**Csci 1523** **Lab Partner 1(Print):** _____
**Lab on Functions - Lab05A** **Lab Partner 2(Print):** _____

---

This lab contains **??** pages (including this cover page) and **??** problems. Check to see if any pages are missing. Enter all requested information on the top of this page, and put your initials on the top of every page, in case the pages become separated.

You may use your books, notes, calculator or internet sources while completing this laboratory.

Please try to answer the sections clearly and PRINT your answers legibly Please follow these guidelines when completing the sections:

You will work with a partner on this exercise during your lab session. Two people should work at one computer. Occasionally switch the person who is typing. Talk to each other about what you are doing and why so that both of you understand each step.

**Introduction**

This laboratory provides practice with functions in Python. Referencing our other materials the following are characteristics of functions in Python and to an extent functions in other programming languages:

1. Functions are a syntactical means of representing a *unit* of code containing logic which is needed at various times in program execution or logic which is highly specialized requiring it to be separated from other source code.

2. Functions have an entry point to which we pass program execution and into which we may pass data in the form of arguments copied to local formal parameters.

3. Functions have an exit point from which program control is passed back to the line of source code following the line of source which called the function. Data may be passed back to this point using function return values or altered arguments (if these arguments are mutable, i.e. lists)

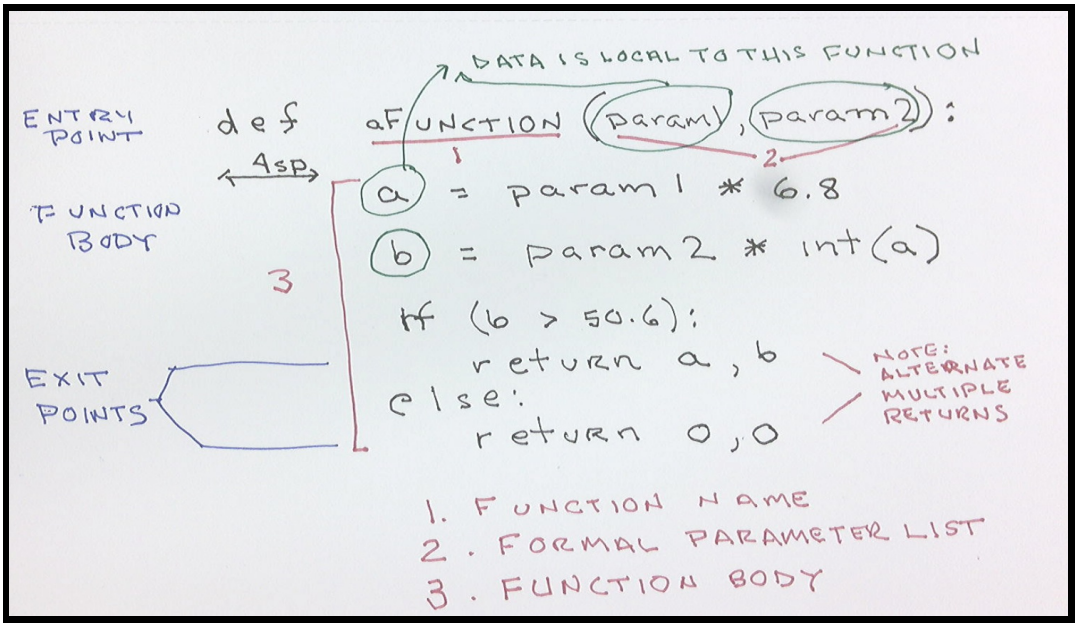
4. Functions are uniquely named via their *signature* that is the combination of the function name and the formal parameter list.

5. Function naming follows the rules for creating variable names in Python.

6. Functions, while executing have their own memory space for the variables declared within the function or as parameters in the formal parameter list. These variables only come into existence once the function is executed, are not accessible from outside the function (unless specifically declared within the function as global variables) and cease to exist once the function is exited.

7. Functions can receive zero or more arguments and can return zero or multiple values.

Below find a function drawn and hand labeled with many of the aspects discussed above.



It is generally a good practice to use psuedo-code and write out functions by hand prior to attempting to write the Python code for them. It is also a good practice to use the hand written functions to test the proposed unit of code prior to implementing it. Also it is good practice to establish test data prior to running the new functional unit so that its predicted behavior can be determined examined prior to its execution in order to confirm its validity.

Below find the source for this function:

Listing 1: Example Python Function

```
# Implementation of the hand-drawn sample above

def aFunction(param1, param2):
    # now define the body of the function
    # note a, b, param1 and param2 are local to this
    # function. They are only available here and cannot
    # be accessed from outside the function code.
    a = param1 * 6.8
    b = param2 * int(param1)
    if (b > 50.6):
        return a, b # here we use multiple return values
    else:
        return 0, 0

# now some source to execute the function defined above
print(aFunction(10.6, 34.2))
```

## Laboratory Exercises

1. The effective use of functions is an important component to developing efficient and maintainable programs and systems. Understanding how functions receive, process and return data to a calling program or function is key to understanding how to use them effectively. In this section we will examine some basic functions called by an application program and answer questions about them..

   (a) Examine the Python program below which utilizes a function and predict the values of all variables after the function is called.

Listing 2: Calling Python Functions

```
# Question 1 Python function example



# First lets define a function

def Question1Function(aaa, bbb, ccc):
    aaa = aaa * aaa
    bb = aaa * bbb
    ccc = 'String from function'

    return aaa, bb, ccc

# set some data values within the calling program and
# pass them to the function

AAA = 3.0
BBB = 34.0
CCC = 'String for question 1'


# now call the function passing in the values given
# and returning the 3 values from the function

aa, bb, cc = Question1Function(AAA, BBB, CCC)
```

In the space provided review the code above and report the values here:

AAA : _____
BBB : _____
CCC: _____
aaa: _____
bbb: _____
ccc: _____
aa: _____
bb: _____

cc: _____

(b) After completing the above, use IDLE to implement this function and check your work. If any of your answers are incorrect, rework the appropriate questions.

2. As mentioned in our text Python functions cannot modify an argument (this was shown above) unless the argument is a mutable type. In the example below we use lists as arguments.

(a) Examine the Python program below which utilizes a function and predict the values of all variables after the function is called.

Listing 3: Mutable Arguments to Python Functions

```
# Question 2 Python function example



# First lets define a function

def Question1Function2(aaa, bbb, ccc):
    aaa[4] = -35
    bbb[1] = 0
    ccc = 'a new string to view'
    return sum(aaa), min(bbb), ccc

# set some data values within the calling program and
# pass them to the function

AAA = [4.5, 5, 6.9, 23.5, 45]
BBB =  [ 98, 23, 45, 22, 44]
CCC = 'String for function 2'


# now call the function passing in the values given
# and returning the 3 values from the function

aa, bb, cc = Question1Function2(AAA, BBB, CCC)
```

In the space provided review the code above and report the values here:

AAA : _____

BBB : _____

CCC: _____

aaa: _____

bbb: _____

ccc: _____
aa: _____
bb: _____
cc: _____

(b) After completing the above, use IDLE to implement this function and check your work. If any of your answers are incorrect, rework the appropriate questions.

3. As mentioned in our text Python functions can set default values for parameters and functions can call parameters in an order other than they appear in the formal parameter list. In the example below we use exercise there options.

(a) Examine the Python program below which utilizes a function and predict the values of all variables after the function is called. In this case we have the function called 3 times. Please complete the following for each function call.

Listing 4: Mutable Arguments to Python Functions

```
# Question 3 Python function example



# First lets define a function

def Question1Function3(aaa=10.0, bbb=4.0, ccc='no string'):
    return aaa, bbb, ccc

# set some data values within the calling program and
# pass them to the function

AAA = 20.0
BBB = 8.0
CCC = 'a new string'


# now call the function passing in the values given
# and returning the 3 values from the function

# first call to function
aa, bb, cc = Question1Function3(AAA, BBB, CCC)

# second call to function
aa, bb, cc = Question1Function3()

# last call to function
aa, bb, cc = Question1Function3(aaa=25, ccc='new string' )
```

```
30 print(aa, bb, cc)
```

i. In the space provided report the values of the variables after the first function call here:

AAA : _____
BBB : _____
CCC: _____
aaa: _____
bbb: _____
ccc: _____
aa: _____
bb: _____
cc: _____

ii. In the space provided report the values of the variables after the second function call here:

AAA : _____
BBB : _____
CCC: _____
aaa: _____
bbb: _____
ccc: _____
aa: _____
bb: _____
cc: _____

iii. In the space provided report the values of the variables after the third function call here:

AAA : _____
BBB : _____
CCC: _____
aaa: _____
bbb: _____
ccc: _____
aa: _____
bb: _____
cc: _____

iv. After completing the above, use IDLE to implement this function and check your work. If any of your answers are incorrect, rework the appropriate questions.

4. Python program with functions

Develop a Python program which contains a Python function, *poundsToMetric* which converts weights given in pounds to kilograms and grams.

For example rather than print out 2.2 kilograms the correct answer would be 2 kilograms and 200 grams.

To aid your work the following conversions hold:

1. 1 pound = 0.455 kilograms
2. 1 kilogram = 1000 grams

Your program should prompt the user for the number of pounds and output the results in kilograms and grams.