Lab 07 - Using Recursion

200pts pts total.

Part 1 - Fibonacci Numbers

200pts - Due Nov 1, 2021

An example of a recursive definition is:

```
sum(n) = { 0 if n <= 0
         { n + sum(n-1) if n > 0
```

Then we can build a function that matches this.

```
 1:
 2: def recursive_sum ( n ):
 3:     if n <= 0:
 4:         return 0
 5:     return n + recursive_sum(n-1)
 6:
 7:
 8:
 9: # Automated Test
10: if __name__ == "__main__":
11:     n_err = 0
12:     x = recursive_sum ( 5 )
13:     if x !=  15:
14:         n_err = n_err + 1
15:         print ( "Error: Test 1: sum not working, expected {} got {}".format (  15, x ) )
16:     x = recursive_sum ( 0 )
17:     if x != 0:
18:         n_err = n_err + 1
19:         print ( "Error: Test 2: sum conversion not working, expected {} got {}".format ( 0, x ) )
20:
21:     if n_err == 0 :
22:         print ( "PASS" )
23:     else:
24:         print ( "FAILED" )
25:
```

Given our definition for a Fibonacci number (this is directly from the example in lass last Thursday)

```
fib(n) = { 0 : n = 0
     { 1 : n = 1
     { fib(n-1) + fib(n-2)
```

Implement a recursive function that calculates this in python. Supply an automated test.

Part 2 - Recursive check for palindrome.

A palindrome is a string where the beginning is the reverse of the end of the string.

Examples:

```
abccba
aba
aa
x
```

All of these are palindromes.

Given definition for a palendrone:

```
palendrone(s) = { if the string is length 0 or 1:  True
                { if the string is 2 or longer then:
                      if the first character matches the last
                         and if palendrone(s[2:len(s)-1]) then True
                      else return `False`
```

Using the recursive definition implement a function that returns `True` if the string is a palindrome.

Turn in your code for both parts.