# Tree Climber Documentation

## *Licence Information*


Version: 1.1
E-mail: marco.terzuoli@libero.it


## *Before you read on*



## *Requirements*

This program has been developed on Linux Mandrake 9.2 and 10.0 Community Edition and tested on these operating systems, plus some additional testing under Windows 2000 and Xp. Because the program is entirely written in Python, it should work on any sort of architecture, provided that you have Python correctly installed on the system. This is a default for most Linux distribution, whereas if you use Windows you have to download it from www.python.org. The Python version I have been used throughout the development process is 2.3, but different version should work just as well (unless maybe you have a very old version, in which case this is a good chance to get an upgrade).
On Linux you just have to expand the file treeclimber_1.1.tar.gz and give permission to run the file, which is accomplished using the following commands:
tar -xvzf treeclimber_1.1.tar.gz
chmod a+x treeclimber.py
On Windows you can expand the tar.gz archive by means of wome program such as Winzip or Winrar and what you get should be ready to be run. To run the program under Linux you just have to launch at like any other script or executable, which means type its name and hit return (provided the path it is installed in is in you PATH variable). Under Windows you have to run a command line shell, move to the directory where you decompressed the program, the run “python treeclimber.py”.
I am sorry not to give you any clue on how to decompress and run the program under other architectures such as Mac, but I don' have any way to access a system of that sort, so mine would only be guesswork.

## *How to create decision trees*

The first thing you have to do in order to use the program is create a file which describes a decision trees. This is a normal text file, with any extension you wish to use, which describes the topology of the trees, meaning that it tells the program how the node are connected to one another and what their values are. Each line of text consists of a letter describing what information is contained on the line, followed by one or mode parameters, which are separated by one or more spaces. Here follow the commands along with their sintax and axplanation:

| | |
|---|---|
| v <variable name> <value> | Creates a variable, assigning it a value. Variable definitions must be done before any other command, otherwise you will get an error saying that the variable was not initialized. Variables are particularly used when you want to reuse the same value multiple times in you tree file, thus avoiding a lot of retyping in case you need to change that value. |
| n <node name> <value> | Defines a leaf node, along with its value. Only leaf nodes need to be defined, while the others are automatically generated when you create the arcs that connect them to other nodes. The value parameter can be any sort of expression, made up of different funcions, constants and variables. See further on for a description of the functions ypu can use. |
| a <parent node> <son node> <probability> | Creates an arc. No probability check are performed in order to verify if the sum the probabilities of the arcs that come out of a single parent node is 1, so you should take care of it yourself. Further versions of the program are likely to implement this feature. Probability can be an expression of any sort. |
| d <node name> | Defines a decision nodes, whereas all the others are lottery nodes by default. The arcs that come out of a decision node need to have a probability associated to them, even though this value is not used by the program (this will be fixed soon). |

If you are not very familiar with decision trees, be informed that a decision node value is equal to the greatest value among its sons' while a lottery node value is equal to the mean of its sons' values, weighed with the arcs probabilities.
Once you have created you tree file, just run tree-climber, supplying the tree file as only parameter. As an answer, you will get all the stages of resolution, plus a list of the decisions that should be taken.

## *Functions available*

The functions that you can use to define probabilities and leaf node values are all those provided by the Python interpreter. Apart from standard mathematical functions such as +, -, *, / (standard division, but remember to use numbers with at least one decimal digit), // (integer division), % (modulus), abs (absolute value), max, min; you can also use the functions contained in the math library. For a full list run you python interpreter, then write:
import math
dir(math)
If you want to see how a function named func works, just write
help(math.func)