**Discussion 7: Intro to Recursion**

**Factorials are Factorials Times Factorials**

**Factorials are defined as the product of a positive integer and all consecutive smaller  positive integers. For example, factorial (5) = 5 x 4 x 3 x 2 x 1. Fill in the code below to  recursively compute a factorial. Don’t worry about the case of n < 1.**

factorial(n):

if n==1 :

report (1)

else:

report(n\*factorial (n-1)

**PalindromeemordnilaP**

1. **A palindrome is a word that is spelled the same way forwards and backwards. In other words,  the first letter must equal the last letter, the second letter must equal the second to last letter  ... etc. For the purposes of this problem, all zero-letter and one-letter words are palindromes.**

def palindrome (string):

    a=0

    b=-1

    for item in string:

        if (string[a]==string[b]):

            a+=1

            b-=1

        else:

            return (False)

        return (True)

**B) Fill in the progression of calls to: is (racecar) a palindrome?**

**ing the above information, fill in the recursive palindrome function.**

**You have access to  the two functions below.**

is (racecar) a palindrome?

is (aceca) a palindrome?

is (cec) a palindrome?

is (e) a palindrome?

TRUE

**Where are These Cats Coming From?!**

**In the following exercise, we will address how to construct and how to think about fractals  recursively. You may assume that the sprite starts off at the leftmost part of each level, facing  right. *Note that each level is 1/3 of the size of the previous level.***

https://lh4.googleusercontent.com/cT_j6o9Tj607eJhQkiWvOTOsdss4UVnSHnPHL9lcxptgmQFYlcS5vpSmWGbJ04EG2J_ulJqjVU1QjqBwCF3e29-0tGav6CiSx-liMJuABDozoEQpziyaSP7RVbv_JaJ4wtO7nwBm

***Level 1***

https://lh6.googleusercontent.com/7aRTUHPPp9ymfEj-q2hhULgYzpDXUItuJNMQS3jOML_O2ZzcITOwI6ddbeQ0v-g9QYu8hnYDbP4qHgeUjFxS84cOlscQP7ArkYAioIwDhJlljXH9X_LQF0ltm5rMNyeFiWdTLCfb**Level 2**

https://lh6.googleusercontent.com/Z7RKeCh5sxHWviGjHiIBHTQCaLaVzhSDzJqDotlntE9nQiArfog7fMHVC8P6wntEZd3xkfzk2TfEgyc_yA-xBfLA9irQ_hC0XQNUP7HErth4Pnmppc80rLXqrw0V2NUZar5Eb1HC**Level 3**

**Level 4** https://lh5.googleusercontent.com/oXdaDtE6SM4WyeXIr2kEKTE5IlKXa_zoKdKJ8cmtXXvnG5pm4qFZEC6U3TS3wN7YZS-Tzepjvi-houtCTmal2VTmD593P1WrUbs7W1uBLxpJHjR4-kf30Gc8gLGmmUatCuppkp13

1. Which level corresponds to the base case?

Level 1

2) For our base case, in what direction does our sprite start and end in?

Right (90°)

3) In each level, circle each instance of the previous level. Each of these instances refers to one recursive call.

See above

4) What does the sprite do between each of the recursive calls?

Draws a horizontal line

**Challenge Problems**

**1) Write the function “Boring Multiply,” which takes as input a number and a list,**

**and recursively  multiplies every item of the list by the number. It should**

**output a new list containing the  multiplied values (in order) without modifying**

**the input list.**

def mal (Number, list):

    i= [Number\*a for a in list]

    return (i)

2) Now the real fun begins. Write the function “Index Multiply,” which takes as input a list, and recursively multiplies every item of the list by its index (i.e., position) in the list. It should output a new list containing the multiplied values (in order) without modifying the input list. This problem is harder than it may seem at first glance…don’t be afraid to think creatively!

def Index \_ Multiply (list):

    list1 = []

    b=0

    a=1

    for i in list:

        list1.insert(b, a \* i)

        a+=1

        b+=1

    return(list1)