Modular Programming

Examine the following code:

```
number1 = int(input("Please enter a number."))
number2 = int(input("Please enter a second number."))
number3 = int(input("Please enter a third number."))

sum12 = number1 + number2
ave12 = sum12/2

sum13 = number1 + number3
ave13 = sum13/2

sum23 = number2 + number3
ave23 = sum23/2
```

Notice the repetition of code?

The idea of modular programming is the process of *subdividing a computer program into separate sub-programs*.

The above program can be subdivided by using a function called average that will find the average of any two numbers. Our program now looks as follows:

```
def calcAverage(a, b):              # defined calcAverage with variables a and b sent
   sum = a + b
   average = sum/2
   return average                   # returns the calculation back

# main program starts
number1 = int(input("Please enter a number."))
number2 = int(input("Please enter a second number."))
number3 = int(input("Please enter a third number."))

ave12 = calcAverage(number1, number2)   # function call
ave13 = calcAverage(number1, number3)   # the value returned
ave23 = calcAverage(number2, number3)   # is assigned to the variable
```

In the code above, a function named average was created using the **def** keyword.

```
def functionName(variables sent):
        body of the function
        return value (if needed)
```

The function can now be “called” or executed by the following code:

```
functionName(variables to send)
```

If the function returns a value, then you can set a variable equal to it.

```
variable = functionName(variables to send)
```

For example:
We have used the input function which someone else has defined.
We know it takes a string sent to it as a parameter and returns a string.

```
name = input(“Please enter a name.”)
```

Another thing to note is that the “variables sent” (or **formal parameters**) and the “variables to send” (**actual parameters**) do not need to have the same names. The value stored in the actual parameter gets sent and stored into the variables of the formal parameters.

Also of note, in the example above, a and b are called **local variables** (to the calcAverage function) and can only be used in that function.

And a third note! Functions have different names in different programming languages.
We may use the following terms to signify a function:

    Function, method, subroutine, subprogram

Exercises

1) What is the output of the following program?

```
def doAdd(num1, num2):
   return num1 + num2

def doSubtract(num1, num2):
   return num1 - num2

def doMult(num1, num2):
   return num1*num2)

total = doAdd(5, 7)
diff = doSubtract(14, 8)
mult = doMult(total, diff)
print(mult)
```

For each of the following, create a function that will do the indicated task and use each in a program (add a function call).

2) **a)** Write the function called printStars() that displays 55 asterisks in a line.

b) Write a function called lineOfStars(n) that displays n stars in a line.

c) Use both printStars() and lineOfStars(n). In the main program, ask the user for the number of asterisks they'd like to see and then display 55 asterisks, followed by the number of asterisks requested then another 55 asterisks.

3) **a)** Write a function that takes four parameters, the x and y value of one point, and the x and y value of a second point and returns the slope between the two points.

Note: there is a predefined method in `math` that allows you to take the square root of a number (use `math.sqrt`).

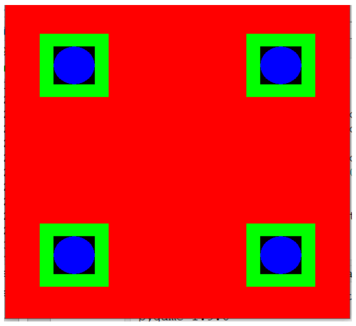
b) Write a function that takes four parameters,
the x and y value of one point, and the x and y value of a second point and returns the distance between the two points.

c) Write a function that takes a point as an ordered pair in string form:
eg: (3, 5) and returns the x and y values separately.
Note: you can return multiple values. Eg: return a, b

d) Write a program that asks the user to input two points (as shown below) and then proceeds to find the distance and the slope between the points. You are to use the three functions from the previous parts to do so.

```
Enter the first coordinate: (0, 0)
Enter the second coordinate: (2, 1)
The distance between points (0,0) and (2,1) is 2.236.
The slope between points (0,0) and (2,1) is 0.500.
```

4) Produce the following code in pygame where the designs in the corners are written in a function called drawDesign which takes in the x and y locations of the top left of the design as parameters.



5) Write a function that returns the number of occurrences of the capital letters A, B, C, D or E in a string. Using this function, get a string from the user and output how many of those capitals are in the string, total.

For example:

```
Please enter a string.   All BaD DanCErs eAt KALE
There were 9 occurrences of A, B, C, D or E.
```

6) Write a centre function that takes a string and a field size as parameters then prints the string centered within that field, with dots "." on either side. You may assume that the field size is larger than the length of the string.