tVar

Init

```
require("../tVar.lua")
matrix = require("../matrix")
```

Global tVar.

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

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. Schritte
entf.
Tipp: können verkettet werden
```

Misc

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

Math

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

Example

```
\begin{luacode*}
require("../tVar.lua")
matrix = require("../matrix")

numFormat = "%.2f"
outputMode = "RES_EQ_N”
numeration = false

N_D_2=((-V_LF * (0.4*d) + M_LF + H_LF *
H_RB_PL)/(R_Hebel_2)):CRLFb("="):setName("N_
{D,2}"):setUnit("\\kNpm"):print()

\end{luacode*}
```