Lua

Lua is a cross-platform, extension programming language that appeared 23 years ago in 1993 – publically in 1994. It was created to serve as a lightweight scripting language that could extend and customise the functionality of other programs. In other words, the Lua programming language was not the primary programming language of a program, but rather a side language, which helped to increase development efficiency.

Lua was developed in the Pontifical Catholic University by three people: Roberto Ierusalimschy *(Ee êh roo zah leems key)*, Luiz Henrique de Figueiredo, and Waldemar Celes. From 1977 until 1992, Brazil had a strict market reserve (aka trade barrier) that made it difficult for them, as members of their university’s Computer Graphics Technology Group (Tecgraf) to buy nor sell custom computer software. So, in response? They built their own software.  
  
Lua is a programming language very similar to other languages, and is relatively easy to pick up on. It has the dynamic scripting feel like JavaScript, where the variables are not assigned a type, but rather the values only - and also has a similar code syntax with C and Pascal, two other programming languages.

Lua can be considered as both an object-orientated language, as well as functional programming language – two different programming paradigms. In essence, object-orientated programming refers to grouping sets of data-specific code, and hence the data itself, into things called ‘objects’. Functional programming refers to the use, *and possibly the overuse…* of functions to complete the necessary task, using given inputs, processing the input, and outputting a result.  
  
Fun fact: the name of this programming language, Lua, is the Portuguese word for Moon. This programming language was built as a successor of a previous language, called SOL (Simple Object Language). As SOL meant Sun in Portuguese, the creators of Lua decided to name their programming language the opposite of Sol, as a joke.  
  
Fun fact #2: the programming language was awarded the 2011 Game Developer Magazine Front Line Award

-------------------------------------------------------------------------------------------------------------------------

Creating Lua Scripts

Like basically every source code file, Lua scripts are simply plain-text files, and hence can be written with any text editor. However, special editors, known as Integrated Development Environments (IDEs) have features that assist the developer, such as automatic code completion (like autocomplete when you type on the phone), syntax highlighting, error checking, and etcetera.  
  
Lua scripts can be identified if their file extension are “.lua”

Here on the screen is an example Lua script written with notepad variation. It demonstrates the programming syntax for Lua

-------------------------------------------------------------------------------------------------------------------------

Code Similarities

This slide here shows the same program also written in JavaScript and Python. (It’s abit small, but they are basically identical, with a few syntax differences here and then)

-------------------------------------------------------------------------------------------------------------------------

Lua Syntax

Lua’s basic code syntax can be seen here. Like other programming languages, Lua uses standard symbols and keywords as operators. At a basic level of Lua, it is also able to do mathematical calculations and comparisons. --explain code | mathematical operations, comparisons, LENGTH func, comment, keyword--

-------------------------------------------------------------------------------------------------------------------------

Lua Conventions

Programming conventions refer to the way how programs are written, as implied.  
In Lua:

* Whitespaces (aka space/tab indents) are insignificant – the program will function regardless
* Great for some, EOL semicolons are optional
* Braces {} are replaced with do and end.
* A number is a number (That is, there is no difference between integers and floats [decimal point no’s])
* Undefined/null values are labelled as nil

The majority of programming languages utilise 0-based arrays. That is, the positions of items in an array begin from the index 0. In other words, the first item in an array would be referred to as the 0th item or index.

However, in Lua arrays are 1-based that is, meaning that the first item in an array would be referred to as index 1.

To those relatively new to this programming, and the idea of 0 or 1 based indexes, Lua’s use of 1-based indexes will seem to make more sense than that of 0-bases indexes. However those who have experience in other languages may find the use of 1-based indexes in Lua annoying.

Lua also does not incorporate ternary operator (a?b:c) – aka no ‘one liners’, and is able to assign values to multiple variables at a time.

-------------------------------------------------------------------------------------------------------------------------

Variables

Variables are data items which store information. In Lua, also known as an identifier, a variable can take form as a global variable or a local variable (**or** a property – a variable in a table)  
A global variable is accessible at any point of the code.  
A local variable can only be accessed by the functions in the same nesting, and can be declared by appending the keyword local before the variable declaration. --explain code--

If a variable is referenced without being declared, it will be undefined, or nil

In terms of naming variables, they can be named anything, provided that they:

* Are not the same as Lua’s programming keywords
* Contain only alphanumeric and underscore characters (A-Z, a-z, 0-9, \_)
* Do not begin with a digit

--demonstrate valid and invalid variable names--

-------------------------------------------------------------------------------------------------------------------------

In context of software development, programmers need to be aware of how much memory their software is consuming. If their code is laden with too many unused variables, or has a memory bug, the application may freeze and crash, or even worse, crash the whole computer. To stop these memory leaks, the programmer must be aware of how much memory their application is using, and take necessary memory management precautions  
  
The Lua programming language has inbuilt automatic memory management, also known as Garbage Collection (GC). At regular intervals, the program will automatically free up memory by releasing variables that will no longer be used. Garbage Collection, however, does not affect global variables, nor properties in a table.

To delete variables, the value of a variable can simply be set to nil

-------------------------------------------------------------------------------------------------------------------------

Why Lua

So why Lua?

* As an extension programming language, Lua is inherently capable on working on practically anything that can interpret the host language.
* Compared to other programming languages, prototypes written in Lua can be easily modified and incorporated, shortening development time.
* Lua also features JIT, or Just In Time compilation. This is a dynamic compilation method, where the code can be compiled just before it runs, dropping the need to manually compile the application
* It also small and lightweight, yet powerful enough to work on a range of applications, from microcontrollers to *game engines*.
* Lua also uses relatively syntax, which could be thought of as pseudocode

-------------------------------------------------------------------------------------------------------------------------

History

Lua has undergone many updates, and is currently at version 5.3.2, which was released in November 2015. Each iteration brought new features, functions, bug fixes (and bugs…), more efficient code execution, and support to more and more devices.

-------------------------------------------------------------------------------------------------------------------------

Uses of Lua

As mentioned earlier, Lua is not intended as the primary programming language of an application (albeit possible). Instead, Lua is embedded in a program with another main language. It is commonly used in games for scripts that are triggered in a level.  
  
Well known software/games/whatever that use Lua include: the VLC player, TeamSpeak, Garry’s Mod, CryEngine, Adobe Lightroom, World of Warcraft, and many more. It also can be used to script the LEGO Mindstorms NXT unit.