5.16. Glossary

base case

A branch of the conditional statement in a recursive function that does not give rise to further recursive calls.

data structure

An organization of data for the purpose of making it easier to use.

exception

An error that occurs at runtime.

handle an exception

To prevent an exception from terminating a program by wrapping the block of code in a `try / except` construct.

immutable data type

A data type which cannot be modified. Assignments to elements or slices of immutable types cause a runtime error.

infinite recursion

A function that calls itself recursively without ever reaching the base case. Eventually, an infinite recursion causes a runtime error.

mutable data type

A data type which can be modified. All mutable types are compound types. Lists and dictionaries (see next chapter) are mutable data types; strings and tuples are not.

raise

To cause an exception by using the `raise` statement.

recursion

The process of calling the function that is already executing.

recursive call

The statement that calls an already executing function. Recursion can even be indirect — function *f* can call *g* which calls *h*, and *h* could make a call back to *f*.

recursive definition

A definition which defines something in terms of itself. To be useful it must include *base cases* which are not recursive. In this way it differs from a *circular definition*. Recursive definitions often provide an elegant way to express complex data structures.

tuple

A data type that contains a sequence of elements of any type, like a list, but is immutable. Tuples can be used wherever an immutable type is required, such as a key in a dictionary (see next chapter).

tuple assignment

An assignment to all of the elements in a tuple using a single assignment statement. Tuple assignment occurs in parallel rather than in sequence, making it useful for swapping values.

(Discussion_Questions.html) (Exercises.html)

user not logged in

(DiscussionQuestions.html)

(Exercises.html)