Kjc4 4CREDITS

NARRATIVE FOR HOMEWORK5

D1

Functions reduces code duplication: In my understanding, functions are subprograms containing instructions that are executed anytime the function is called. This means that if I define a function to carry out a specific operation , whenever I need to perform that operation within my program I do not need to continuously write the same code for that operation anymore, I just need to call the function that performs that operation.

Functions save me a lot of time : As a follow up to the above point, whenever I define a function that performs certain instructions, I do not need to spend time writing code to perform that same operation again in my program, all I need to do is call my function.

D3

a)Parameters are included in the definition of a function as well as the call of the function. Basically, I view parameters as these special variables that are used in a function to ascertain what is expected as input for the function and are used in the function definition body to perform the desired operations. They are also used to pass values to the function.

b) formal parameters are the input parameters which are included in the function defintion and are the variables that will be used in the writing of the instructions that the function will execute while the actual parameters are included in the function call and are used to pass values to the function.

c)I think parameters are different form ordinary variables in the following ways;

I observe that python views parameters as means of obtaining inputs, specifying the number of inputs for the function and the position those inputs should be . Parameters are used in passing values to the functions. Variables are used to store values and as long as you follow the standard naming conventions, everything should work well.

PE4

I defined my function sumN(n) and created an accumulator sum with value 0, I then used a for loop with a range(1,n+1) and added the values of num to sum with each iteration then I returned the sum. I then defined my function sumNCubes(n) and created an acummulator cubes\_sum. I created the same for loop as before, but cubed each value of num and added it to cubes\_sum , then returned cubes\_sum. I then asked the user for input and stored it in a variable “user\_input”, used it as input for my two functions and stored the functions’ outputs in two variables “natural\_sum” and “cubes\_sum” then printed them out.

D5

a) This function cubes the input parameter x, stores the output in a variable “answer” and answer is then returned by the function.

b) **def** cube(x):  
 answer = x \*x \*x  
 **return** answer  
  
 y = eval(input(**"please input y :"**))  
 cubed\_y = cube(y)  
 print (**"the cube of y is"**, cubed\_y)

c) The “answer” within the cube function is a local variable existing within a function definition. So the answer variable which can be accessed is the “answer= 4” which was defined later, which is why that was printed out .

PE10

I defined my function acronym(input\_phrase). I then split input\_phrase using split() and stored the output in “split\_phrase”. I created an accumulator “create\_acronym” . I then used a for loop to go through split\_phrases , get the first character of each word and concatenate it to create\_acronym. I applied upper() to create\_acronym and then returned the output.

I then asked the user for input, stored it in a variable “user\_phrase” and used it as input for my acronym function then stored the output of my function in a variable “acronym\_result” and printed it out.