Worksheet 3

Tree Recursion

Tree Recursion uses the same logic as recursive functions. However, the difference is that we use two different recursive cases to build our function. This can best be seen with an example implementing a function that determines the nth Fibonacci number.

```
def fib(n):
        if n == 0:
                return 0
        elif n == 1:
                return 1
        else:
                return fib(n - 1) + fib(n - 2)
```

Let's break it down. Remember, a Fibonacci number is defined as the sum of the two numbers before it.

1. **Base Case: We have two.**

Our first base case checks the lowest possible n value we can reach. If n is 0, there is no 0th Fibonacci number, so we return 0. If n is 1, then we return the first Fibonacci number, 1.

2. **Recursive Calls**

Remember, a Fibonacci number is just the sum of the two numbers before it. Let us generalize this for our function. We know the function 'fib' will return the nth Fibonacci number (using our recursive leap of faith). Therefore, we just add fib(n-1) and fib (n-2), or the two Fibonacci numbers before the nth number.

This is best seen in practice!

Code Writing Questions (Recursion & Tree Recursion)

```
def rock_fact(n):
        """Return the number of different ways to order the digits in n.
        >>> rock_fact(123) # 123, 132, 213, 231, 321, 312
        6
        >>> rock_fact(4235)
        24
        >>> rock_fact(0)
        1
        >>> rock_fact(9)
        1
        """

        ________________________________

                ________________________________

        ________________________________

                total, ________________ = ________________, ______________

                while ____________________________

                        ________________ = ___________________________

                        ________________ = ___________________________

                ___________________________________
```

```
def weird_pow(x, y):
        """Returns 2^(y-x) for any x and y. If number is less than 1, return
1. Do not use pow(x,y) or the '*' operator.
        >>> weird_pow(2, 3)
        2
        >>> weird_pow(5, 8)
        8
        >>> weird_pow(8, 2)
        1
        >>> weird_pow(0, 0)
        1
        """

        if ______________________________

             _________________________________

        else:

             _________________________________________
```