In this module we will be covering the Void function. We will also look at both local and global variables. We will examine functions that return values, and will look at the math module.

Functions

A function is a group of statements that exist that we can use in a program for the purpose of performing a specific task. We call this modular programming. Dividing up code into sections that can be executed as needed. We can also "store" these sections of code to be reused in other programs.

Void Functions and Value-returning functions

A void function simply executes the statements is contains and then terminates. A value-returning function executes the statement that it contains, then returns a value back to the statement of the original program that "called" the function. The *input* function is a value-returning function. It get the data that the user types on the keyboard and returns that data as a string to the program.

The code for a function is known as a function definition. To execute the function, you write a statement that calls it - such as **input.**

You name your own functions (except for PYTHON defined functions already created).

1. you can't use one of Python's keywords
2. a function name cannot contain spaces
3. the first letter must be A-Z, a-z or underscore
4. after the first charter must use A-Z, a-z, underscore, 0-9
5. uppercase and lowercase are different

Defining a function and calling it

When creating a new function the SYNTAX is:

def *function_name():*

    statement

    statement

    etc.

The first line is the function header. The "statements" are the block that are code for that function. You MUST indent the statements in the block in PYTHON>

When we want to "call" a function - have it executed from our program we:

function_name()

Hierarchy Charts

One of the ways we can visualize the relationships between our programs and any functions that they "use" can be done through a *hierarchy chart* - also called a *structure chart*. This type of chart uses rectangles to represent all of the functions that may be "used - called" from our main program.

Local Variables

One of the very IMPORTANT things about using functions is to understand two terms - Local Variables and Global Variables. With local variables - these are created inside a function and CANNOT be accessed by statements that are outside of the function. Different functions CAN have the same variable names because functions cannot see each other's local variables.

Anytime you assign a value to a variable inside of a function you have created the *local variable.*

Scope

A variable's scope is the part of a program where the variable may be accessed. A variable is only visible (can be used and recognized) by statements within the variable's scope.

Passing Arguments to Functions

An argument is any piece of data that is passed into a function when that function is called (remember to call a function means we want to have that particular function executed). A parameter is a variable that receives an argument that is passed into the function.

The main program has the ARGUMENT (local variable name with data) and the function has the PARAMETER (local variable name) where the data will be sent (it's like an assignment statement). Once the function has the data inside of the parameter (local variable) it can use it in the function. Page 182 of your textbook gives an excellent example of how this works.

You CAN pass multiple arguments (more than one piece of data) to a function to be used. Both the main program (calling program) and the function (called program) MUST have the same number and data types of the arguments and parameters being used. Page 185-186 shows this example.

You may also pass keyword arguments to functions where parameter_name=value You can see this example on pages 189-190.

Global Variables

Global variables are accessible to ALL the function listed in a program. When you begin a program - BEFORE defining the main() program, you can create and assign your Global variables. These are ALWAYS done before the definition of any of your programs or functions. Then EACH part of the program - main and functions can use the Global variable as defined.

HOWEVER - these Global variables can be dangerous to use. It is recommended wherever possible to only use local variables. However - you can create a Global constant variable whose value doesn't change through the program and functions. This will avoid any other other problems with global variables.

Value-Returning Functions

The functions we have looked at before were only those where the main() program called the function and the function performed some action before returning back to the main program. We have certain types of functions where we want to "pass back" calculated values back to the main() program. We use the RETURN statement to return a value back to the main(). You can return multiple values as well.

PYTHON uses a standard library of functions that have already been written for you. You just need to become familiar with these various functions and use them as needed. These are called LIBRARY FUNCTIONS.

Some of the library functions deal with random numbers which can be very useful for many programs. Some of these are *randint, randrange, random, and uniform.*

Random numbers need to be generated with a "seed" value - this value is used in a calculation to generate the random number.

IPO Charts

The book suggests using Input-process-output (IPO) charts. This is simply a table that shows all inputs coming into a function and those going out of the function.

***math* Module**

In PYTHON there is a collection of several mathematical functions together in a Module. To use this module - you MUST name the math module "math" followed by a "." followed by the name of the function.

example: result = math.sqrt(16) (this will pass the value 16 to the square root function in the math module).

Page 219 of your book lists various functions in the math module. We also have math.pi and math.e functions.

Storing user defined functions in a Module

As programs get larger and larger, it might be wise to actually store several functions together into a Module and then use the module.function to use what you need in your program. This is called modular programming.

Menu-Driven Programs

A men-driven program displays a list of operations on the screen, and allows the user to select the operation that they want to be performed. The list displayed on the screen is called a *menu.* you can choose the operation by using an if-elif-else statement.