## About neToolkit

neToolkit is a program designed to allow for quick prototyping of graphical user interfaces. Instead of directly manipulating toolkit code from a particular programming language, the GUI is described in a somewhat visual text input format. From this generalized input description, neToolkit will generate the code which it represents, using a toolkit of the users choice in a language of the users choice. In this manner, the input which describes the resulting interface is both programming language and toolkit agnostic.

This project is developed and maintained by Michael Leimon, a graduate student of the TAMU Nuclear Engineering Department. The naming of this project stems from an effort to write more user friendly programs for the nuclear engineering field. These resulting programs must work on multiple platforms, even if the final product uses different toolkits on different platforms. This goal of this project is to make this possible and simple.

It should also be mentioned that, neToolkit is licensed under a FreeBSD license.

## Examples

### Hello World

This example demonstates, in likely the simplest case, the layout of a general GUI description. Notice that indentation is what signifies children of a section. In this example, there are two top-level sections, *test_test*, and *add_prog*.

The '@' symbol appearing in the '-frame' line of *add_prog*, serves as a element insertion operator. Essentially, the '-frame' element containing '@test_test' is replaced by the code section *test_test*.

**hello.ne**

```
# Hello world example (comment line)
frame test_test
  -text: "Hello World!!"

application   add_prog
  -frame       : @test_test
  -output_name : "hello"
  -source      : "hello_src.py"
  -description : "obligatory first program"
  -language    : python
  -toolkit     : tk
```

**hello_src.py**

```
# hello_src.py
# this file does nothing
```

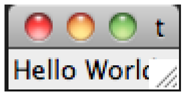


Figure 1: The resulting GUI for ‘hello.ne‘ as seen on OSX.

### Columns and Rows

STUB

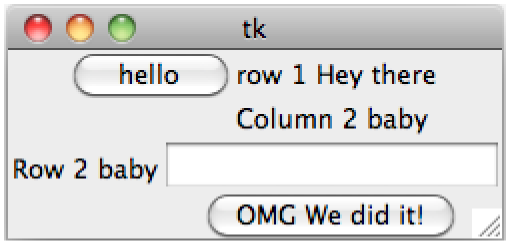
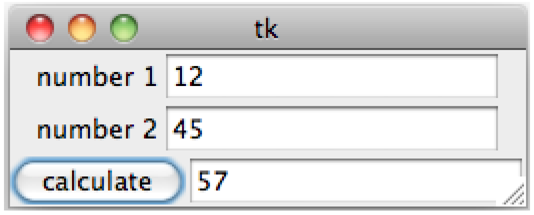


Figure 2: The resulting GUI for ‘short.ne‘ as seen on OSX.

### Addition Program

STUB



number 1 12
number 2 45
calculate 57

Figure 3: The resulting GUI for ‘evalprog.ne‘ as seen on OSX.

### Python Line Evaluator

### evalprog.ne

```
call complex
  -function: *eval_func
  +args
    - (input_line.text)
  -return: (output_line.text)
```

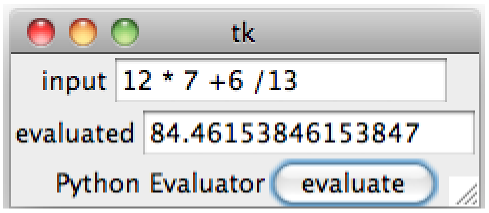
```
frame  eval_frame
  +row
    -text:"input"
    -entry: & input_line
  +row
    -text:"evaluated"
    -entry: & output_line
  +row
    -text: "Python Evaluator"
    +button
      -text: "evaluate"
      +on_click
        -@complex

application   add_prog
  -frame       : @eval_frame
  -output_name : "evalprog"
  -source      : "EvalProg_src.py"
  -description : "evaluate a line of text using python"
  -language    : python
  -toolkit     : tk
```

### EvalProg_src.py

```
#!/usr/bin/env python3
# EvalProg_src.py

def eval_func(a):
  "use python to evaluate a line"
  return eval(a)
```



```
12 * 7 +6 /13
84.46153846153847
```

Figure 4: The resulting GUI for ‘evalprog.ne‘ as seen on OSX.