

TECHNICAL REPORT

(Report No. : NUKLEARMALAYSIA/2020/122)

Macro language construction for *ombakwarna* software.

Part 1: The stand-alone *owl* interpreter.

Megat Harun Al Rashid bin Megat Ahmad

*Materials Structure and Integrity Group,
Industrial Technology Division,
Agensi Nuklear Malaysia, 43000 Kajang, Selangor, Malaysia*

November 6, 2020

Abstract

Spreadsheet based software like *ombakwarna* has data lay out on its spreadsheet. Analysis and computation perform on this data usually require a macro language. This compiled or interpreted based macro language was commonly build using a lexer and parser constructor. In the case of *ombakwarna*, a simple macro language named *owl* was constructed for this purpose. Instead of building a comprehensive lexer and parser constructor, *owl* utilizes the *numpy* library and *python eval()*function. *Owl* was tested as stand-alone interpreter running on terminal and was found quite stable and robust for such task.

Keywords: language, interpreter, lexer, parser.

1 Introduction.

The *ombakwarna*[1, 2] software is a spreadsheet based graphical analysis software utilizing *numpy*[3] and *matplotlib*[4] as analysis and graphical tools. Its design was inspired by *IgorPro*[6] software. It is licensed under GNU General Public License Version 2[7]. The software is still under development starting as pre-alpha version first released on 24 January 2020. A major feature intended for the software is the ability to process data on the its spreadsheet mathematically. This feature is not available on the pre-alpha version and the implementation of this feature on future alpha version requires the construction of a specialized macro language.

This alpha version was planned to be released in the first quarter of 2021. Even though the macro language is targeted to function like a fully functional programming language; to construct such a language will definitely take more times. In retrospect, to achieve a practical macro language that able to do simple mathematical process on data in spreadsheet for now, a very simple language design was envisioned and effectuated. This language does not necessitates the complicated design of lexer and parser, but instead utilizes the *numpy* library and specifically the *python eval()* function. It can be considered as a very simple macro language with its own discerning syntax, making it unique to *ombakwarna*. It is called *owl* for obvious reason *i.e.* ombakwarna language.

2 Design consideration.

The *ombakwarna* was written almost entirely in *python*, so it is reasonable to implement *owl* in *python* as well, for compatibility and continuity. Execution of the *owl* language is envisaged to be done as a line by line interpretation in

ombakwarna command line of text interface, *i.e.* *owl* functions as an interpreter. *Owl* is also designed to operate on array data structure. The operation on array is important because *ombakwarna* transforms data in its spreadsheet cell or column as array only, even though the spreadsheet contains only a number. The ability to process array allows *owl* to function without sequence operation. *Owl* utilizes *numpy* library for this array operation.

2.1 The syntax of the *owl* language.

The *owl* language contains the usual simple mathematical notations as well as some basic mathematical and statistical functions. Logical conditional operation is possible both on single value and array. Table 1 to 3 show the constructs of the *owl* language.

Table 1: Mathematical constructs.

notation	operation	notation	operation
$:=$	assignment	$+$	addition
$-$	substraction	$*$	multiplication
$/$	division	$^$	exponentiation

Table 2: Inequality (logical) constructs.

notation	operation	notation	operation
$=$	equal	\neq	not equal
$<$	less than	$>$	greater than
\leq	less than or equal	\geq	greater than or equal

Initial value for a variable must be set with the assignment statement in the beginning before operation can be done to the variable. A variable need not be declared first but is created the moment a value is assign to it. The variable name characteristics are:

1. Starts only with a letter.
2. Cannot start with a number.
3. Contains alphabet, numerical integer characters and underscores only (A-z, 0-9, _).
4. Case-sensitive.

Table 3: Basic function constructs (example shown for the x variable).

statement	return
$\log(x)$	natural logarithmic value of x
$\sin(x)$	trigonometric sine value of x
$\cos(x)$	trigonometric cosine value of x
$\tan(x)$	trigonometric tangent value of x
$\exp(x)$	exponential value for x
$\text{sum}(x)$	*summation value for x
$\text{mean}(x)$	*average value for x
$\text{median}(x)$	*median value for x
$\text{std}(x)$	*standard deviation for x
$\text{var}(x)$	*variance of x
π	reserved string for constant value of π

* only works for array variable

2.2 Internal structure and perusing of the language.

Owl utilizes the *python eval()* function for parsing and evaluating the interpreter line statement. There is no proper lexer for classifying a line statement, instead a very simple matching and replace routine was implemented to replace the *owl*'s syntax into a *python*'s syntax and thereon to the *eval()* function. This results in a very small but functionally working program. The *owl* program itself has less than a hundred line of working codes. The program code is shown in Appendix A. *Owl* will be assessed as a stand-alone command line interpreter before implementing it in *ombakwarna*. Installation procedure as well as simple instructions on using *owl* as stand-alone interactive interpreter are lay out in Appendix B.

3 Conclusion.

Initial testing of the interpreter indicates reasonable stability and robustness. Future works will focus on its implementation in *ombakwarna*. This shall include the ability to pass array data structure to specified worksheet and vice versa. An eventual full-fledged lexer and parser programs nevertheless are more desirable that will provide *ombakwarna* better capability for data analysis.

References

- [1] <https://sourceforge.net/projects/ombakwarna/>
- [2] <https://github.com/megatharun/ombakwarna>
- [3] <https://numpy.org/>
- [4] <https://matplotlib.org/>
- [5] <https://www.python.org/>
- [6] <https://www.wavemetrics.com/>
- [7] <https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html>

Appendices

A *Owl* program code.

```
#!/usr/bin/env python3
```

```
'''owl: ombakwarna macro language (pre-alpha version)
owl is a very simple interactive language intended to
be implemented as macro language in ombakwarna software.
```

Ombakwarna websites:

- 1) Executibles: <https://sourceforge.net/projects/ombakwarna/>
- 2) Codes: <https://github.com/megatharun/ombakwarna>

Author: Megat Harun Al Rashid bin Megat Ahmad
Copyright (C) 2020 Agensi Nuklear Malaysia

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.'''

```
#-----START-----
```

```
import os
from termcolor import colored
import numpy as np
```

```
print('{owl: ombakwarna macro language (pre-alpha version)}')
print('interactive interpreter, License: GPL version 2')
```

```
print ('Copyright (c) 2020 Agensi Nuklear Malaysia, Bangi}\n')
```

```
dataH = []  
_1 = dict()
```

```
RSV = [                                     # RSV = Reserved string value  
    (':=', '=', '?01?'),  
    ('^', '**', '?02?'),  
    ('=', '==', '?03?'),  
    ('array', 'np.array', '?04?'),  
    ('sum', 'np.sum', '?05?'),  
    ('log', 'np.log', '?06?'),  
    ('sin', 'np.sin', '?07?'),  
    ('cos', 'np.cos', '?08?'),  
    ('tan', 'np.tan', '?09?'),  
    ('pi', 'np.pi', '?10?'),  
    ('exp', 'np.exp', '?11?'),  
    ('mean', 'np.mean', '?12?'),  
    ('median', 'np.median', '?13?'),  
    ('std', 'np.std', '?14?'),  
    ('var', 'np.var', '?15?')  
]
```

```
class Macro:
```

```
    def __init__(self, lineStr):
```

```
        dataH.append(lineStr)  
        self.lineStr = lineStr
```

```
    def evaluate(self):
```

```
        self.lineStrLen = self.lineStr.replace(' ', '').  
                               split(':=')
```

```
        if len(self.lineStrLen)==2:
```

```
            # Right assignment for eval() later  
            self.lineStr = self.lineStrLen[1]
```

```
        else:
```

```

        pass

    for i1,j1 in enumerate(RSV):
        # enumerating RSV
        self.lineStr = self.lineStr.
            replace(j1[0],j1[2])

    for i2,j2 in enumerate([*_1]):
        # embedding reference to data structure
        self.lineStr = self.lineStr.
            replace(j2,"_1['" + (j2) + "']")

    for i3,j3 in enumerate(RSV):
        # enumerating RSV again
        self.lineStr = self.lineStr.replace(j3[2],j3[1])

    if len(self.lineStrLen)==2:
        # passing evaluated expression to
        # data structure dictionary
        _1[self.lineStrLen[0]] = eval(compile
            (self.lineStr,'<string>', 'eval'))
    else:
        # printing output
        print (colored('[out]:\n','red','on_cyan',
            attrs=['bold']), (eval(compile
            (self.lineStr, '<string>', 'eval'))))

while(True):

    try:
        a = input(colored('[in]: ', 'yellow', 'on_cyan', attrs=['bold']))

        if a=='q()': # To exit
            break
        elif a=='quit()': # To exit
            break
        elif a=='': # Empty input
            pass
        elif a==b'\x0c'.decode(): # To clear the screen
            os.system('clear') # for Linux

```



```
        # os.system('cls') # for Windows
        pass

    else:
        b = Macro(a.replace('[in]: ', ''))
        b.evaluate()

except Exception as err: # Error management

    err = str(err)
    for i4,j4 in enumerate(RSV): # enumerating RSV
        err = err.replace(j4[1],j4[0])

#-----END-----
```

B Installation and simple instructions.

B.1 Installation instructions can be found on:

<https://github.com/megatharun/owl/blob/main/README.md>

* upon installation, the interactive interpreter can be invoked by typing '*owl*' and press ENTER in a *bash* terminal for *Linux* (or *command prompt* for *Windows*).

```
~$ owl
{owl: ombakwarna macro language (v. 0.1)
interactive interpreter, License: GPL version 2
Copyright (c) 2020 Agensi Nuklear Malaysia, Bangi}
```

```
[in]:
```

B.2 Performing simple mathematical operations:

```
[in]: 4+8-9
```

```
[out]:
```

```
3
```

```
[in]: 3/7
```

```
[out]:
```

```
0.42857142857142855
```

```
[in]: 2.2^3
```

```
[out]:
```

```
10.648000000000003
```

```
[in]: 4-3/2
```

```
[out]:
```

```
2.5
```

```
[in]: (4-3)/2
```

```
[out]:
```

```
0.5
```

```
[in]: array([1,2,3,4,5])
```

```
[out]:
```

```
[1 2 3 4 5]
```

```
[in]: array([1,2,3,4,5])^3
[out]:
[ 1  8 27 64 125]
```

* Terminal/command prompt can be cleaned by pressing simultaneously *Ctrl* and *L* and then press ENTER.

B.3 Simple mathematical functions, assignment and array operations:

```
[in]: pi
[out]:
3.141592653589793
```

```
[in]: exp(1)
[out]:
2.718281828459045
```

```
[in]: log(exp(1))
[out]:
1.0
```

```
[in]: c:=array([[1,2,3],[7,8,9]])
[in]: c
[out]:
[[1 2 3]
 [7 8 9]]
```

```
[in]: c:=c^0.36
[in]: c
[out]:
[[1.          1.2834259  1.4851272 ]
 [2.01481555  2.11403608  2.20560279]]
```

```
[in]: mean(c)
[out]:
1.6838345872405311
```

```
[in]: mean(c,0)
[out]:
[1.50740778 1.69873099 1.845365  ]
```

```
[in]: mean(c,1)
[out]:
[1.25618437 2.11148481]

[in]: e:=c.T
[in]: e
[out]:
[[1.          2.01481555]
 [1.2834259   2.11403608]
 [1.4851272   2.20560279]]

[in]: e*c[:,1]
[out]:
[[1.2834259  4.25939277]
 [1.64718203 4.46914855]
 [1.90605071 4.66272389]]
```