# Lesson 6 - Structured programming

|  |  |
| --- | --- |
| **Lesson Outcomes**  In this lesson you will learn:   * how to structure a program; * how to create user-defined functions; * the use of parameters in functions; * the difference between global and local variables. | **C:\Users\Graham\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\X6CHINOH\MC900441498[1].png** |

## Introduction to Structured Programming

When creating a program it is important that some sort of structure in the design and approach is considered. For a program to be maintainable it has to be easy to follow and debug; and this is where the layout and structure is the key.

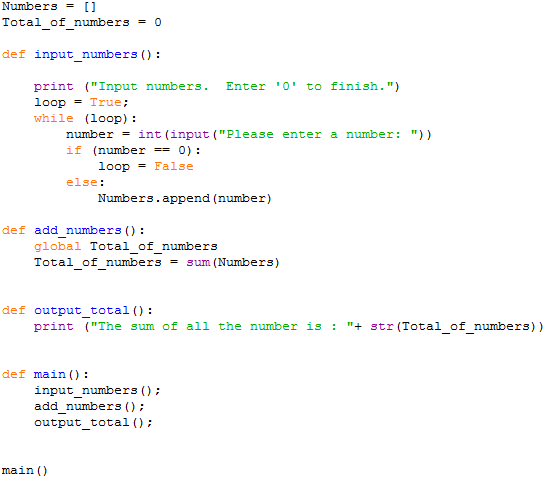
To help us design a structured a program we can use charts as visual aid in identifying the key processes and sequence of events. For example, let’s look at a simple program which prompts the user to enter a series of numbers, adds them together, and outputs the result:

Figure 1 Structured Chart

The chart, starting from left to right, splits the program up into three main parts. We could break down the chart further by adding sub functions to “input numbers” e.g. input a number, store in a list. Obviously, for a more complex program, you would have many sub-components potentially with 4 or 5 levels.

Once the structure has been designed the next stage is to implement it in Python.

## Defining Functions

In order to implement structure into our program we have to use functions. Functions are sub-routines or blocks of code which perform a specific operation. If we take the structured chart in **Figure 1**, there are three separate functions which we need to include in our Python program:

When the program starts this is the first command to be run. This calls the “main” function.

Main Function which calls all the other functions in order.

Simply applies the “sum” function to the list

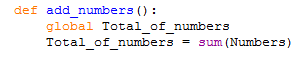
Asks the user for a series of numbers and stores in a list.

Initialise global variables

Figure 2 Add Numbers Program

Although not exactly the same as our structured chart, the three main functions are there – input\_ numbers, add\_numbers and output\_total. Each of these functions is called from the main() function which represents the first box in the **Figure 1 Structured Chart.**

Each function is **self-contained** e.g. performs a single operation. You will notice the layout for a defined function is as follows:



Global variable used. This will be discussed later.

Function name. Brackets contents are left empty as no parameters needed.

## Global and Local Variables

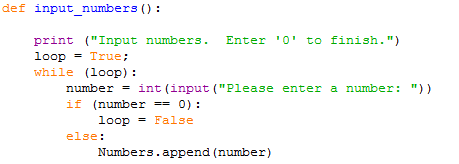
When creating functions it is very important you understand the concept of **GLOBAL** and **LOCAL** variables. Without knowing the difference you could find yourself defining variables which are inaccessible in other functions which may return some very strange results.

A **Global variable** is a variable which is accessible and defined for use across **the entire program**. If we look at the above program you will notice two variables are defined at the beginning:

Numbers starts off as an empty List which is used throughout the different functions. We define it **OUTSIDE** a function at the beginning of the program to make it **GLOBAL**. The next variable keeps a TOTAL value, which again is accessible by all the functions in the program i.e. a **GLOBAL variable**.

When accessing a global variable within a function you have to use the command **GLOBAL:**

In this function the command **GLOBAL** is used to reference the “Total\_of\_numbers” variable outside the function. If this command wasn’t specified then **a new LOCAL variable** “Total\_of\_numbers” would be initialized i.e. two different variables with the same name, one local the other global. You can see how this may get confusing.

**Local variables** are variables defined within the function – the contents can only be accessed and used within that function. For example:

LOCAL VARIABLE called “loop” defined within the function. Set as a BOOLEAN value e.g. True or False

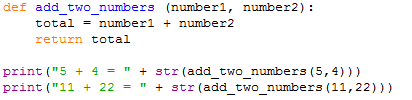
GLOBAL variable as defined outside the function.

## Parameters

As you can see when mixing global and local variables across functions confusion can arise. In the **Figure 2 Add Numbers Program** example you have two variables called **“Total\_of\_numbers”** with one as a global and the other local. Having to refer to one as global and the other local could easily get mixed up. If this was repeated in a large and complicated program then the potential for bugs is a likely possibility. Without the use of Parameter this is the only way the program can function. In programming this is considered bad practice.

When creating a good structured program each function should be self-contained, using only local variables. To pass local variable contents from one function to another we need parameters.

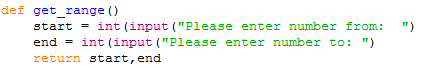
Parameters are a great way of passing and return values from functions. It allows us much more flexibility in the **re-usability of code**. For example, I could write a simple function to add two numbers together:



Here we have the function add\_two\_numbers(number1, number2). The parameters are specified in the brackets as number1 and number2. Within the function we can then use these parameter variables as normal. When the function is called, two parameters are required. In the example above the function is called first with the parameters “5” and “4”. In the function the value “5” will be assigned to number1, and the value “4” assigned to number2.

To return a value or series of values from a function we use the **return** command followed by the variables we want to return e.g. in the above we return the contents of the ***total*** variable.

The returned value is then used in the expression where the function was originally called, in this case the two print commands at the end of the program. You will notice the **str** function is used to convert the returned value from an **integer** to a **string.**

Sometimes you may wish to return more than one value, in which case you specify the each value separated by a comma, this is repeated in the original call e.g.:

Two Variables assigned to returned values

Two variables returned

### Tasks

6.1 Write a function that will take three numbers as input and return the largest number. Incorporate this function into a simple program which ask the user for three numbers and displays the largest.

6.2 Write a function which given any LIST of numbers will return the average number.

6.3 Using structured programming techniques (think about structure first) construct a program which prompts the user for a series of numbers and then outputs the average of those numbers. You can use the function created in task 6.2 as part of the program.

6.4 Using functions, write a program that asks the user to enter an odd number, validates the number and then prints an inverted pyramid of stars based on that number. For example, entering the value 5 will produce:

\*\*\*\*\*  
 \*\*\*  
 \*

6.5 Write a program to let the computer guess a number the user has thought of, with a range specified by the user. After each guess the user will tell the computer if it higher or lower and continue until the computer guesses the correct number. The program should also output the number of attempts. Ignore validation for incorrect higher and lower user entry.