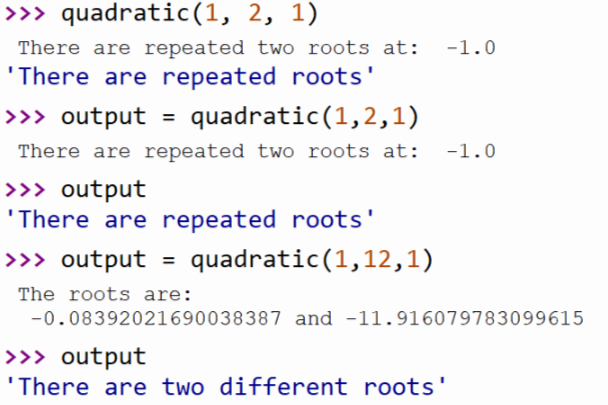
It is what the function return

This is the solution of the quadratic equation.



有用句型：

We can then execute this sequence at any time by referring to the name/variable.

When the function is used in a program, we say the definition is called or invoked.

A parameter is a variable that is initialized when the function is called.

The only way for a function to see a variable from another function is for that variable to be passed as a parameter. (and it is the value that is passed.)

**When the function is called, <name> (< actual-parameter>)**

**The difference: def <name> (<formal-parameters>):**

**<body>**

When python comes to a function call, it initiates a four-step process.

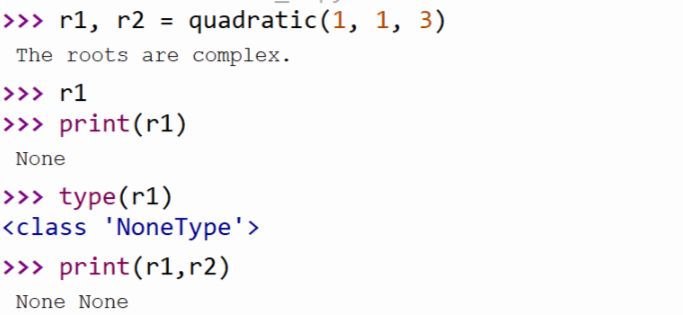
1. The calling program suspends execution at the point of the call.
2. The **formal parameter (a list of variable names)** of the function get assigned the values **supplied by the actual parameters** in the call.
3. The body of function is executed.
4. Control returns to the point just after where the function was called.

All python functions return a value, whether they contain a return statement or not.

(return -> means nothing)

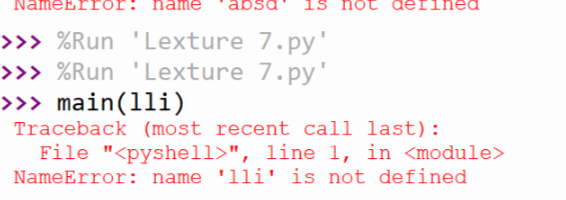
**A function will return one or more results**

**Function without a return hand back a special object, denoted none. (none is nothing)**



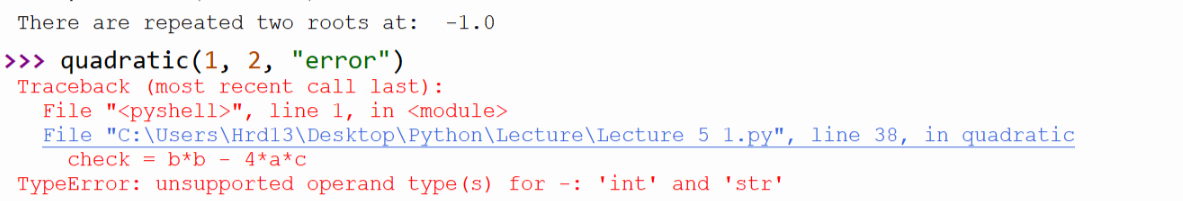
A common mistake of me

The formal parameter needs to be use “” or ‘’ (double quote or apostrophe)



whatever you input, the calling function does not care the type you input,

but if it does not match the type inside the function, the error will occur.



A smart method to deal with the unmatched parameters.

