tVar

Init

```
require("tVar/init.lua")
```

Quick Input

```
tVar.q([string],[bool]optional)
x_lim_b_2 --> x_{lim,b}^2
tVar.q({
"a_asd=10",
"x_lim_b=20"
})
tVar.q("a_asd=10",true) -- with output
```

Global tVar.

```
numFormat = "%.3f"
mathEnviroment = "align"
debugMode = "off"
outputMode = "RES" --RES, RES_EQ, RES_EQ_N
numeration = true
decimalSeparator = "."
eqTexAsMatrix = false
```

New

```
tVar:New(0.04,"r_{se}")
tVec:New({10,2,7},"v_{1}")
tMat:New({{10,2,5},{2,4,3},{7,4,3}},"a_{2}")
```

Output

```
:print() --abh. v OutputMode
:outRES_EQ_N(number[bool],enviroment[bool])
:outRES_EQ([bool],[bool])
:outRES([bool],[bool])
:out() --nur Wert
```

Set [tVar]

```
:setName([string])
:setUnit([string])
:clean(name[string]) --berechn. Schr. entf.
```

Misc

```
[tVar]:bracR() --Runde Klammern
[tVar]:CRLF([string]) --neuwline, [string]
wird vor und nach Umbruch eingefügt
[tVar]:CRLFb([string]) --Umbruch vor [tVar]
[tVar]:copy()
tex.print([string]) --print string to LaTeX
```

Math

```
tVar.sqrt([tVar],[number])
tVar.PI
[tMat]:T() --Transponieren
[tMat]:Det()
[tMat]:Inv()
[tVec]:crossP()

Converted math functions:
[tVar].abs,[tVar].acos, [tVar].cos,
[tVar].cosh, [tVar].asin, [tVar].sin,
[tVar].sinh, [tVar].atan, [tVar].tan,
[tVar].tanh, [tVar].ceil, [tVar].floor,
[tVar].exp, [tVar].log, [tVar].log10,
[tVar].rad, [tVar].deg, [tVar].atan2
```