# Python

*by* Iwan Binanto

**Submission date:** 18-Jan-2018 10:46AM (UTC+0700)
**Submission ID:** 862006451
**File name:** paperTools-Halstead.docx (1.07M)
**Word count:** 1882
**Character count:** 10272

# Python Based Tools for Further Processing of Halstead Metric

[1,2]Iwan Binanto
[1]Informatics Department,
Faculty of Science and Engineering
Sanata Dharma University
Yogyakarta, Indonesia 55002
[2]Computer Science Department, BINUS Graduate Program – Doctor of Computer Science
Bina Nusantara University
Jakarta, Indonesia 11480
iwan@usd.ac.id

Bahtiar Saleh Abbas[1],
Agung Trisetyarso[2], Wayan Suparta[3], Chul - Ho Kang[4]
Computer Science Department, BINUS Graduate Program – Doctor of Computer Science
Bina Nusantara University
Jakarta, Indonesia 11480
bahtiars@binus.edu[1], atrisetyarso@binus.edu[2],
drwaynesparta@gmail.com[3], chkang5136@kw.ac.kr[4]

***Abstract*—To retrieve some data from huge .html files and convert it to .csv files respectively is a long time work if done manually. Consequently, this work needs a tool. This paper discusses developing tool utilizing Python language. It is done using method based on waterfall. Utilizing Pythons's code makes the tools very small in line of code but powerful. It successfully retrieves some data as needed. The tools have successfully been tested to other case studies.**

***Keywords—python programming; halstead metric; waterfall;***

## I. Introduction

There are many published papers with topic of software metrics with various metrics [1].The widely used measurement is Halstead Metric [2], although some researchers found vagueness measuring with Halstead Metrics and considered that Maurice Halstead did not understand the theory of measurement[2][3]. Nevertheless, many papers have been published using this metric[3]–[10].

To calculate Halstead Metrics, it can be done manually or using software. There is a free software written in Java by Ramasubramanian namely *HalsteadMetrics.jar* and *HalsteadMetricsCMD.jar* which easily help calculating Halstead Metrics[11]. Our upcoming research uses one of this software which is *HalsteadMetricsCMD.jar* because it is faster than the other. In our opinion, this software is very helpful in research about measurement software, especially in Halstead Metrics. Those software will generate .html or .pdf files as output files.

We need a .html instead of .pdf files for further processing in our upcoming research. The problem arises since the .html files generated by *HalsteadMetrics.jar or HalsteadMetricsCMD.jar* are more than 1000 files. It will take a long time to retrieve some of the data in the .html files manually, such as blocking and copying one part of data and pasting it into, example, Microsoft Excel, for further processing. Consequently, we need a tool to do it automatically.

This paper discusses about developing tools to fit our needs that will retrieve a part of data from the output of *HalsteadMetricsCMD.jar,* which are .html files, and store it as .csv files. Hopefully, this tool will be helpful for others too.

## II. Literature Review

Spinell has created a tool based on Java to fit his needs and discovered something important, that wrote stand-alone tools, which became a forgotten art[12]. Python is a high-level programming language that allows programmers to work faster and solve problems effectively. It is freely available in source or binary form from the Python site and can be distributed for free[13]. It was designed as a general purpose programming language and then became the valid choice of scientists to help their researches by developing scientific programs. The use of Python has been growing rapidly due to the dedicated work of many contributors over the years. Python has many libraries to express many complex algorithmic questions with clarity and efficiency [14]. Many published papers are related to Python and software metric. Misra and Cafer have worked on evaluating Python, Java, and C++, which are object oriented programming language, and concluded that Python was better than the other two on their case study[15]. Thirumalai, et. al. used a program written in Python as the subject of their research to evaluate Halstead Metric [10]. Pawade et. al. has developed a software to calculate Halstead metrics using Python for automation in order to determine complexity software quality [5]. Cornwell used Python package namely "radon" to estimate software maintainability based on lexical approaches[6]. Shudrak and Zolotarev, by using Python, have found that Halstead's B metrics were the best and most appropriate metric to find bugs[16]. In addition, many published papers were related to Python for various purposes and their usage. Sanner have been using Python as a platform to be redeveloped and operable components related to various aspects of structural bioinformatics[17]. Furthermore, Sanner stated that dynamic is easy to simplify. Besides that, easy-to-use in Python makes this language an excellent choice to create

powerful modern software[18]. Etherington has developed tools in Python for file conversion in his research for use in landscape genetics[19].

## III. Method

Our method is based on steps of waterfall for software engineering [20], which are analysis, design, coding, testing, and maintenance. However, the maintenance step on this waterfall method has never been done because the tools do not need this step yet.

Python is chosen to be the programming language in this paper because it is a high-level language which is easy to help solving technical problems like what is in this paper and has many libraries ready to use [14].

### *A. Analysis*

There are 3 (three) sections of .html display structure generated by halstead.jar as in Fig.3.We need to retrieve the value in VALUES Section only. Consequently, we will prune two previous sections then the result will be converted into one .csv file for data in tabulation. It will ease the further processing. The pruning .html file is illustrated as in Fig.4.

These .html files can be up to 1000 for our upcoming research and must combine these files to be one .csv file only. In the other words, the VALUES Sections, especially the numbers, should only be merged into one .csv file. The data that will be combined are all in the second column of the .csv file. This is where the required data resides. This step is illustrated in Fig. 5. Otherwise, the content of merged .csv files is illustrated in Fig. 6.

### *B. Design and Coding*

We are developing 2 tools for this purpose, which convert the .html file to .csv per file and combine the second column of all generated .csv file. Below are the algorithms. Fig. 1 and Fig. 2 are their flow chart.

Algorithm to convert .html to .csv:

1. Read the .html files in the current directory
2. Convert the existing table in .html files into .csv
3. Save the files with the same .html file name but with.csv extension
4. Repeat steps 1 - 3 until all .html files are read and processed

To implement the algorithm, some libraries must be imported to run properly. Python's code for this implementation is shown as in Fig. 7.

Algorithm to combine all generated.csv files into one file:

1. Read the .csv files in the current directory
2. Retrieve the contents in column 2 and put into the data structure
3. Repeat steps 1 - 2 until all .csv files are read and processed
4. Combine the contents of data structure horizontally and save it into a .csv file

To implement the algorithm, we must import some libraries to make code run properly. Python's code for this implementation is shown as in Fig. 8.

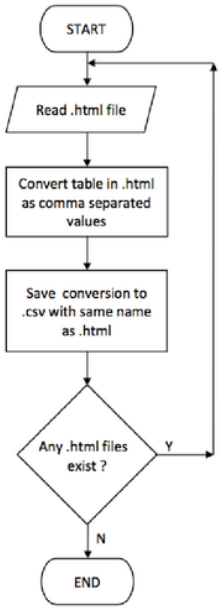


Fig. 1. Flowchart converting .html to .csv

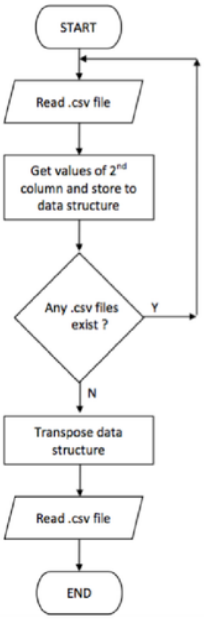


Fig. 2. Flowchart combining all generated .csv files into one file

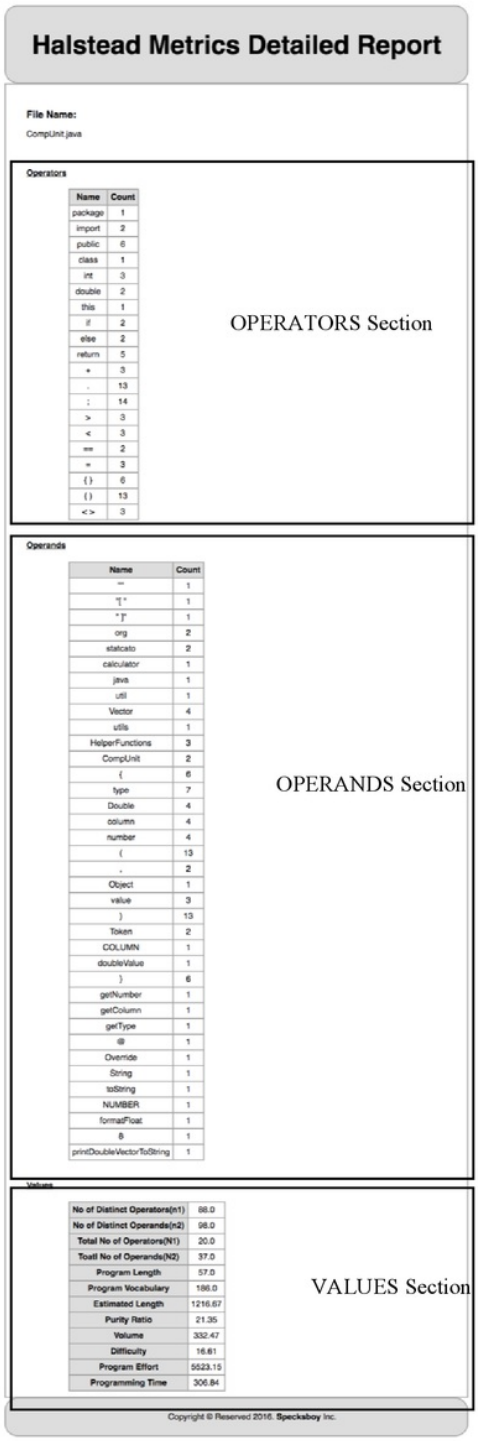


Fig. 3. The .html page structure as output from HalsteadMetrics.jar

### C. Testing

Testing is done with some .html files generated from *HalsteadMetricsCMD.jar*. The result shows that these tools are running well and generating .csv files as needed for further processing. They have already been piloted with other data from different case studies and produce .csv file that can be processed further as needed.

| | |
|---|---|
| No of Distinct Operators(n1) | 88.0 |
| No of Distinct Operands(n2) | 98.0 |
| Total No of Operators(N1) | 20.0 |
| Toatl No of Operands(N2) | 37.0 |
| Program Length | 57.0 |
| Program Vocabulary | 186.0 |
| Estimated Length | 1216.67 |
| Purity Ratio | 21.35 |
| Volume | 332.47 |
| Difficulty | 16.61 |
| Program Effort | 5523.15 |
| Programming Time | 306.84 |

Fig. 4. The results of data pruning in the .html file

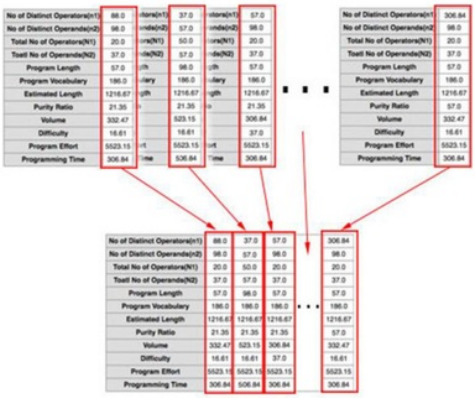

Fig. 5. Illustration of how the second columns merge into one

| | A | B | C | ... | GR | GS |
|---|---|---|---|---|---|---|
| 1 | No of Distinct Operators(n1) | 259 | 551 | ... | 1348 | 1358 |
| 2 | No of Distinct Operands(n2) | 381 | 1121 | ... | 5971 | 6066 |
| 3 | Total No of Operators(N1) | 17 | 24 | ... | 17 | 18 |
| 4 | Toatl No of Operands(N2) | 90 | 191 | ... | 564 | 575 |
| 5 | Program Length | 107 | 215 | ... | 581 | 593 |
| 6 | Program Vocabulary | 640 | 1672 | ... | 7319 | 7424 |
| 7 | Estimated Length | 5342.91 | 16373.73 | ... | 88913.4 | 90361.64 |
| 8 | Purity Ratio | 49.93 | 76.16 | ... | 153.04 | 152.38 |
| 9 | Volume | 721.34 | 1665.86 | ... | 5334.97 | 5462.65 |
| 10 | Difficulty | 30.59 | 46.94 | ... | 63.66 | 64.36 |
| 11 | Program Effort | 22066.1 | 78196.67 | ... | 339644.04 | 351591.66 |
| 12 | Programming Time | 1225.89 | 4344.26 | ... | 18869.11 | 19532.87 |
| 13 | | | | | | |

Fig. 6. The result of merging multiple .csv files into one

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 25 12:12:55 2017

@author: iwanbinanto

Program ini untuk konversi .html hasil pengujian halstead.jar ke .csv
utk memudahkan pengolahan selanjutnya

"""

import pandas as pd
import os
import fnmatch

for filename in os.listdir("./"):
# baca semua file yang ada di directory aktif
    if fnmatch.fnmatch(filename, '*.html'):
    # jika ketemu file .html maka diproses:
        for i, df in enumerate(pd.read_html(filename)):
        # baca isi file .html dan jika ketemu table akan langsung diconvert
        # ke .csv
            df.to_csv('%s.csv' % os.path.splitext(filename)[0])
            # hasil .csv menggunakan nama asal .html tanpa .html
            # nama file .csv akan selalu direplace dengan table baru
            # yang ditemukan, sehingga table terakhir yang akan disimpan
            # dan digunakan.
```

Fig. 7. Pythons's code for converting .html to .csv file

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 25 14:37:33 2017

@author: iwanbinanto

Program ini untuk menggabungkan file-file .csv hasil konversi dari .html
agar menjadi satu file utuh untuk 1 versi software
"""
import os
import pandas as pd
import fnmatch

dfs = []
i=0
for filename in os.listdir("./"):
# baca semua file yang ada di directory aktif
    if fnmatch.fnmatch(filename, '*.csv'):
    # jika ketemu file .csv maka diproses:
        if i == 0 :
        # jika file pertama kali ditemukan, maka 2 kolom pertama diambil,
        # karena kolom pertama merupakan keterangan dari nilai-nilai yg ada
            df = pd.read_csv(filename, usecols=[1,2])
            i=i+1
        else:
            df = pd.read_csv(filename, usecols=[2])
            # baca csv di kolom 2 -> berisi data numerik utk diolah
        df.columns = [filename + str(cname) for cname in df.columns]
        # mengubah nama kolom agar tidak tumpang tindih pada waktu
        # digabungkan
        dfs.append(df)

merged = pd.concat(dfs,axis=1)
# penggabungan secara horisontal, dg opsi axis = 1
merged.to_csv("merged.csv", header=None, index=None)
# tulis ke file yang lain
```

Fig. 8. Pythons's code for combining all generated .csv files

## IV. Conclusion

These tools are successfully developed in Python with a bit line of code. This is because Python has many ready and easy-to-use libraries. These tools help us to get data faster as needed than manually.

## References


[1] Z. G. Dand and H. Vasishtha, "Analysis and Evaluation of Quality Metrics in Software Engineering," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, no. 4, pp. 235–240, 2015.

[2] H. Zuse, "Resolving the Mysteries of the Halstead Measures," in *Büren/Bundschuh/Dumke: Metrikon 2005 – Praxis der Software-Messung, Shaker-Verlag, Aachen, 2005*, 2005, pp. 107–122.

[3] R. E. Al-Qutaish and A. Abran, "An Analysis of the Design and Definitions of Halstead ' s Metrics," *Proc. 15th Int. Work. Softw. Meas. IWSM2005*, pp. 337–352, 2005.

[4] M. J. P. Van Der Meulen and M. A. Revilla, "Correlations between internal software metrics and software dependability in a large population of small C/C++ programs," in *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, 2007, pp. 203–208.

[5] D. Y. Pawade, M. Metha, K. Shah, and J. Rathod, "Python Based Software Complexity Calculator using Halstead Metrics," *Int. J. Adv. Found. Res. Comput.*, vol. 2, no. Special Issue (NCRTIT 2015), pp. 390–394, 2015.

[6] A. I. Cornwell, "Software Maintainability," 2015.

[7] R. Sehgal and D. Mehrotra, "Predicting faults before testing phase using Halstead's metrics," *Int. J. Softw. Eng. its Appl.*, vol. 9, no. 7, pp. 135–142, 2015.

[8] J. Moreno-Leon, G. Robles, and M. Roman-Gonzalez, "Comparing computational thinking development assessment scores with software complexity metrics," *IEEE Glob. Eng. Educ. Conf. EDUCON*, vol. 10–13–Apri, no. April, pp. 1040–1045, 2016.

[9] T. Hariprasad, C. Thirumalai, G. Vidhyagaran, and K. Seenu, "Software Complexity Analysis Using Halstead Metrics," in *International Conference on Trends in Electronics and Informatics ICEI 2017*, 2017, no. May, pp. 1109–1112.

[10] C. Thirumalai, S. R.R., and R. R. L, "An Assessment of Halstead and COCOMO Model for Effort Estimation," in *International Conference on Innovations in Power and Advanced Computing Technologies [i-PACT2017]*, 2017, no. April, pp. 1–4.

[11] H. Ramasubramanian, "Halstead Metrics Tool: Software Metrics Analyzer for Java Programs," 2016. [Online]. Available: https://sourceforge.net/projects/halsteadmetricstool/files/CLI/v2.0/. [Accessed: 05-Jan-2018].

[12] D. Spinellis, "Tool Writing: A Forgotten Art?," *IEEE Softw.*, vol. 22, no. 4, pp. 9–11, Jul. 2005.

[13] Python Software Foundation, "The Python Tutorial — Python 2.7.14 documentation." [Online]. Available: https://docs.python.org/2/tutorial/index.html. [Accessed: 09-Jan-2018].

[14] F. Pérez, B. E. Granger, and J. D. Hunter, "Python: An ecosystem for scientific computing," *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 13–21, 2011.

[15] S. Misra and F. Cafer, "Estimating complexity of programs in python language," *Tech. Gaz.*, vol. 18, pp. 23–32, 2011.

[16] M. Shudrak and V. V. Zolotarev, "Improving Fuzzing Using Software Complexity Metrics," in *Information Security and Cryptology - ICISC 2015: 18th International Conference, Seoul, South Korea, November 25-27, 2015, Revised Selected Papers*, 2016, no. March, pp. 246–261.

[17] M. F. Sanner, "PYTHON: A PROGRAMMING LANGUAGE FOR SOFTWARE INTEGRATION AND DEVELOPMENT," *J. Mol. Graph. Model.*, vol. 17, no. 1, pp. 55–84, 1999.

[18] M. F. Sanner, "The Python interpreter as a framework for integrating scientific computing software-components," *Scripps Res. Inst.*, vol. 26, no. 1, pp. 1–12, 2008.

[19] T. R. Etherington, "Python based GIS tools for landscape genetics: Visualising genetic relatedness and measuring landscape connectivity," *Methods Ecol. Evol.*, vol. 2, no. 1, pp. 52–55, 2011.

[20] I. Binanto, "Analisa Metode Classic Life Cycle ( Waterfall ) Untuk Pengembangan Perangkat Lunak Multimedia," in *Conference: Seminar Nasional Sains dan Teknologi Informasi (SeNASTI) 2014*, 2014, no. MAY 2014, pp. 33–38.

# Python

ORIGINALITY REPORT

| 3% | 1% | 0% | 2% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | Submitted to Binus University International<br>Student Paper | 2% |
| 2 | indico.cern.ch<br>Internet Source | 1% |
| 3 | zenodo.org<br>Internet Source | 1% |

| | | | |
|---|---|---|---|
| Exclude quotes | On | Exclude matches | Off |
| Exclude bibliography | On | | |