1.8 Math Class

Several values and functions that are commonly associated with mathematics are useful in writing computer programs. In python, these are included in the math library. It's common in python and nearly all other programming languages to have many extra libraries that contain functions to help the programmer. In python, in order to use a library, an import statement must be made at the beginning of the python file. To import the math library, we simply add the line:

```
import math
```

at the beginning of our python file.

Let's look at some of the functions and constants available to us from the math library in python:

Note: A complete list is located at http://docs.python.org/library/math.html.

math.pi - an approximation of the ratio of the circumference to the diameter of a circle.
math.e - an approximation of the base of the natural logarithm.

math.ceil(x) - Returns the smallest integer greater than or equal to it, as a float.
math.fabs(x) - Returns the absolute value of x.
math.factorial(x) - Returns x factorial, which is 1 * 2 * 3 * ... *x.
math.floor(x) - Returns the greatest integer less than or equal to x as a float.

math.exp(x) - Returns e^x.
math.log(x) - Returns ln x.
math.log(x, base) - Returns $\log_{base} x$.
math.sqrt(x) - Returns the square root of x.

math.cos(x) - Returns the cosine of x radians.
math.sin(x) - Returns the sine of x radians.
math.tan(x) - Returns the tangent of x radians.

As is indicated in this list above, in order to call these functions, we must type "math." before the name of each function. Namely, we must specify from which library the function is from, so that it doesn't get confused with other functions with the same name.

Let's look at a couple examples of python programs that make calls to the math library.

Circle Area and Circumference Example

A standard formula taught to all geometry students is the area of a circle ($A = \pi r^2$). In this program, we'll ask the user to enter the radius of a circle and print out the corresponding area and circumference ($C = 2\pi r$). We'll use the math library's value for pi in our calculations.

```
# Arup Guha
# 6/23/2012
# Calculates the area and circumference of a circle, given its radius.

import math

radius = int(input("What is the radius of your circle?\n"))

area = math.pi*(radius**2)
circumference = 2*math.pi*radius

print("The area of your circle is ",area,".",sep="")
print("The circumference of your circle is ",circumference,".",sep="")
```

Here is one sample of this program running:

```
>>>
What is the radius of your circle?
5
The area of your circle is 78.53981633974483.
The circumference of your circle is 31.41592653589793.
>>>
```

Koolaid Example Revisited

One of the most difficult issues we faced in the Koolaid program was correctly calculating the number of cups we had to sell to meet our profit goal. We avoided an "off by one error" with the following line of code:

```
num_cups = (target + profit_per_cup - 1) // profit_per_cup
```

If you carefully think about it, what we really wanted to do was a regular division between the variables target and profit_per_cup, but we just wanted to take the ceiling of that value! Namely, if we needed to sell 7.25 cups to hit our goal, we really couldn't sell .25 of a cup, we just have to sell the whole cup for a total of 8 cups! Now, with knowledge of the math library, we can rewrite this line of code to be much easier to understand, as follows:

```
num_cups = int(math.ceil(target/profit_per_cup))
```

Finally, note that the ceil function in the math library returns its answer as a float. Since we want to print out an integer, we can simply call the int function on the result we get from the ceil function.

Number of Possible Meals

Many restaurants boast that they offer many, many meal choices. Typically, this simply means that for the "meal", they can choose a certain number of items out of a whole batch. (For example, you may be allowed to choose 3 distinct side items out of 15 possible ones.) Mathematically, the number of ways to choose k items out of n (without repeats) is $\binom{n}{k}$, which is read as "n choose k". (Also, some books use the notation ${}_nC_k$ to denote a combination.) The formula for calculating a combination is as follows: $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

For our program, let's assume that a combo meal includes a number of choices of appetizers out of a total AND a number of choices of entrees out of a total. We'll ask the user to enter the following four values:

1) Total possible number of appetizers
2) Number of appetizers allowed for the combo
3) Total possible number of entrees
4) Number of entrees allowed for the combo

We can use the factorial function to calculate both of the necessary combinations. After calculating both combinations, we can just multiply them to get our final answer, because for each possible

choice of appetizers, we can match it with each possible choice of entrees. (You can visualize this with a 2 dimensional grid, with the rows labeled by all appetizers and the columns labeled by all entrees. Each square in the grid corresponds to a unique meal, and clearly the number of meals listed is the product of the number of rows and columns of the grid.) In this example, we'll introduce one final modification to our program. In most programming languages, code starts by executing from a function main. While python doesn't require this, it's good practice to get into the habit of defining a function main anyway. It will help when transitioning to other languages and once the python programs you write exceed a few lines, organizationally it will be advantageous to have a main function. In order to incorporate this into your program, simply write the following line right before your program instructions:

```
def main():
```

Python requires consistent indenting, thus every statement inside of main must be indented. A standard indentation is 4 spaces or a tab. Once you've finished writing your code in main, you must call the function main, since what you've done is just define it. But, defining a function doesn't mean it will get used. It only gets used if it gets called. Here is how to call the function main:

```
main()
```

Putting this all together, we have the following program:

```
# Arup Guha
# 6/26/2012
# Calculates the number of possible combo meals.
import math

# Most languages define a function main, which starts execution.
def main():

    # Get the user information.
    numapps = int(input("How many total appetizers are there?\n"))
    yourapps = int(input("How many of those do you get to choose?\n"))
    numentrees = int(input("How many total entrees are there?\n"))
    yourentrees = int(input("How many of those do you get to choose?\n"))

    #Calculate the combinations of appetizers and entrees.
    appcombos = (math.factorial(numapps)/math.factorial(yourapps)
                 /math.factorial(numapps-yourapps))
    entreecombos = (math.factorial(numentrees)/math.factorial(yourentrees)
                    /math.factorial(numentrees-yourentrees))

    # Output the final answer.
    print("You can order", int(appcombos*entreecombos), "different meals.")

main()
```

One other new feature you'll notice in this program is a single line of code spanning two lines. This occurs with both the assignment statement for appcombos and entreecombos. In order to get python to recognize that the entire expression actually belongs on a single line, an extra set of parentheses are used. Though there are other ways to indicate multiple lines belong to a single line of code, this is the preferred way. The other two ways are splitting the line at a comma, if a comma is already in the expression and adding a single backward slash at the end of each physical line as a separate token. Here is an example of the latter:

```
appcombos = math.factorial(numapps)/math.factorial(yourapps) \
                /math.factorial(numapps-yourapps)
```