```
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
```

function defined - variables are not remembered anywhere else

main program

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

↙ meaningful

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

- many lines

- define all functions above the main program

- never define a function inside another function!!!

- very poor form!

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

functionName(variables to send) - function call

-in main program

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.

↙ parameter

name = input("Please enter a name.")

↑

def input(a)

return string

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.

parameters sent

```
ave12 = calcAverage(number1, number2)   # function call
```

actual parameters

```
def calcAverage(a, b):
    sum = a + b
    average = sum/2
    return average
```

local variables

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.

```
def calcAverage(a, b):
    sum = a + b
    average = sum/2
    return average
```

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

return

- with no value
- Null value

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)
```

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

↑ no parameters

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

↑ has a parameter

c) Use both printStars() and lineOfStars. 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.