Hello Typst

Max Musterman, University of Mustercity

mmuster@uni-mustercity.de

An introduction to Typst: https://typst.app

First Section

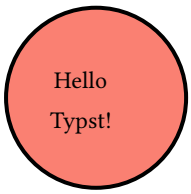
- item 1
- item 2

```
puts "Hello Tcl"
```

```
Hello Tcl
```

```
lappend auto_path ../modules
package require tsvg
tsvg set width 100
tsvg set height 100
# Tcl like syntax without hyphens
tsvg circle cx 50 cy 50 r 45 stroke black stroke-width 2 fill salmon
tsvg text x 29 y 45 Hello
tsvg text x 27 y 65 Typst!
tsvg write hello-typst.svg
### cleanup
tsvg set code ""
puts done
```

```
done
```



An example equation:

```
E = mc^2
```

$$E = mc^2$$

Tcl Lists to Tables

An example for a table given as a Tcl list:

````
```{.tcl results=asis,echo=false}
set h [list A B C]
set d [list 1 2 3 4 5 6 7 8 9]
puts [list2tab $h $d]
```
````

A	B	C
1	2	3
4	5	6

7	8	9

CSV Display

Since version 0.15.0 with Typst it is as well to embed CSV (Comma Separated Values) data into the documents and return the data as text. Here an example:

```
col1,col2,col3,col4
1,2,3,4
5,6,7,8
9,10,11,12
```

col1	col2	col3	col4
1	2	3	4
5	6	7	8
9	10	11	12

As with other code chunks you can as well hide the input by using `echo=false`. Here data from an other table:

col1	col2	col3	col4
11	12	13	14
15	16	17	18
19	20	21	22

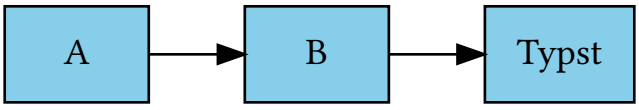
Shell Chunks

Here the code. Replace the single quote with a back tick before the other two back ticks.

```
'''{.shell label=tdot2 cmd="dot -Tsvg %i -o%o" chunk.ext=dot ext=svg}
digraph G {
    rankdir="LR";
    node[style=filled,fillcolor=skyblue,shape=box];
    A -> B -> Typst
}
'''
```

Here the output:

```
digraph G {
    rankdir="LR";
    node[style=filled,fillcolor=skyblue,shape=box];
    A -> B -> Typst
}
```

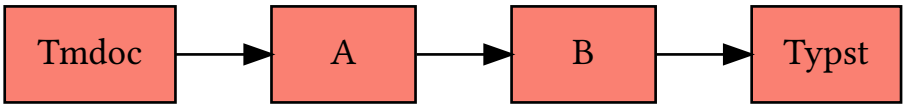


Using the option `echo=false` you can hide the input and only display the image:

```
'''{.shell label=tdot3 cmd="dot -Tsvg %i -o%o" chunk.ext=dot ext=svg echo=false}
digraph G {
    rankdir="LR";
    node[style=filled,fillcolor=salmon,shape=box];
    Tmdoc -> A -> B -> Typst
```

}
```

Here the output:



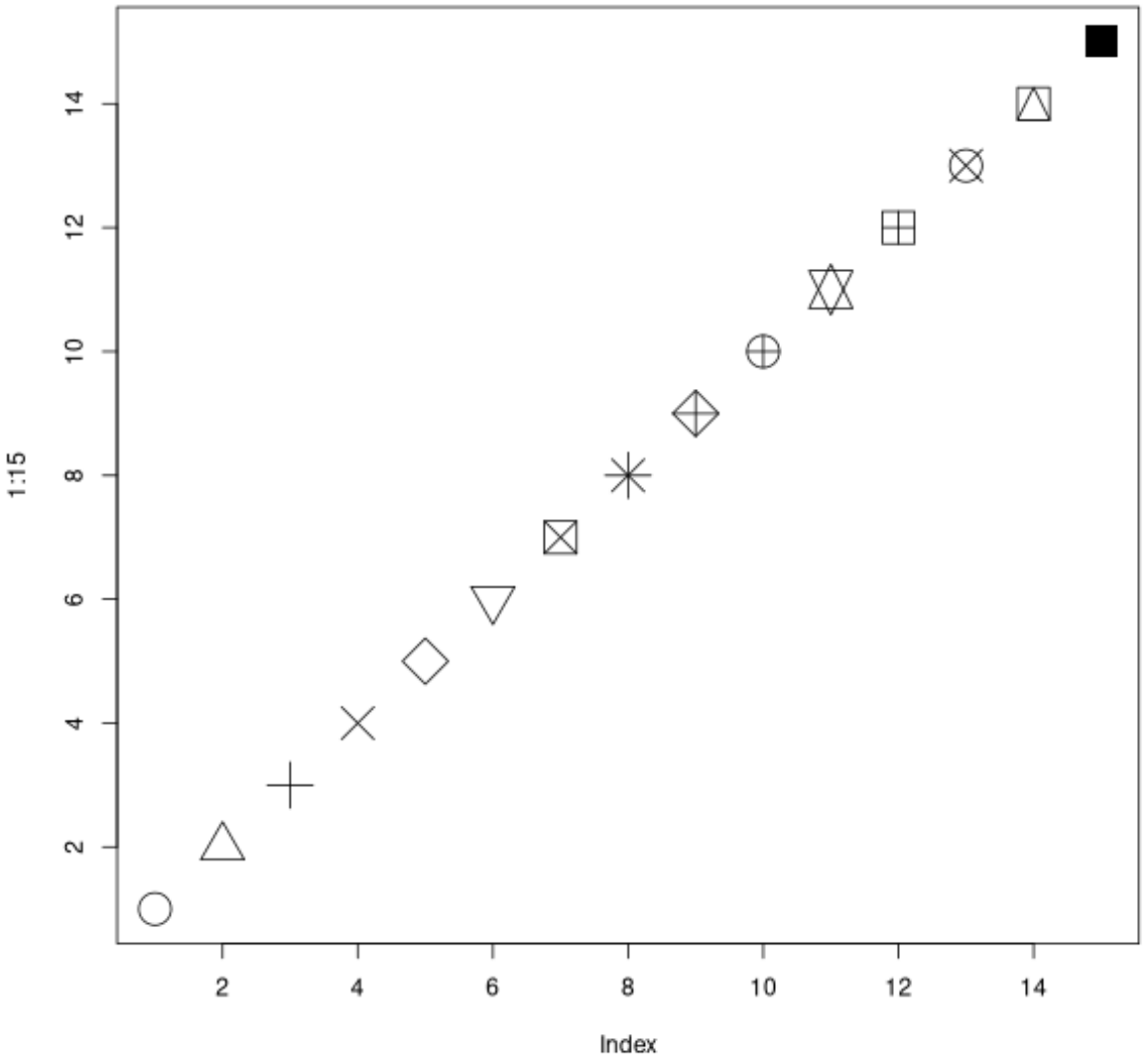

## R Examples

```r
x=1
print("Hello R World!")
```

```
> x=1
> print("Hello R World!")
[1] "Hello R World!"
```

```r
plot(1:15,pch=1:15,cex=3)
```

```
> plot(1:15,pch=1:15,cex=3)
```


```

The value of x is 1!

Python Example

```
x=2
print(x)
x=x+1
import sys
print(sys.version)
```

```
>>> x=2
>>> print(x)
2
>>> x=x+1
>>> import sys
>>> print(sys.version)
3.13.5 (main, Jun 25 2025, 18:55:22) [GCC 14.2.0]
```

The value of Python's x is 3!

Abbreviations

Typst uses it's own abbreviation system. We usually declare some variables and then within the text prefix them with a hash. Here an example:

```
#let abbrev = "Abbreviation"

To use the  abbreviation  we use a hash like this: An #abbrev is a short  text
symbol.
```

And here the output:

To use the abbreviation we use a hash like this: An Abbreviation is a short text symbol.

To use yaml based abbreviations which can be declared within yaml files like this:

```
MM: Max Musterman, University of Mustercity
```

And then in the text you place the abbreviations within curly braces like this:

```
This document was not written by {MM}.
```

The output is then the following:

This document was not written by Max Musterman, University of Mustercity.

To process the abbreviations you need to give the abbreviation file on the command line of tmdoc like this:

```
tmdoc input.ttyp output.typ --abbrev abbrev.yml
typst compile output.typ
```