```
#! nevermind
? Require "OPToolkit" from "optk.proto"
Program "Operatoren"
    ? Module-Deklaration
    &&mod "Name"
    ? Parser
    #! path
    ? Type-Convertion
    num: Zahl = "3.14" !num
    ? And-Toolkit
    &cos()
    &sin()
    &tan()
    &cot()
    &arccos()
    &arcsin()
    &arctan()
    &arccot()
    &pow()
    &pi() = 3.14...
    &e() = 2.71...
    &hex()
    &bin()
    &num()
    &rad()
    &deg()
    ? Mathematical Operation
    10 + 4  = 14 (plus)
    10 - 4  =  6 (minus)
    10 * 4  = 40 (times)
    10 / 4  =  2 (integer division)
    10 _ 4  =  2.5 (float division)
    10 % 4  =  2 (modulo)
     2 ^ 3  = 16 (power)
     3 $ 16 =  2 (root)
     a++    = a = a + 1
     a--    = a = a - 1
     a$$    = a = 2 $ a
     a__    = a = 1 _ a
    ? Identifiers
    <a>    = Variable 'a'
    <~a>   = Variable 'a', private
    <a()>  = Function 'a'
    <~a()> = Function 'a', private
    ? Equality
    (object|string|num|list|path|bin|heX): name = value
    ? Boolean
    smaller, << smaller
    greater, >> greater
    equal, ==   equal
    diff, !!    different
    !  not
    () condition-limiters
    :  TypeOf (var : type)
    ? Keywords

    ? Require "Name" from "Path"
    ? Program "Name"
    ? EndProgram
    ? If (cond)
    ? EndIf

    ? Lists
    []          Empty List
    [a,b,c,...] List containing a,b,c,...
    [] + a      Add a to list
```

```
    [] - a      Remove a from list
    [] \ a      List contains a
    [](a...b)   List Items from a to b
    [a...b...c] Step from a to c with step-length b
    [a...b]     Step from a to b with step-length 1
    [](...)     Clone List
    [](n...)    From the n-th element
    [](...n)    To the n-th element
    [](n...-m)  from n up to the m-th-last-element
    ? <- this is the Comment mark

    ? Empty lines will be ignored.
    ? Vartypes
    str: Text = "Hallo Welt!"
    num: Nummer = 3.14
    list: Liste = [1, "2"]
    object: Objekt = NewPUI{}
    path: PfadZurDatei = /usr/bin/var/
    bin: Dual = LOOOLLLOO (L/O)
    hex: Hexadezimal = ABC123
    ? Functions:
    function: name (attr1, attr2) =>
        ...
    EndFunction
EndProgram
```