# CSCI3230 Fundamentals of Artificial Intelligence
# Neural Network Project – Directed Marketing
# Due date: 23:59:59 (GMT +08:00), 10 December, 2012

## 1. Project Specification

In the world of business, enterprises promote products and services using two main approaches: mass campaigns such as advertisements, targeting the general public or directed marketing such as telephone calls, targeting a specific set of contacts. Nowadays, positive responses to mass campaigns are typically low, less than 1%. Alternatively, by focusing on the targets assumed to be interested in that specific product or service, directed marketing is far more attractive than the other approaches in terms of cost and effectiveness. However, it is an art rather than a science to select a few special ones from the tremendous size of customer database.

In this project, given real-world data collected from a marketing campaign, you are helping a bank to develop a classifier which can predict if a client will subscribe a deposit. The classifier can be applied to select potential customers. After careful consideration, you choose to develop the classifier using *Neuron Network*. You have to design, train, validate, test and output your neural network in a specified format to accomplish the task.

Your neural network should learn its weights by using the *Back-Propagation* algorithm. You may choose one of the available programming languages, which are listed in the Appendix. However, you are not allowed to use any data mining or machine learning packages.

You are required to output your neural network in a format as described below and name the output file *best.nn* for submission; Also, you have to a submit a *preprocessor* program to convert the raw data to the inputs to neural network; a *trainer* program to train the neural network (*best.nn*) and *two shell scripts* to compile and run the preprocessor and the trainer program respectively.

## 2. Dataset

You are provided with the *raw-training.name* and *raw-training.data*. The raw data file contains 5906 records. Each record containing 17 attributes represents information about a customer.

| | **Attribute** | **Type** | **Description** |
|---|---|---|---|
| | *Personal Information* | | |
| 1 | ***age*** | Numeric | Age |
| 2 | ***job*** | Categorical | Type of job |
| 3 | ***marital*** | Categorical | Marital status |
| 4 | ***education*** | Categorical | Education level |
| 5 | ***default*** | Yes/No | Has credit (debit) card in default? |
| 6 | ***balance*** | Categorical | Average yearly balance, in euros |
| 7 | ***housing*** | Yes/No | Has housing loan? |
| 8 | ***loan*** | Yes/No | Has personal loan? |
| | *Attributes related to the* ***last contact*** *of the current marketing campaign* | | |
| 9 | ***contact*** | Categorical | Contact communication type |
| 10 | ***day*** | Numeric | Last contact day of the month |
| 11 | ***month*** | Categorical | Last contact month of year |
| 12 | ***duration*** | Numeric | Last contact duration, in seconds |
| | *Other attributes* | | |
| 13 | ***campaign*** | Numeric | No. of contacts performed during this campaign for this client |

| | | | |
|---|---|---|---|
| 14 | ***pdays*** | Numeric | No. of days that passed by after the client was last contacted from a previous campaign |
| 15 | ***previous*** | Numeric | No. of contacts performed before this campaign and for this client |
| 16 | ***poutcome*** | Categorical | Outcome of the previous marketing campaign |
| 17 | *y* | Yes/No | Outcome of the current marketing campaign: Has the client subscribed a term deposit? |

## 3. The Neural Network and the File Format of *best.nn*

As you are suggested to do data preprocessing, we do NOT limit the number of input neurons, the number of hidden layers and the number of neurons in each hidden layer. You can choose them as appropriate.

However, we have limited the number of output neurons to be **1**. The output variable ***y*** is represented by 1 neuron, where **1** means **Yes** and **0** means **No**. If its output is equal to or greater than 0.5, it is considered to be 1, otherwise 0.

To transform categorical data into numeric data, you can use 1-of-C encoding, which uses one bit to represent a category. For example, *contact* includes 3 type of data: “Unknown”, “Telephone” and “Cellular”. You can use [1 0 0], [0 1 0], [0 0 1] to represent them respectively.

The activation function used for a neuron is the sigmoid function, which is

$$g(t) = \frac{1}{1+e^{-t}}$$

The possible structure of neural network allowed is illustrated in figure 1.

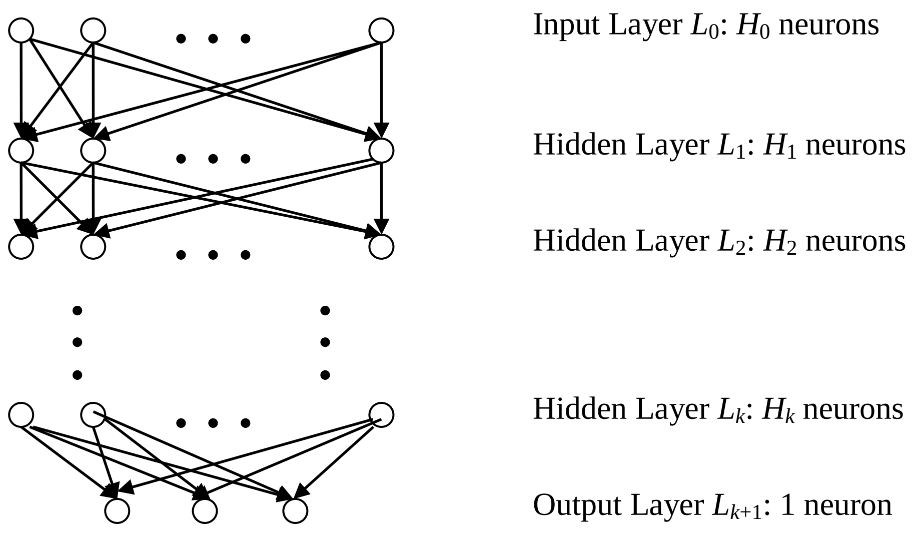
Input Layer $L_0$: $H_0$ neurons
Hidden Layer $L_1$: $H_1$ neurons
Hidden Layer $L_2$: $H_2$ neurons
Hidden Layer $L_k$: $H_k$ neurons
Output Layer $L_{k+1}$: 1 neuron

Figure 1

The first line of the output should be the classification accuracy (0 to 1.0), the second line should be the number of neurons in each layer from the input layer to the output layer, i.e. $H_0\, H_{1\ldots}\, H_k$ The rest store the weights, divided into $k$+1 section, as follows:

Each section stores the weights and bias terms between two successive layers *i* and *i*+1 (Input to layer 1, layer 1 to layer 2, …, layer *k* to Output). Each section starts with a line with 4 items, separated with space:

$$label_i \quad H_i \quad label_{i+1} \quad H_{i+1}$$

where $label_i$ is the label of layer *i* ('I' for input layer, 'H' for hidden layer, 'O' for output layer), $H_i$ is the number of neurons in layer *i*, $label_{i+1}$ is the label of layer *i*+1, $H_{i+1}$ is the number of neurons in layer *i*+1. Then each of the next $H_{I+1}$ lines should store the bias term and weights of the *r*-th neuron in layer *i*+1. Each line should look like:

$$w_{i,1} \quad w_{i,2} \quad \dots \quad w_{i,Hi} \quad w_{i,0}$$

where $w_{i,0}$ is the bias term,(which acts like a weight of an input of constant -1) the following $H_i$ numbers $w_{i,j}$ should be the weight from the *j*-th neuron in layer *i* to the *r*-th neuron in layer *i+1*.

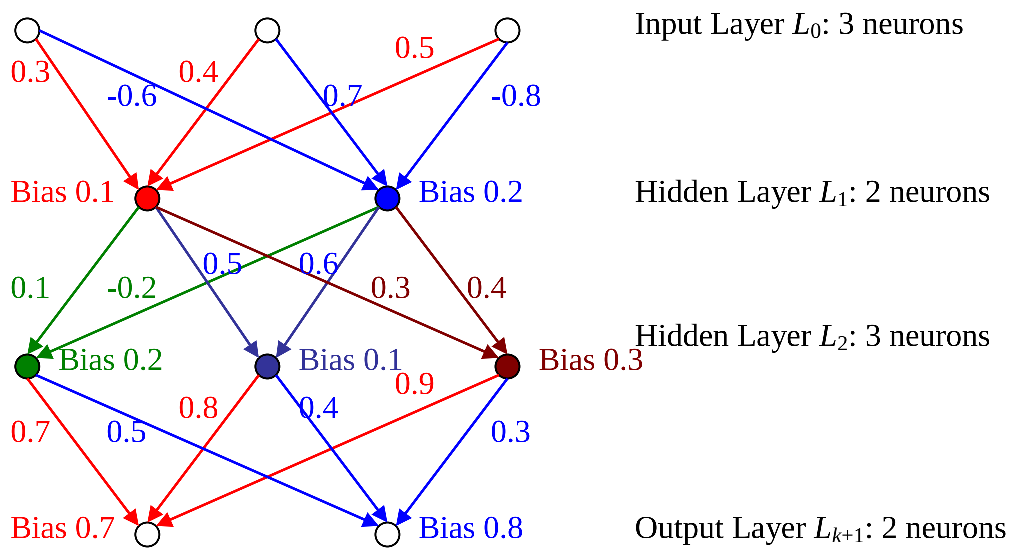
Input Layer $L_0$: 3 neurons
Hidden Layer $L_1$: 2 neurons
Hidden Layer $L_2$: 3 neurons
Output Layer $L_{k+1}$: 2 neurons
0.3
0.4
0.5
-0.6
0.7
-0.8
Bias 0.1
Bias 0.2
0.1
-0.2
0.5
0.6
0.3
0.4
Bias 0.2
Bias 0.1
Bias 0.3
0.7
0.5
0.8
0.4
0.9
0.3
Bias 0.7
Bias 0.8

Figure 2

For a complete example, for the neural network in figure 2 with classification accuracy of 0.9, the output file would look like:

```
0.9
3 2 3 2
I 3 H 2
0.3 0.4 0.5 0.1
-0.6 0.7 -0.8 0.2
H 2 H 3
0.1 -0.2 0.2
0.5 0.6 0.1
0.3 0.4 0.3
H 3 O 2
0.7 0.8 0.9 0.7
0.5 0.4 0.3 0.8
```

## 4. Preprocessing Program

In this project, you are given only raw data. You are advised to preprocess the raw data to help building a good neural network. But you **MUST** submit a preprocessor program which converts raw data without class labels **(class labels would appear as '?' when missing)** into inputs to the neural network. (With the same relative order i.e. do not shuffle the data records) The format is: for raw data with *N* records to be classified, there should be *N* line(s) in the output. Each of the lines corresponds to a record and should have $H_0$ numbers, to be used as the $H_0$ inputs to the neural network, where $H_0$ is the number of input neurons you have chosen. For example, if the raw data contains 2 records, and you choose to have 3 inputs neurons, the file may look like:

```
5 1 8
6 7 3
```

In order for your trainer program to train the neural network, an answer file containing only transformed class labels ("yes"->1, "no"->0) has to be generated by your preprocessor program. (If a question mark instead of a class label is given, please return a question mark correspondingly.) Each of the *N* lines is the expected output for the corresponding record. For the above example, the answer file may look like:

```
1
0
```

## 5. Usage and Format of Shell Scripts

In this project, you are required to submit 2 shell scripts to compile and run the preprocessor and trainer programs, named "*preprocessor.sh*" and "*trainer.sh*" respectively.

For *preprocessor.sh*:
The required execution command is:
`./preprocessor.sh raw-training.data dataset.dat answer.dat`
where `raw-training.dat` is the raw dataset that you can find in our course homepage;
`dataset.dat` is the preprocessed data, the inputs of the neural network;
`answer.dat` is the class label for the preprocessed data
Example (if your preprocessor is written in C language):

```
#!/bin/sh
gcc -o preprocess preprocess.c -Wall -lm
./preprocess $1 $2 $3
```

For *trainer.sh*:
The required execution command is:
`./trainer.sh dataset.dat answer.dat best.nn <max_iteration> <small_sample_mode>`
where `dataset.dat` is the preprocessed data generated by *preprocessor.sh* as input;
`answer.dat` is the class label for the preprocessed data;
`best.nn` is the neural network your program trained as output;
`<max_iteration>` is the maximum number of iterations for the learning process†;
`<small_sample_mode>` is the flag indicating the mode of training†

Example (if your trainer is written in Java language):

```
#!/bin/sh
javac trainer.java
java trainer $1 $2 $4 $5 > $3
```

† In order for the tutors to examine the correctness and performance of your backward propagation algorithm, two extra arguments are required to be accepted by your trainer program. The first one is `<max_iteration>`, the maximum number of iteration for the learning process, it is one of the early stopping criteria. The second one is `<small_sample_mode>`, the flag indicates the training mode, 1 for small sample mode and 0 for typical mode. During small sample mode, your trainer program should be able to handle a tiny dataset with about 500 observations only.

## 6. Grading Criteria

We will test your neural network on the training data set and a secret testing data set. We will use your preprocessing program to get the inputs to be fed to your neural network, and then we will simulate the neural network using our own program to calculate the classification accuracy.

The division of marks is:

| | |
|---|---|
| - Implementation of back-propagation | 50% |
| - Classification accuracy of Training dataset | 20% |
| - Classification accuracy of Testing dataset | 30% |

## 7. Assignment Submission

Zip the following with filename as "NN.zip":

- Source code of your preprocessor program
- Source code of your trainer program
- Your best neural network named "*best.nn*" (case-sensitive)
- A shell script to compile and run the preprocessor program
- A shell script to compile and run the trainer program
- A readme file that explain how to use your shell scripts and programs

You **MUST** submit the ZIP file to the submission system on our course homepage (within CUHK network), otherwise, we will **NOT** mark your assignment.

## 8. Important Points

You MUST STRICTLY follow these points:

- You MUST strictly follow the submission guidelines.
- Your trainer is only given 1 MINUTE of execution time.
- Late submission will NOT be entertained according to our submission system settings.
- Plagiarism will be SERIOUSLY punished.

## 9. Late Submission

According to the course homepage, late submission will lead to marks deduction.

| No. of Days Late | Marks Deduction |
|---|---|
| 1 | 10% |
| 2 | 30% |
| 3 | 60% |
| 4 or above | 100% |

## Appendix

| **Language** | **Version** | **Usage** |
|---|---|---|
| C | gcc version 4.3.3 (Ubuntu 4.3.3-5ubuntu4) | `Usage: gcc [options] file...`<br>`Options:`<br>`  -pass-exit-codes         Exit with highest error code from a phase`<br>`  --help                   Display this information`<br>`  --target-help            Display target specific command line options`<br>`  --help={target\|optimizers\|warnings\|undocumented\|params}[,{[^]joined\|[^]separate}]`<br>`                           Display specific types of command line options`<br>`  (Use '-v --help' to display command line options of sub-processes)`<br>`  -dumpspecs               Display all of the built in spec strings`<br>`  -dumpversion             Display the version of the compiler`<br>`  -dumpmachine             Display the compiler's target processor`<br>`  -print-search-dirs       Display the directories in the compiler's search path`<br>`  -print-libgcc-file-name  Display the name of the compiler's companion library`<br>`  -print-file-name=<lib>   Display the full path to library <lib>`<br>`  -print-prog-name=<prog>  Display the full path to compiler component <prog>`<br>`  -print-multi-directory   Display the root directory for versions of libgcc`<br>`  -print-multi-lib         Display the mapping between command line options and`<br>`                           multiple library search directories`<br>`  -print-multi-os-directory Display the relative path to OS libraries`<br>`  -print-sysroot-headers-suffix Display the sysroot suffix used to find headers`<br>`  -Wa,<options>            Pass comma-separated <options> on to the assembler`<br>`  -Wp,<options>            Pass comma-separated <options> on to the preprocessor`<br>`  -Wl,<options>            Pass comma-separated <options> on to the linker`<br>`  -Xassembler <arg>        Pass <arg> on to the assembler`<br>`  -Xpreprocessor <arg>     Pass <arg> on to the preprocessor`<br>`  -Xlinker <arg>           Pass <arg> on to the linker`<br>`  -combine                 Pass multiple source files to compiler at once`<br>`  -save-temps              Do not delete intermediate files`<br>`  -pipe                    Use pipes rather than intermediate files`<br>`  -time                    Time the execution of each subprocess`<br>`  -specs=<file>            Override built-in specs with the contents of <file>`<br>`  -std=<standard>          Assume that the input sources are for <standard>`<br>`  --sysroot=<directory>    Use <directory> as the root directory for headers`<br>`                           and libraries`<br>`  -B <directory>           Add <directory> to the compiler's search paths`<br>`  -b <machine>             Run gcc for target <machine>, if installed`<br>`  -V <version>             Run gcc version number <version>, if installed`<br>`  -v                       Display the programs invoked by the compiler`<br>`  -###                     Like -v but options quoted and commands not executed`<br>`  -E                       Preprocess only; do not compile, assemble or link`<br>`  -S                       Compile only; do not assemble or link`<br>`  -c                       Compile and assemble, but do not link` |

```
  -o <file>                Place the output into <file>
  -x <language>            Specify the language of the following input files
                           Permissible languages include: c c++ assembler none
                           'none' means revert to the default behavior of
                           guessing the language based on the file's extension

Options starting with -g, -f, -m, -O, -W, or --param are automatically
 passed on to the various sub-processes invoked by gcc.  In order to pass
 other options on to these processes the -W<letter> options must be used.

For bug reporting instructions, please see:
<file:///usr/share/doc/gcc-4.3/README.Bugs>.
```

| | |
|---|---|
| C++ | gcc version 4.3.3 (Ubuntu 4.3.3-5ubuntu4) |

```
Usage: g++ [options] file...
Options:
  -pass-exit-codes         Exit with highest error code from a phase
  --help                   Display this information
  --target-help            Display target specific command line options
  --help={target|optimizers|warnings|undocumented|params}[,{[^]joined|[^]separate}]
                           Display specific types of command line options
  (Use '-v --help' to display command line options of sub-processes)
  -dumpspecs               Display all of the built in spec strings
  -dumpversion             Display the version of the compiler
  -dumpmachine             Display the compiler's target processor
  -print-search-dirs       Display the directories in the compiler's search path
  -print-libgcc-file-name  Display the name of the compiler's companion library
  -print-file-name=<lib>   Display the full path to library <lib>
  -print-prog-name=<prog>  Display the full path to compiler component <prog>
  -print-multi-directory   Display the root directory for versions of libgcc
  -print-multi-lib         Display the mapping between command line options and
                           multiple library search directories
  -print-multi-os-directory Display the relative path to OS libraries
  -print-sysroot-headers-suffix Display the sysroot suffix used to find headers
  -Wa,<options>            Pass comma-separated <options> on to the assembler
  -Wp,<options>            Pass comma-separated <options> on to the preprocessor
  -Wl,<options>            Pass comma-separated <options> on to the linker
  -Xassembler <arg>        Pass <arg> on to the assembler
  -Xpreprocessor <arg>     Pass <arg> on to the preprocessor
  -Xlinker <arg>           Pass <arg> on to the linker
  -combine                 Pass multiple source files to compiler at once
  -save-temps              Do not delete intermediate files
  -pipe                    Use pipes rather than intermediate files
  -time                    Time the execution of each subprocess
  -specs=<file>            Override built-in specs with the contents of <file>
  -std=<standard>          Assume that the input sources are for <standard>
  --sysroot=<directory>    Use <directory> as the root directory for headers
                           and libraries
```

| | | |
|---|---|---|
| | | `  -B <directory>           Add <directory> to the compiler's search paths`<br>`  -b <machine>             Run gcc for target <machine>, if installed`<br>`  -V <version>             Run gcc version number <version>, if installed`<br>`  -v                       Display the programs invoked by the compiler`<br>`  -###                     Like -v but options quoted and commands not executed`<br>`  -E                       Preprocess only; do not compile, assemble or link`<br>`  -S                       Compile only; do not assemble or link`<br>`  -c                       Compile and assemble, but do not link`<br>`  -o <file>                Place the output into <file>`<br>`  -x <language>            Specify the language of the following input files`<br>`                           Permissible languages include: c c++ assembler none`<br>`                           'none' means revert to the default behavior of`<br>`                           guessing the language based on the file's extension`<br><br>`Options starting with -g, -f, -m, -O, -W, or --param are automatically`<br>` passed on to the various sub-processes invoked by g++.  In order to pass`<br>` other options on to these processes the -W<letter> options must be used.`<br><br>`For bug reporting instructions, please see:`<br>`<file:///usr/share/doc/gcc-4.3/README.Bugs>.` |
| Java | java version "1.6.0_0" OpenJDK Runtime Environment (IcedTea6 1.4.1) (6b14-1.4.1-0ubuntu12) OpenJDK Client VM (build 14.0-b08, mixed mode, sharing) | `Usage: javac <options> <source files>`<br>`where possible options include:`<br>`  -g                         Generate all debugging info`<br>`  -g:none                    Generate no debugging info`<br>`  -g:{lines,vars,source}     Generate only some debugging info`<br>`  -nowarn                    Generate no warnings`<br>`  -verbose                   Output messages about what the compiler is doing`<br>`  -deprecation               Output source locations where deprecated APIs are used`<br>`  -classpath <path>          Specify where to find user class files and annotation processors`<br>`  -cp <path>                 Specify where to find user class files and annotation processors`<br>`  -sourcepath <path>         Specify where to find input source files`<br>`  -bootclasspath <path>      Override location of bootstrap class files`<br>`  -extdirs <dirs>            Override location of installed extensions`<br>`  -endorseddirs <dirs>       Override location of endorsed standards path`<br>`  -proc:{none,only}          Control whether annotation processing and/or compilation is done.`<br>`  -processor <class1>[,<class2>,<class3>...]Names of the annotation processors to run; bypasses`<br>`default discovery process`<br>`  -processorpath <path>      Specify where to find annotation processors`<br>`  -d <directory>             Specify where to place generated class files`<br>`  -s <directory>             Specify where to place generated source files`<br>`  -implicit:{none,class}     Specify whether or not to generate class files for implicitly`<br>`referenced files`<br>`  -encoding <encoding>       Specify character encoding used by source files`<br>`  -source <release>          Provide source compatibility with specified release`<br>`  -target <release>          Generate class files for specific VM version`<br>`  -version                   Version information` |

| | | |
|---|---|---|
| | | `  -help                        Print a synopsis of standard options`<br>`  -Akey[=value]                Options to pass to annotation processors`<br>`  -X                           Print a synopsis of nonstandard options`<br>`  -J<flag>                     Pass <flag> directly to the runtime system` |
| swi-Prolog | SWI-Prolog (Multi-threaded, 32 bits, Version 5.6.59) | `swipl: Usage:`<br>`    1) swipl --help     Display this message (also -h)`<br>`    2) swipl --version  Display version information (also -v)`<br>`    3) swipl --arch     Display architecture`<br>`    4) swipl --dump-runtime-variables[=format]`<br>`                     Dump link info in sh(1) format`<br>`    5) swipl [options]`<br>`    6) swipl [options] [-o output] -c file ...`<br>`    7) swipl [options] [-o output] -b bootfile -c file ...`<br>`Options:`<br>`    -x state         Start from state (must be first)`<br>`    -[LGTA]size[KMG] Specify {Local,Global,Trail,Argument} limits`<br>`    -t toplevel      Toplevel goal`<br>`    -g goal          Initialisation goal`<br>`    -f file          User initialisation file`<br>`    -F file          System initialisation file`<br>`    -s file          Script source file`<br>`    [+/-]tty         Allow tty control`<br>`    -O               Optimised compilation`<br>`    --nosignals      Do not modify any signal handling`<br>`    --nodebug        Omit generation of debug info`<br>`    --quiet          Quiet operation (also -q)`<br>`    --home=DIR       Use DIR as SWI-Prolog home` |
| CLisp | GNU CLISP 2.44.1 (2008-02-23) (built 3436675404) | `GNU CLISP (http://clisp.cons.org/) is an ANSI Common Lisp.`<br>`Usage:  clisp [options] [lispfile [argument ...]]`<br>` When 'lispfile' is given, it is loaded and '*ARGS*' is set`<br>` to the list of argument strings. Otherwise, an interactive`<br>` read-eval-print loop is entered.`<br>`Informative output:`<br>` -h, --help    - print this help and exit`<br>` --version     - print the version information`<br>` --license     - print the licensing information`<br>` -help-image   - print image-specific help and exit`<br>`Memory image selection:`<br>` -B lisplibdir - set the installation directory`<br>` -K linkingset - use this executable and memory image`<br>` -M memfile    - use this memory image`<br>` -m size       - memory size (size = xxxxxxxB or xxxxKB or xMB)`<br>`Internationalization:`<br>` -L language   - set user language`<br>` -N nlsdir     - NLS catalog directory`<br>` -Edomain encoding - set encoding` |

| | | |
|---|---|---|
| | | `Interoperability:` |
| | | ` -q, --quiet, --silent, -v, --verbose - verbosity level:` |
| | | `     affects banner, *LOAD-VERBOSE*/*COMPILE-VERBOSE*,` |
| | | `     and *LOAD-PRINT*/*COMPILE-PRINT*` |
| | | ` -w            - wait for a keypress after program termination` |
| | | ` -I            - be ILISP-friendly` |
| | | `Startup actions:` |
| | | ` -ansi         - more ANSI CL compliance` |
| | | ` -traditional  - traditional (undoes -ansi)` |
| | | ` -modern       - start in a case-sensitive lowercase-preferring package` |
| | | ` -p package    - start in the package` |
| | | ` -C            - set *LOAD-COMPILING* to T` |
| | | ` -norc         - do not load the user ~/.clisprc file` |
| | | ` -i file       - load initfile (can be repeated)` |
| | | `Actions:` |
| | | ` -c [-l] lispfile [-o outputfile] - compile lispfile` |
| | | ` -x expressions - execute the expressions, then exit` |
| | | ` Depending on the image, positional arguments can mean:` |
| | | `   lispscript [argument ...] - load script, then exit` |
| | | `   [argument ...]            - run the init-function` |
| | | `  arguments are placed in EXT:*ARGS* as strings.` |
| | | `These actions put CLISP into a batch mode, which is overridden by` |
| | | ` -on-error action - action can be one of debug, exit, abort, appease` |
| | | ` -repl            - enter the interactive read-eval-print loop when done` |
| | | `Default action is an interactive read-eval-print loop.` |
| Python | Python 2.6.2 | `usage: python [option] ... [-c cmd \| -m mod \| file \| -] [arg] ...` |
| | (release26-maint, | `Options and arguments (and corresponding environment variables):` |
| | Apr 19 2009, | `-B     : don't write .py[co] files on import; also PYTHONDONTWRITEBYTECODE=x` |
| | 01:56:41) | `-c cmd : program passed in as string (terminates option list)` |
| | | `-d     : debug output from parser; also PYTHONDEBUG=x` |
| | | `-E     : ignore PYTHON* environment variables (such as PYTHONPATH)` |
| | | `-h     : print this help message and exit (also --help)` |
| | | `-i     : inspect interactively after running script; forces a prompt even` |
| | | `         if stdin does not appear to be a terminal; also PYTHONINSPECT=x` |
| | | `-m mod : run library module as a script (terminates option list)` |
| | | `-O     : optimize generated bytecode slightly; also PYTHONOPTIMIZE=x` |
| | | `-OO    : remove doc-strings in addition to the -O optimizations` |
| | | `-Q arg : division options: -Qold (default), -Qwarn, -Qwarnall, -Qnew` |
| | | `-s     : don't add user site directory to sys.path; also PYTHONNOUSERSITE` |
| | | `-S     : don't imply 'import site' on initialization` |
| | | `-t     : issue warnings about inconsistent tab usage (-tt: issue errors)` |
| | | `-u     : unbuffered binary stdout and stderr; also PYTHONUNBUFFERED=x` |
| | | `         see man page for details on internal buffering relating to '-u'` |
| | | `-v     : verbose (trace import statements); also PYTHONVERBOSE=x` |
| | | `         can be supplied multiple times to increase verbosity` |
| | | `-V     : print the Python version number and exit (also --version)` |

| | | |
|---|---|---|
| | | -W arg : warning control; arg is action:message:category:module:lineno<br>-x     : skip first line of source, allowing use of non-Unix forms of #!cmd<br>-3     : warn about Python 3.x incompatibilities that 2to3 cannot trivially fix<br>file   : program read from script file<br>-      : program read from stdin (default; interactive mode if a tty)<br>arg ...: arguments passed to program in sys.argv[1:]<br><br>Other environment variables:<br>PYTHONSTARTUP: file executed on interactive startup (no default)<br>PYTHONPATH   : ':'-separated list of directories prefixed to the<br>               default module search path.  The result is sys.path.<br>PYTHONHOME   : alternate \<prefix\> directory (or \<prefix\>:\<exec_prefix\>).<br>               The default module search path uses \<prefix\>/pythonX.X.<br>PYTHONCASEOK : ignore case in 'import' statements (Windows).<br>PYTHONIOENCODING: Encoding[:errors] used for stdin/stdout/stderr. |
| Ruby | ruby 1.8.7 (2008-08-11 patchlevel 72) [i486-linux] | Usage: ruby [switches] [--] [programfile] [arguments]<br>  -0[octal]       specify record separator (\0, if no argument)<br>  -a              autosplit mode with -n or -p (splits $_ into $F)<br>  -c              check syntax only<br>  -Cdirectory     cd to directory, before executing your script<br>  -d              set debugging flags (set $DEBUG to true)<br>  -e 'command'    one line of script. Several -e's allowed. Omit [programfile]<br>  -Fpattern       split() pattern for autosplit (-a)<br>  -i[extension]   edit ARGV files in place (make backup if extension supplied)<br>  -Idirectory     specify $LOAD_PATH directory (may be used more than once)<br>  -Kkcode         specifies KANJI (Japanese) code-set<br>  -l              enable line ending processing<br>  -n              assume 'while gets(); ... end' loop around your script<br>  -p              assume loop like -n but print line also like sed<br>  -rlibrary       require the library, before executing your script<br>  -s              enable some switch parsing for switches after script name<br>  -S              look for the script using PATH environment variable<br>  -T[level]       turn on tainting checks<br>  -v              print version number, then turn on verbose mode<br>  -w              turn warnings on for your script<br>  -W[level]       set warning level; 0=silence, 1=medium, 2=verbose (default)<br>  -x[directory]   strip off text before #!ruby line and perhaps cd to directory<br>  --copyright     print the copyright<br>  --version       print the version |
| Perl | perl, v5.10.0 built for i486-linux-gnu-thread-multi | Usage: perl [switches] [--] [programfile] [arguments]<br>  -0[octal]         specify record separator (\0, if no argument)<br>  -a                autosplit mode with -n or -p (splits $_ into @F)<br>  -C[number/list]   enables the listed Unicode features<br>  -c                check syntax only (runs BEGIN and CHECK blocks)<br>  -d[:debugger]     run program under debugger<br>  -D[number/list]   set debugging flags (argument is a bit mask or alphabets) |

```
-e program        one line of program (several -e's allowed, omit programfile)
-E program        like -e, but enables all optional features
-f                don't do $sitelib/sitecustomize.pl at startup
-F/pattern/       split() pattern for -a switch (//'s are optional)
-i[extension]     edit <> files in place (makes backup if extension supplied)
-Idirectory       specify @INC/#include directory (several -I's allowed)
-l[octal]         enable line ending processing, specifies line terminator
-[mM][-]module    execute "use/no module..." before executing program
-n                assume "while (<>) { ... }" loop around program
-p                assume loop like -n but print line also, like sed
-P                run program through C preprocessor before compilation
-s                enable rudimentary parsing for switches after programfile
-S                look for programfile using PATH environment variable
-t                enable tainting warnings
-T                enable tainting checks
-u                dump core after parsing program
-U                allow unsafe operations
-v                print version, subversion (includes VERY IMPORTANT perl info)
-V[:variable]     print configuration summary (or a single Config.pm variable)
-w                enable many useful warnings (RECOMMENDED)
-W                enable all warnings
-x[directory]     strip off text before #!perl line and perhaps cd to directory
-X                disable all warnings
```