffle(word):
    result = ""
    while word:
        i = random.randrange(len(word))
        result += word[i]
        word = word[:i] + word[i+1:]
    return result
```

14. Write a program to produce circular permutations of a given word[4]

```
import random

def circular(word):
    for i in range(len(word)):
        word = word[1:] + word[0]
        print word
```

15. If `a = [5, 4, 3, 2, 1, 0]` evaluate the following expressions:

- `a[0]` this is the first element of a
- `a[-1]` this is the last element of a
- `a[a[0]]` the first element is 5 so this is the 6th element of a
- `a[a[-1]]` this is the element with index 0 in a (0 is last in a)
- `a[a[a[a[2]+1]]]` this can be done in stages as follows a[2] is 3 plus 1 it's 4, so a[4] is 1 and we have a[a[1]] =a[4]=1

16. Consider the following code: what is the value of a at the end?

```
a = [1, 2, 3]
a[2]=0
a[a[2]] = 5
a[1:2] = []
```

[5, 0] and the key is the last statement (see next problem)

17. What is the difference between `a[1:2] = []` and `a[1] = []`? How would `a` change if we make this change in the last statement of the fragment presented above?

The first replaces a slice (splice) the second changes an element.

---

19. [3] Randomly. For example with input python it could produce poynth

[4] Give it the word apple and it produces: pplea, pleap, leapp, eappl, and apple.

18. Write a function that receives a list of integers and returns their sum[5]. How do you determine the length of a list? If `a = [1, 2, [3, 4], 5]` what is the length of `a`? How many elements does it contain? How many integers? What does `len(a)` return? Same questions after we make `a[1:2] = [[2, 3], 4]`.What does `a[len(a)]` evaluate to[6]? Write a function that receives a list (that could contain nested lists) of integers and reports the total the number of integers at all levels[7].

I am going to simply include the conclusion of our discussion in class:

```
def example(lst):
    if lst == []:
        return 0
    elif type(lst[0]) == int:
        return 1 + example(lst[1:])
    else:
        return example(lst[0]) + example(lst[1:])

def sum(lst):
    if lst == []:
        return 0
    elif type(lst[0]) == int:
        return lst[0] + sum(lst[1:])
    else:
        return sum(lst[0]) + sum(lst[1:])
```

19. Write a function that sorts an array of integers in ascending order.

One method, select sort:

```
def sort(lst):
    sorted = []
    while lst:
        sorted.append(min(lst))
        lst.remove(min(lst))
    return sorted
```

The other one, bubble sort:

```
def sort(lst):
    for i in range(len(lst)):
        for j in range(len(lst)-1):
            if lst[j] > lst[j+1]:
                a = lst[j]
                lst[j]=lst[j+1]
                lst[j+1] = a
    return lst
```

[5] What is list traversal?
[6] Assume `a` is a list: how do you obtain its last element?
[7] For example [1, [2, 3, 5], 4] is a list of three elements with five integers inside.

20. Using nested lists write a function that implements matrix addition. Provide all the ancillary code necessary to test your function complete with random generation of matrices and properly formatted matrices display.

```
import random

def gen(lines, columns):
    mat = []
    for i in range(lines):
        row = []
        for j in range(columns):
            row.append(random.randrange(10))
        mat.append(row)
    return mat

def show(mat):
    for row in mat:
        for elem in row:
            print str(elem).rjust(3),
        print

def add(m1, m2):
    m = []
    for i in range(len(m1)): # index for rows
        row = []
        for j in range(len(m1[i])): # index for columns
            row.append(m1[i][j] + m2[i][j])
        m.append(row)
    return m
```

21. Write a function to generate a matrix of given size (rows and columns) full of random values (positive integers between -50 and 50). Work out a show function that receives a matrix and prints it nicely, with all the values properly aligned, like this:

```
>>> show([[1, 2, 3], [101, 102, 19], [1, 12, 123]])
  1   2   3
101 102  19
  1  12 123
>>> show([[12, 201, 3], [1, 19], [123]])
 12 201   3
  1  19
123
>>>
```

Note the function works even if the argument is not a matrix proper.

Has been provided in the previous exercise.

22. Redo exercises 20 and 21 using dictionaries.

```
import random

def gen(lines, columns):
    m = {}
    m['height'] = lines
    m['width'] = columns
    for i in range(lines):
        for j in range(columns):
            m[i, j] = random.randrange(10)
    return m

def show(m):
    for i in range(m['height']):
        for j in range(m['width']):
            print str(m[i,j]).rjust(3),
        print

def add(m1, m2):
    result = {}
    result['height'] = m1['height']
    result['width'] = m1['width']
    for i in range(m1['height']):
        for j in range(m1['width']):
            result[i,j] = m1[i,j] + m2[i,j]
    return result
```

23. Write a program that reverses the order of words in a sentence[8].

See 25 below.

24. Write a program that reverses the order of the letters in the words in a sentence, leaving the order of the words in the sentence unchanged.

See 25 below.

25. Combine the two programs above into one that reverses the order of the words in a sentence and the order of the letters in each word:

```
Echo> these are the days of miracle and wonder
these are the days of miracle and wonder
rednow dna elcarim fo syad eht era eseht
eseht era eht syad fo elcarim dna rednow
wonder and miracle of days the are these
Echo>
```

[8] Use split from the string module.

```
def one(word):
    drow = ""
    for i in word:
        drow = i + drow
    return drow

def two(sentence):
    words = sentence.split()
    sdrow = []
    for i in words:
        sdrow[0:0] = [i]
    return sdrow

def three(sentence):
    words = sentence.split()
    sdrow = []
    for i in words:
        sdrow[0:0] = [one(i)]
    return sdrow

def four(sentence):
    words = sentence.split()
    sdrow = []
    for i in words:
        sdrow[-1:-1] = [i]
    return sdrow
```

26. What is the difference between

```
a, b = b - a, a - b
```

and

```
a = b - a
b = a - b
```

Parallel assignment in the first case, so a basic swap can be done like: `a,b = b,a`

27. Create an application that simulates playing the world-famous dice game "Craps". In this game, a player rolls two dice. Each die has six faces. Each face contains 1, 2, 3, 4, 5 or 6 spots. After the dice have come to rest, the sum of the spots on the two top faces is calculated. If the sum is 7 or 11 on the first roll, the player wins. If the sum is 2, 3 or 12 on the first roll (called "craps"), the player loses (the "house" wins). If the sum is 4, 5, 6, 8, 9 or 10 on the first roll, that sum becomes the player's "point." To win, a player must continue rolling the dice until the player rolls the point value. The player loses by rolling a 7 before rolling the point.

Here’s the code, followed by an example session with it:

```
import random

class Craps(object):
    def __init__(self):
        self.won = 0
        self.lost = 0
        self.point = 0
    def move(self):
        d1 = random.randrange(6) + 1
        d2 = random.randrange(6) + 1
        sum = d1 + d2
        if self.point:
            if sum == self.point:
                self.point = 0
                self.won += 1
                print "(" + str(d1) + ", " + str(d2) + ")",
                print "You just won."
            elif sum == 7:
                self.lost += 1
                self.point = 0
                print "(" + str(d1) + ", " + str(d2) + ")",
                print "You just lost"
            else:
                print "(" + str(d1) + ", " + str(d2) + ")",
                print "Keep going"
        elif sum in [7, 11]:
            print "(" + str(d1) + ", " + str(d2) + ")",
            print "You just won."
            self.won += 1
        elif sum in [2, 3, 12]:
            print "(" + str(d1) + ", " + str(d2) + ")",
            self.lost += 1
            print "You just lost."
        else:
            print "(" + str(d1) + ", " + str(d2) + ")",
            self.point = sum
            print "You're trying to match", self.point
    def report(self):
        print "Score is:", self.won,"-",self.lost
        if self.point:
            print "You are trying to match", self.point
        else:
            print "You are just starting a new game."
```

And here's how it works:

```
>>> c = Craps()
>>> c.report()
Score is: 0 - 0
You are just starting a new game.
>>> c.move()
(2, 3) You're trying to match 5
>>> for i in range(10):
      c.move()
```

```
(5, 4) Keep going
(4, 1) You just won.
(6, 5) You just won.
(6, 5) You just won.
(2, 3) You're trying to match 5
(3, 1) Keep going
(4, 2) Keep going
(6, 2) Keep going
(5, 6) Keep going
(6, 4) Keep going
>>> c.report()
Score is: 3 - 0
You are trying to match 5
>>> for i in range(10):
      c.move()


(5, 5) Keep going
(4, 2) Keep going
(3, 5) Keep going
(1, 3) Keep going
(1, 5) Keep going
(6, 1) You just lost
(1, 2) You just lost.
(3, 3) You're trying to match 6
(4, 4) Keep going
(4, 1) Keep going
>>> c.report()
Score is: 3 - 2
You are trying to match 6
>>> c.move()
(6, 2) Keep going
>>> c.move()
(6, 2) Keep going
>>> c.move()
(6, 5) Keep going
>>> for i in range(10):
      c.move()


(5, 6) Keep going
(2, 5) You just lost
(3, 6) You're trying to match 9
(4, 3) You just lost
(3, 1) You're trying to match 4
(4, 5) Keep going
(1, 2) Keep going
(2, 2) You just won.
(6, 6) You just lost.
(5, 2) You just won.
>>> c.report()
Score is: 5 - 5
You are just starting a new game.
>>>
```

28. Implement matrix multiplication using dictionaries.

29. Implement the following procedure to construct magic `n-by-n squares;`

- it works only if `n` is odd.
- Place a `1` in the middle of the bottom row.
- After `k` has been placed in the `(i, j)` square, place `k+1` into the square to the right and down, wrapping around the borders.
- However,
    1. if the square to the right and down has already been filled, or
    2. if you are in the lower right corner,

  then you must move to the square straight up instead.

Here's the `5`-by-`5` square that you get if you follow this method:

```
11  18  25   2   9
10  12  19  21   3
 4   6  13  20  22
23   5   7  14  16
17  24   1   8  15
```

Check that the square above is magic.

Calculate the `3`-by-`3`, `7`-by-`7` and `13-by-13 magic squares.`

30. Define a function that takes two points as arguments and reports the distance between them. Points are created and initialized outside, they each contain an `x` and a `y` coordinate.

31. Define a function that takes two rectangle objects (whatever your design choice is for rectangles) and decides if they overlap or not. Same problem for circles.

32. Define a class Circle as a Point (for the center) and a float (the radius) and instruct the circles to be able to tell if they overlap another circle. Define a class Line as two Points. Define a class Triangle as being three Lines coming out of three Points; make Points be able to calculate how far they are from another point, and teach Triangles how to report their area(s), calculated with Heron's formula (given). Define a function that finds the

midpoint[9] of a Line (segment[10]). Define a Fraction[11] as a pair of numbers and teach Fractions how to add themselves to another Fraction, producing a new Fraction (sum).

33. Define a class Robot that has a direction and can move forward. Then create such an object and simulate a random walk for it[12]. When the random walk is over report the distance from the robot's location and the location where the random walk started.

34. Define a class of objects that implement the game of Nim. When the object is created the height of the pile of marbles is specified. The user moves first. When the object is asked to process a move it will determine whose turn it is and if it's the user's turn it will ask for the number of marbles to be removed. One can only remove up to half the size of the pile, but at every move one has to remove at least one marble. If the user makes an illegal move the user loses. When it's the computer's turn the computer will make a random legal move. Whoever brings the pile of marbles to size one wins the game.

Here's how my game runs:

```
>>> game = Nim(50)
>>> game.move()
How many? 24
The height is now: 26
The computer moves.
>>> game.move()
The computer chooses 7
The height is now: 19
The user moves.
>>> game.move()
How many? 9
The height is now: 10
The computer moves.
>>> game.move()
The computer chooses 3
The height is now: 7
The user moves.
>>> game.move()
How many? 1
The height is now: 6
The computer moves.
>>> game.move()
The computer chooses 2
The height is now: 4
The user moves.
>>> game.move()
```

[9] That is, it returns a Point.
[10] A segment is the same as a Line in this context.
[11] Define a gcd function that finds the greatest common divisor of two numbers.
[12] There's more than one way to do it, so implement the most appealing of these.

```
How many? 2
The height is now: 2
The computer moves.
>>> game.move()
The computer chooses 1
The height is now: 1
The user moves.
>>> game.move()
user has lost the game.
>>>
```

35. Define a class of objects that implements a Matrix type:

```
>>> m = Matrix(3, 5)
>>> m.show()
7 6 6 9 5
7 1 1 8 2
2 7 9 2 6
>>> m = Matrix(4, 2)
>>> m.show()
7 2
9 8
4 1
2 9
>>> m = Matrix(3, 3)
>>> m.show()
2 3 4
6 2 0
2 3 7
>>>
```

36. Define a class of objects called `Clock`. An object of type `Clock` stores time (hours and minutes) in military time format, in two instance variables of type `int`. Objects of type `Clock` have two instance methods: `report` which is simply returning the time, and `tick` which advances the clock by one minute. The constructor for class `Clock` takes one argument, a `String` that represents the time the clock is originally set to. Write a test program too, that illustrates how your objects are working (tick the clock 10-20 minutes and show it).

37. Nobody bounces like Tigger! Years of research have finally revealed the special mechanism of Tigger's bouncy step. You are to design a `Tigger` class that implements this unique movement, which I will describe below. A `Tigger` always starts in a random point (with coordinates `x` and `y`). When it decides to `bounce` a `Tigger` changes its `x` and `y` by the following rule(s): `x` becomes the sum of the squares of its digits and `y` becomes the

sum of the squares of its digits. Example: if `x` is `37`, then one `bounce` turns `x` into $3^2 + 7^2$ (`=9+49=58`). Both `x` and `y` change by this rule. And the `bounce` goes on (as expected).

38. Define a class Fraction whose initialization method ensures that the fraction created is already in its lowest terms. Teach these fractions to add, subtract, multiply and divide. Add a `__str__()` function that prints Fractions nicely (`-2 / 3` or some such thing). Note: see also problem 32.