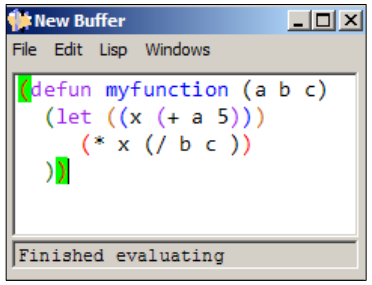
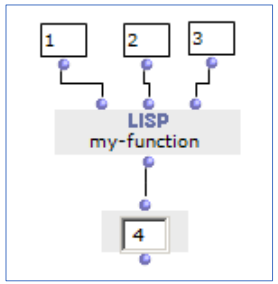
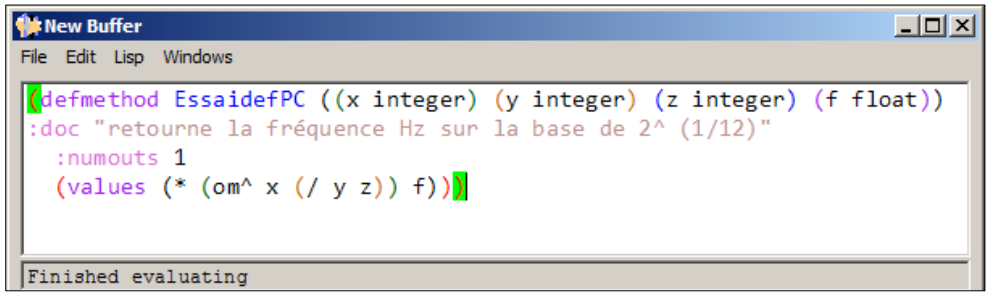
Fonctions Lisp avec PC sous windows 7 et 10

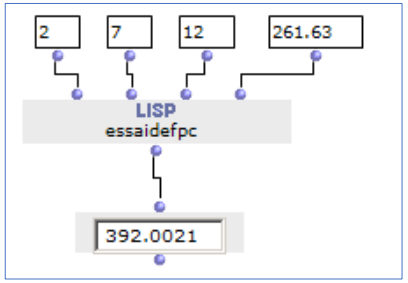


OM > myfunction

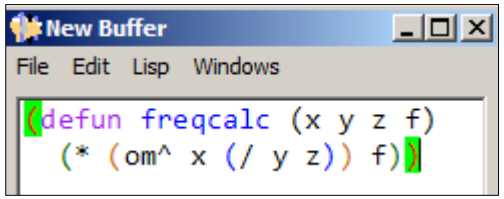


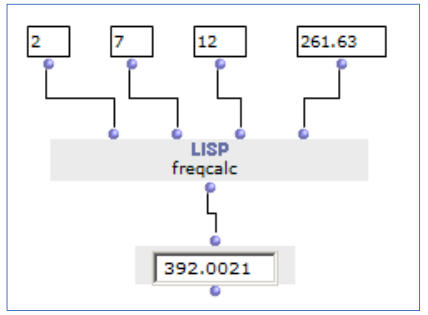

```
(defun myfunction (a b c)
 (let ((x (+ a 5)))
  (* x (/ b c ))
 ))
```



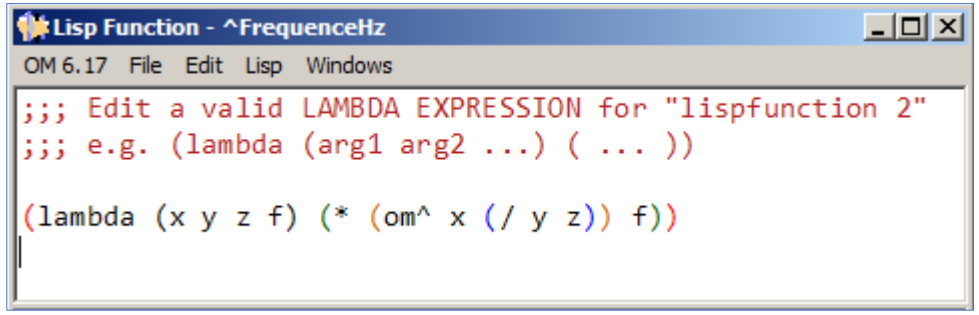


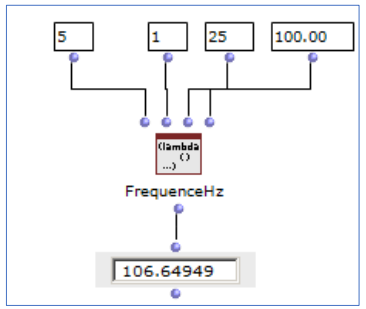

```
(defmethod EssaidefPC ((x integer) (y integer) (z integer)
(f float))
:doc "retourne la fréquence Hz sur la base de 2^ (1/12)"
 :numouts 1
 (values (* (om^ x (/ y z)) f)))
```






```
(defun freqcalc (x y z f)
 (* (om^ x (/ y z)) f))
```

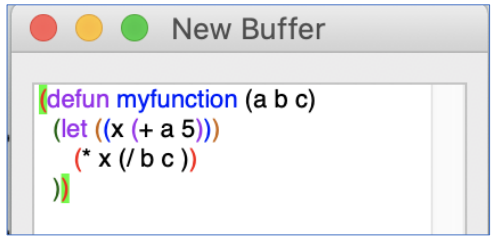





```
(lambda (x y z) (om^ x (/ y z)))   => indice
```

```
(lambda (x y z f) (* (om^ x (/ y z)) f))   => fréquence Hz
```

Fonctions Lisp Mac (Imac OS Mojave, 10.14.6, i7, 32 Go de Ram

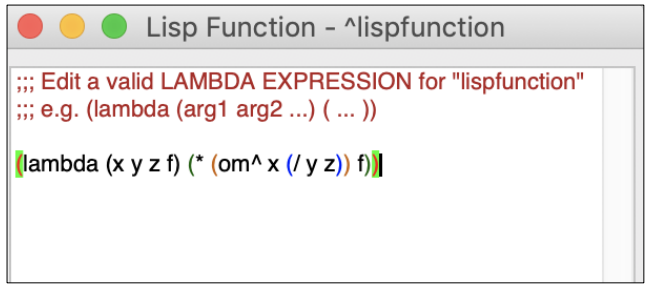


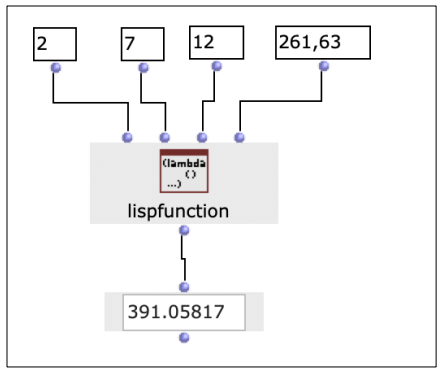
OM > myfunction

myfunction

OM > myfunction
OM > "function myfunction does not exist!"

Example from that proposed in the *OM6 manual*. This example does not pose any problem in the windows version OM 6.17 whereas in my Mac the function is not recognized just like for the examples of the first page made with the PC.

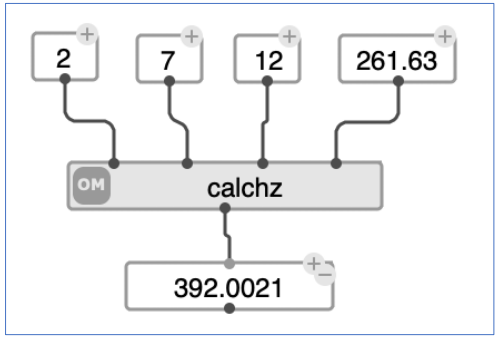




(lambda (x y z f)
(* (om^ x (/ y z)) f))

However, no problem to create Lisp functions with "*lambda expression*".

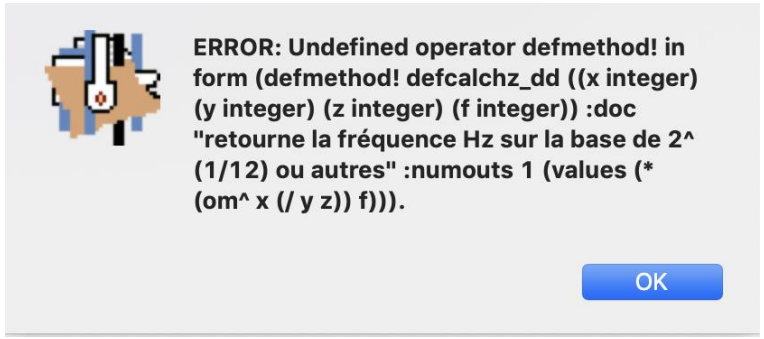
Note that the recognition of Lisp functions also does not pose any problem with *OM-Sharp*.



(lambda (x y z f)
(* (om^ x (/ y z)) f))

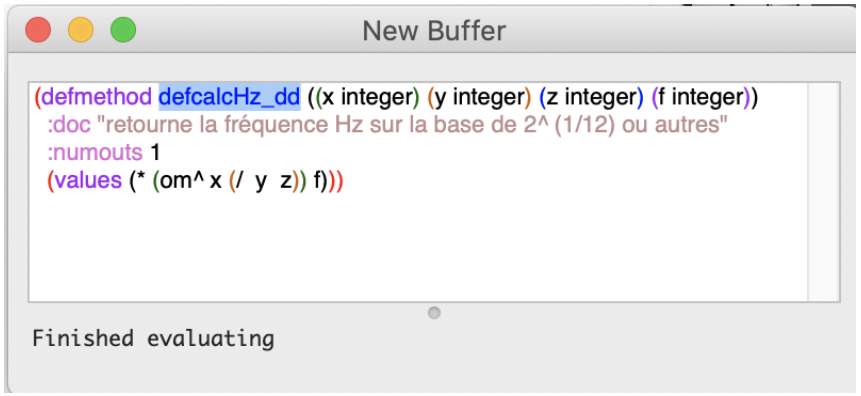
Exemples avec les messages d'erreur

```
(defmethod! defcalcHz_dd ((x integer) (y integer) (z integer) (f integer))
  :doc "retourne la fréquence Hz sur la base de 2^ (1/12) ou autres"
  :numouts 1
  (values (* (om^ x (/  y  z)) f)))
```



OM > "ERROR: Undefined operator defmethod! in form (defmethod! defcalchz_dd ((x integer) (y integer) (z integer) (f integer)) :doc \"retourne la fréquence Hz sur la base de 2^ (1/12) ou autres\" :numouts 1 (values (* (om^ x (/ y z)) f))).

Call to om-lisp::om-error-handler
Call to invoke-debugger
Call to errorCall to eval
Call to editor::editor-eval
Call to (subfunction 3 (subfunction 1 editor::region-lisp-eval))
Call to (subfunction 1 editor::region-lisp-eval)
Call to editor::with-compilation-environment-at-point-fn
Call to editor::region-lisp-evalCall to editor::background-region-eval
Call to capi::funcall-background-job-aux
Call to capi::bind-standard-streams-and-execute
Call to mp::background-execute-loop
Call to mp::process-sg-function"



Suppression du « ! » de defmethod

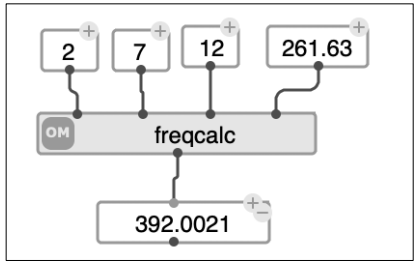
OM > #<standard-method defcalchz_dd (integer integer integer integer) 4020051553>
OM > "function defcalcHz_dd does not exist!"

Exemples avec OM 6.17 et OM Sharp

```
(defun freqcalc (x y z f)
  (* (om^ x (/ y z)) f))
```

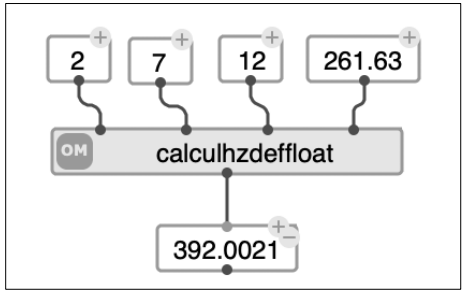
```
(defun freqcalc (x y z f)
  (* (om^ x (/ y z)) f))
```

OM > freqcalc
OM > "function freqcalc does not exist!"



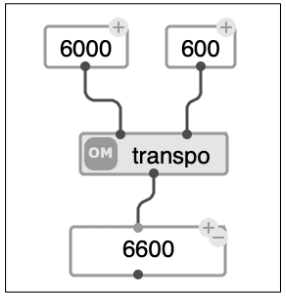

```
(defmethod! calculHzdeffloat ((x integer) (y integer) (z integer) (f float))
:doc "retourne la fréquence Hz sur la base de 2^ (1/12)"   :numouts 1
(values (* (om^ x (/ y z)) f)))
```

OM > calculhzdeffloat
OM > "function calculhzdeffloat does not exist!"



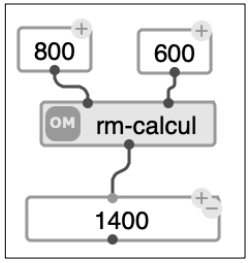

```
(defmethod! transpo ((x integer) (y integer))
:doc "crée une nouvelle note transposée"
:numouts 1
 (values (+ x y)))
```

OM > transpo
OM > "function transpo does not exist!"




```
(defmethod RM-calcul ((a integer) (b integer))
  :doc "Return 2 values : (+ ab ) (- a b)"
  :numouts 2
  (values (+ a b) (- a b)))
```

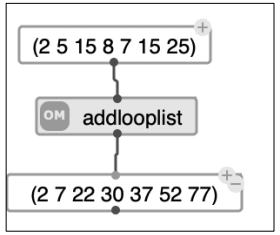
OM > #<standard-method rm-calcul (integer integer) 402000928B>
OM > "function RM-Calcul does not exist!"



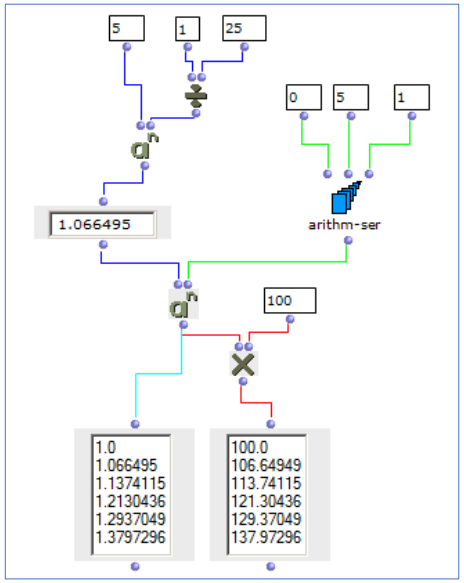
Remarque : seul le résultat de l'addition est affiché (800+600). Le *:numouts 2* n'a pas été pris en compte.

```
(defun addlooplist (list)
   (loop :for x :in list :sum x :into y :collect y))
```

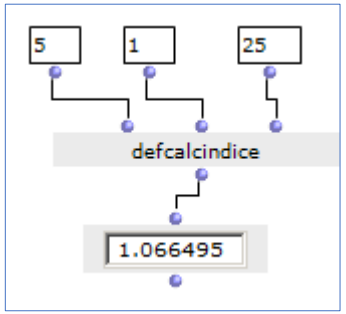
OM > addlooplist
OM > "function addlooplist does not exist!"

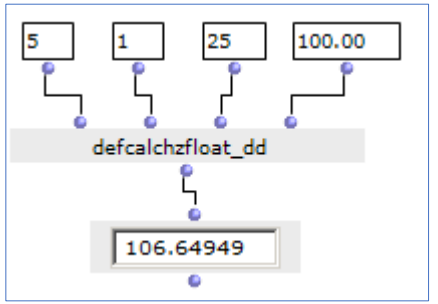


Essai pour créer une table des Fréquences Hz



Calcul des indices et Hz avec OM.

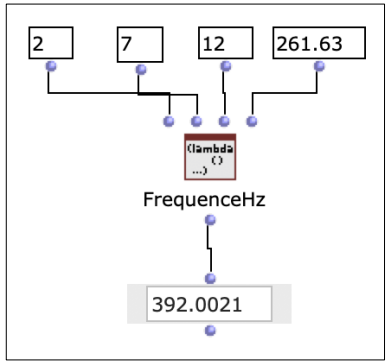




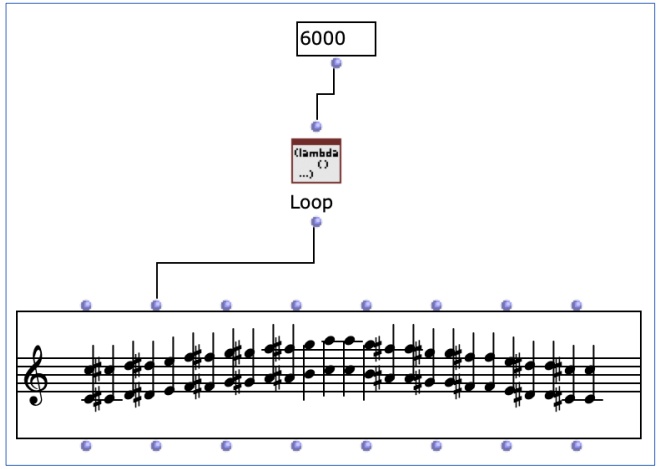
Calcul d'un indice et de la fréquence Hz (Study 2).

```
(defmethod! defcalcindice ((x integer) (y integer) (z integer))
  :doc "retourne la fréquence Hz sur la base de 2^ (1/12) ou autres"
  :numouts 1
  (values (om^ x (/  y  z))))
```

```
(defmethod! defcalcHz_dd ((x integer) (y integer) (z integer) (f integer))
  :doc "retourne la fréquence Hz sur la base de 2^ (1/12) ou autres"
  :numouts 1
  (values (* (om^ x (/  y  z)) f)))
```



How to calculate a table displaying *indices* and *Hz frequencies* with ***defmethod***, ***defun*** or with a ***lisp function lambda***?
What is the syntax for adding the second "*exponentiation*" function and the *arithm-ser* (or *loop*?)?




```
;;; Edit a valid LAMBDA EXPRESSION for "Loop"
;;; e.g. (lambda (arg1 arg2 ...) ( ... ))

(lambda (val)
  (let (Z)
    (setq Z (loop for i from 0 to 12 collect
                  (list (+ val (* i 100)) (+ val (* i 100) 1200))))
    (append Z (reverse Z))))
```

```
(lambda (val)    (let (Z)
(setq Z (loop for i from 0 to 12 collect
(list (+ val (* i 100)) (+ val (* i 100)
1200))))    (append Z (reverse Z))))
```

```
(setf Studie2Indice (mapcar (lambda(x y)
                      (expt x y))
                    (gen-repeat
                     5
                     (expt 5 1/25))
                    (make-scale 0 10)))

;;; => (1.0 1.066495 1.1374115 1.2130436 1.2937049)

(setf Studie2Freq (mapcar (lambda(x)
   (* 100.00 x))
        (mapcar (lambda(x y)
                      (expt x y))
                    (gen-repeat
                     5
                     (expt 5 1/25))
                    (make-scale 0 10))))

;;; => (100.0 106.64949 113.74115 121.30436 129.37048)

(mapcar #'cons Studie2Indice Studie2Freq)

;;; (1.0 . 100.0)
;;; (1.066495 . 106.64949)
;;; (1.1374115 . 113.74115)
;;; (1.2130436 . 121.30436)
;;; (1.2937049 . 129.37048)
```

Calculation of indices and Hz with Opusmodus. Note: the separate calculation of the indices and frequencies allows the "*cons*" function to be used for a joint presentation, otherwise "*Study2Freq*" gives the 2 separate results.