J: Jargon

Gareth McCaughan

Revision 1.8, May 14, 2001

Credits

Introduction

Computing is full of funny words, and of ordinary words used in funny ways. We've tried not to use too much jargon in these sheets, but we haven't always succeeded. So this sheet tries to give brief descriptions of what some funny words mean. If you run across something you don't understand, it's worth taking a quick look at this sheet.

Some of this jargon is general programming jargon. Some is specific to Python.

Lots of these entries refer to other entries, so if you see a word in here that you don't understand you should look it up!

The jargon

A	*argument*	An object passed into a function when you call it. It appears inside the function as a local variable.
	association	A piece of information that says "If you're given X, give back Y". A dictionary is made up of associations.
	attribute	An object that's part of an instance of a class. A little bit like a variable; a little bit like the value in a dictionary association.
B	*body*	The code inside a loop, or function, or other complicated thing. So a "function body" is what gets obeyed when you call a function, and a "loop body" is what gets obeyed each time around a loop.
C	*call*	To "call" a function is to use it. Calling a function is a bit like copying out its definition and doing that.
	class	Classes are kinds of object. (In the real world, things like "computer" and "woman" and "book" are classes.)
	character	A letter, digit or other symbol. A string is made up of a sequence of characters.
	code	The stuff that programs are made of. A "piece of code" is a portion of a program.
	comment	A piece of a program that doesn't do anything, but is just there for people to read. Usually the purpose of a comment is to explain what the code around it is doing.
	condition	Something that might be true or false, like `1>2`.
	conditional	Something that depends on a condition, like an `if` statement.

D	*definition*	A bit of code that tells the computer what something is: what value a variable has, or what a function ought to do, or all about a class.
	dictionary	A Python object a bit like a dictionary, an address book or an encyclopaedia. A dictionary is made up of associations.
E	*element*	An item in a sequence.
	exception	An unusual situation in which something has gone wrong, or a Python object describing such a situation. A pretentious way of saying “error”.
	expression	A calculation that produces a value. The simplest expressions are things like constants (`123`, `'eek'`) and variables (`my_name`). They can get much more complicated: `17*a[a.index(func(99))]>93 and a/(b+c)==2`.
F	*for loop*	A loop whose body gets obeyed once for each item in a sequence.
	function	A set of instructions with a name. You can make the computer obey all the instructions by saying the name of the function.
G	*global variable*	A variable that exists everywhere in your program, not just inside one function.
	graphics	Pictures drawn by a computer.
I	*immutable*	Not allowed to be changed.
	import	To make the things inside a module available for use.
	instance	An object described by a class. In the real world, computers and women and books are instances of the classes “computer”, “woman” and “book”.
	iteration	A single trip through a loop body. When the loop is obeyed, it will usually go through several iterations.
K	*key*	One half of an association in a dictionary: the half you can look things up by. If you say `dict['zog']123`, then the key is `'zog'`.
L	*list*	An object made up of some number of other objects, in order. You can get at them by number.
	local variable	A variable that exists only inside a particular function, not affecting anything outside the function.
	loop	A piece of your program that might get obeyed over and over again.
M	*module*	A bunch of objects with names that you can “import” into your program.
O	*object*	Any piece of information Python can work with, like a number or string or list.
R	*read*	To get information into a program. For instance, if you ask the person sitting in front of the computer to type something in, that’s “reading”.
	return	What a function does when it’s finished. If it “returns a value”, then the function call can be used in an expression.
S	*sequence*	A list or string or tuple. (Actually there are other kinds of sequence, but you don’t need to worry about those.)
	string	A sequence of characters, usually forming a piece of text.
	subclass	A class, all of whose instances are also instances of another class. In the real world, “Apple Macintosh” is a subclass of “computer”, which is a subclass of “electronic device”, which is a subclass of “thing”.
T	*tuple*	A particular kind of sequence, written like this: `(1,2,3)` . Unlike lists, tuples are immutable.
V	*value*	An object. The result of a calculation, or of a function call, or of referring to a variable. Also: one half of an association in a dictionary: the half you’re looking for when you do a lookup. If you say `dict['zog']=123`, then the value is `123`.

variable	A name for an object. After saying `x=6`, `x` is a name for the object usually known as `6`, and using `x` will usually do the same as using `6` would. `x` is called a “variable” because you can change what object it names.