**UNIT 4, PYTHON 3, DISCUSSION ASSIGNMENT**

Computer Science, University of the People

CS 1101: UNIT 4 Discussion Assignment

July 13th, 2022

Programming Assignment Criteria

    Do Exercise 6.4 from your textbook **using recursion** and the is\_divisible function from Section 6.4.  Your program may assume that both arguments to is\_power are positive integers. Note that the only positive integer that is a power of "1" is "1" itself.

After writing your is\_power function, include the following test cases in your script to exercise the function and print the results:

print("is\_power(10, 2) returns: ", is\_power(10, 2))  
print("is\_power(27, 3) returns: ", is\_power(27, 3))  
print("is\_power(1, 1) returns: ", is\_power(1, 1))  
print("is\_power(10, 1) returns: ", is\_power(10, 1))  
print("is\_power(3, 3) returns: ", is\_power(3, 3))

You should submit a script file and a plain text output file (.txt) that contains the test output. Multiple file uploads are permitted. Don’t forget to include descriptive comments in your Python code.

def is\_divisible(a,b):  # This function is from our textbook section 6.4

    if a % b == 0:     # If a % b is 0

        return True    # this function returns "True",

    else:              # otherwise,

        return False   # it will return "False".

#defining is\_power function which takes two arguments.

def is\_power(a,b):  # defining the funtion I will call

    if a == 1 or a==b:# 2 base cases: that both take a true argument combined

  # the only positive integer with a power of '1' is '1', also, if 'a' is equal to 'b', 'a' is a power of 'b'

        return True

    elif b==1:    #base case: so 'b' cannot be 1

        return False

    elif a == b:     #base case: also, if 'a' is equal to 'b', 'a' is a power of 'b'

        return True  # so funtion will return true

    elif not is\_divisible(a,b):  # base case: calling previous defined function to argue these variables

        return False             # if the above call is True, will return False

    else:

        return is\_power(int(a/b),b) # this is a recursive call, it calls itself.

#calling test cases

print ("is \_power(10,2) returns: ", is\_power(10,2))   # will return 'False'

print ("is\_power(27,3) returns: ", is\_power(27,3))    # will return 'True'

print ("is \_power(1,1) returns: ", is\_power(1,1))     # will return 'True'

print ("is\_power (10,1) returns: ", is\_power (10,1))  # will return 'False'

print ("is power (3,3) returns: ", is\_power(3,3))     # will return 'True'

References

Downey, A. (2015). Think Python. Green Tea Press. [**https://greenteapress.com/wp/**](https://greenteapress.com/wp/)