Source Code Example

Example of a machine readable program

In the following we are displaying a program part on cueing from an inventory list, retrieving the individual stock objects from an inventory list, checking their ability to the amortisation period and delivering the yearly amortisation value (if larger than zero) to the list of the yearly amortisation.

```
~
H%‰ÌVMo_E_Õzg–]‘¶ó_Æ †X%‰C
àd_DdƒŒ¥ ü{^õìšX¾p_eGÚ_ÕtW½ªzõ
z®_Óoä¸Š—Ä!5_′  FÜ[¤‡ >%¥?   q?
Òïîì ›_" ýzí®œ-…_®VÕç”+ Ÿ·ßÁß_~ŸÁ
åûËÕx 4[Ž³ ããai_
îM«‡ùøÁìd_d1  |²ZŽO†n__õà«_÷®Ø§É
_v_  Gýí  À_  n"¯Ÿö—ËÕ€p} ÙÇÇÃ|…¬–
~
```

Example of a running program source

The same program part is displayed in the source language „LISP“. A third party has to read up and search intensively to find the part, which has to be changed.

```
~
(defun afa-liniar (obj) (value) (time))
(setq obj (car obj#)) (declare (integer
(setq value (lamda ( ) x)))) (set-pprint-
dispatch `(cons ( obj value)) #`(lamda
(obj list) ( if (and (consp (cdr list)) ( null
cddr list))) (funcall (afa `~obj) time
(cadr list)) (pprint-fill time list)))))
(set-pprint-dispatch `(cons (inventar
value)
(pprint-dispatch `(let) nil))
~
```

Example of „best practice“ programming

And now the same source code commented and structured.

```
~
;;; Module afa-line for the determination of the yearly linear
;;; amortisation. Used variables:
;;; obj (amortisation object from inventory list objects)
;;; time (legally permitted amortisation period)
;;; value (amortisation value – for delivery to list)
;;; Changes for amortisation period have to up-dated in the module
law-time
(defun afa-liniar (obj) (value) (time))
(setq obj (car obj#))
(declare (integer (setq value (lamda ( ) x))))
(set-pprint-dispatch `(cons ( obj value))
#`(lamda (obj list) ( if (and (consp (cdr list))
( null cddr list)))
(funcall (afa `~obj) time(cadr list)) (pprint-fill time list)))))
(set-pprint-dispatch `(cons (inventar value))
(pprint-dispatch `(let) nil))
;;; The value is ignored if the amortisation value is 0 (nil)
;;; otherwise it is delivered to and processed by the module inventar
~
```