Visualizing the Usage of Pythonic Idioms Over Time: A Case Study of the `with open` Idiom

Tattiya Sakulniwat*, Raula Gaikovina Kula†, Chaiyong Ragkhitwetsagul*,
Morakot Choetkiertikul*, Thanwadee Sunetnanta*, Dong Wang †, Takashi Ishio† and Kenichi Matsumoto†
*Faculty of Information and Communication Technology (ICT), Mahidol University
†Nara Institute of Science and Technology (NAIST)
Email: tattiya.sakul@gmail.com, {chaiyong.rag, morakot.cho, thanwadee.sun}@mahidol.ac.th
{raula-k, wang.dong.vt8, ishio, matumoto}@is.naist.jp

***Abstract*—Veterans within the Python community claim that the usage of Pythonic idiomatic writing style is usually preferred. Because of its conciseness and ease of understanding, the idiomatic code tends to be more efficient and less error-prone code. With the growth of Python developers outside the Python community, it is not certain to what extent how Python idiomatic code is used in real software projects, especially if there are consequences. In this paper, our aim is to understand *when* and *how* developers start to use idioms in their software projects. Specifically, we propose a technique to visualize and understand the usage of the `with open` Pythonic idiom, one of the popular idioms. Two visualizations are proposed: (1) a visualization of evolution of non-idiomatic and idiomatic style of writing in four Python software projects over time and (2) a visualization to show the amount of appearing and disappearing idioms by comparing from the first and the latest version of the projects. The results show that developers tend to adopt the idiomatic code over time. We also found that, in three out of the four projects, the developers fixed their code during the evolution of the software to improve their Pythonic coding styles.**

***Index Terms*—Python, Pythonic, Conventions, Programming, Idioms**

I. Introduction

According to the annual GitHub report, the Python programming language ranks in the top three most used by programmers[1]. It is also known as one of the most used language due to its ease of use and understandable syntax[2].

To take full advantage of Python as designed by the founders, the Zen of Python[3] was created with a collection of 19 'guiding principles' for writing computer programs. Over time, the community of Python programmers, especially veterans, has come to regard these styles of programming as *idioms*, that are unique for Python. Like human language, the Pythonic idiom is a series of functional command written in Python language, specified for each specific task [2, 8, 11].

Not all Python developers use idioms, especially if the developers are not from the Python community. The most common case is when the programming language is acquired as a second language. As such, some of the coding styles from another language is implemented. The deficiency of Python writing style also often happens with Python beginners in the case that the way to write proper Python code can be non-obvious to them [7].

Consequently, there are cases where not using an idiom could be harmful to the code quality. One example that obviously shows the benefit of following idiomatic coding style in Python is the case of the '`with open`' idiom, which is designed to automatically close a file after it has been opened. Python developers that do not use this idiom need to explicitly close the file by themselves. The `with` statement is better because it ensures that the file is always subsequently closed, even if an exception is raised inside the `with` block.

In this paper, we would like to explore the usage of Pythonic idioms, especially in the case of `with open` idiom (will be referred to as *idiom* from now on for brevity) over the lifespan of a software project. To track the usage of the idiom, we propose two visualizations to analyze how and when the idiom is adopted and fixed. The results show that developers tend to adopt the idiom over time. We also found that, in three out of the four projects, the developers fixed their code, i.e., non-idiomatic code is removed and the idiom is instead introduced, during the evolution of the software to improve their Pythonic coding styles.

The contributions of this paper are as follow:

- Visualization methods that clearly show the differences between the `with open` Pythonic idiom and its non-idiomatic counterpart adopted in software projects.
- An analysis of the usage of the `with open` idiom, one of the most popular Pythonic idioms in four open source projects spanning over their lifetime of 6 to 13 years.

II. With Open Idiom

In this section, we describe the `with open` idiom in more details. As shown in Figure 1 and Figure 2, two code snippets show the difference between a non-idiomatic and idiomatic way of opening and reading a file.

```
f = open('file.txt')
a = f.read()
print a
f.close()
```

Fig. 1: Example of non-idiomatic code to read a file

[1]https://octoverse.github.com/projects#languages
[2]https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2019
[3]https://www.Python.org/dev/peps/pep-0020/

```
with open('file.txt') as f:
  for line in f:
    print line
```

Fig. 2: Example of idiom to open and read a file

Figure 1 presents an example of non-idiomatic code to open and read a file. We can see that, at the end of the code snippet, we need to explicitly call the `f.close()` function to complete the reading of the file.

In contrast, Figure 2 shows the `with open` syntax to similarly read from a file. This idiomatic code automatically closes the file for the developers after the file reading process is completed. The advantage of using this idiom is a guarantee that the opened file will also be closed after execution, even if an exception is raised inside the `with` block.

III. Empirical Study

In this section, we first present our two research questions with motivations. Then we introduce the criteria for data collection and summarize chosen studied projects. Finally we explain how we extracted the idiomatic and non-idiomatic code during the experiment.

A. Research Questions

We formulate the following two research questions as part of our goal to find when and how the `with open` Pythonic idiom is used in software projects:

- **(RQ$_1$): *Do developers adopt the idiom for file reading statements over time?***
  Motivation: We are interested in the Python reading file statements in this study because it is one of the most frequently found Pythonic idioms in the wild. From the study of Alexandru et al. [2], based on 1,000 popular Python projects, the most popular Pythonic idiom used is `with open`. The answer to this research question would give us insights into the trend of the adoption of the `with open` idiom over the evoluation of software projects.
- **(RQ$_2$): *Do the developers fix their non-idiomatic file reading statements?***
  Motivation: We aim to perform a detailed quantitative analysis of the adoption of the two alternatives of file reading statements, the `with open` idiom and the non-idiomatic counterpart, in each file within the project to see how many non-idiomatic codes are removed and how many idioms are added into each project.

B. Data Collection

We have two criteria for selecting open source Python projects in this study. First, the selected projects must contain the idiom or its counterpart that we are interested in. Thus, either `f=open('file.txt')` or `with open('file.txt')` must appear in the projects. To achieve that, we prepared two code snippets as queries to search for software projects in the source code search engine called *searchcode.com*[4]. After the search, we picked the projects that contain the two code snippets as our candidates. Second, the project candidates are then filtered again based on their number of releases. We are interested in studying the adoption of the idiom over the project lifespan. So, only the projects with multiple releases were kept. By having multiple releases, we can provide a meaningful visualization on the first and the last version of the software project.

TABLE I: Summary of 4 studied projects.

Project	Releases	Contributors	Start Date	End Date
Beaker	432	63	12/08/2006	21/05/2019
DFHack	123	112	24/02/2010	28/12/2018
IPython	94	608	14/09/2008	03/07/2019
TShock	70	62	03/06/2013	01/04/2019

C. Studied projects

Table I provides the overview of the studied projects. In this study, we select four projects, Beaker, DFHack, IPython and TShock, from a set of projects found during the data collection step. All of these projects are written in Python. Beaker is an open source software for managing and automating labs of test computers. DFHack is a Dwarf Fortress memory access library, distributed with a wide variety of useful scripts and plugins. IPython is a command shell for interactive computing in multiple programming languages originally developed for Python. TShock is a toolbox for Terraria servers and communities.

We selected these four projects because they satisfied the previously mentioned criteria: they contain the non-idiomatic and idiomatic codes in file reading statements and they have multiple releases. Also, the duration of the project is long enough to make a useful observation of the evolution of idiom usage. The shortest lifespan of the four projects is TShock with 70 releases over 6 years, and the longest one is Beaker with 432 releases over 13 years.

D. Data Preparation

Before creating the visualizations, we need to gather the number of usage of the non-idiomatic `f=open('file.txt')` code and the `with open('file.txt')` idiom in the four selected projects. Figure 3 summarizes the data preparation process. First, we cloned the projects from GitHub. Then, we used the tool called CCGrep [14] to find the occurrences of the two snippets in the target projects. CCGrep is a code clone detector that searches for clones using pattern matching like grep[5]. We used the tool to query for statements that have the "`with open $`" and "`open =`" sequences. It uses regular expressions and tokenizers to identify these statements. Then, we collected and stored the output of each query in two different files. Lastly, we wrote a Python script to convert the

[4]https://searchcode.com

[5]At the time of writing this paper, the tool is under review at ICSE '19.

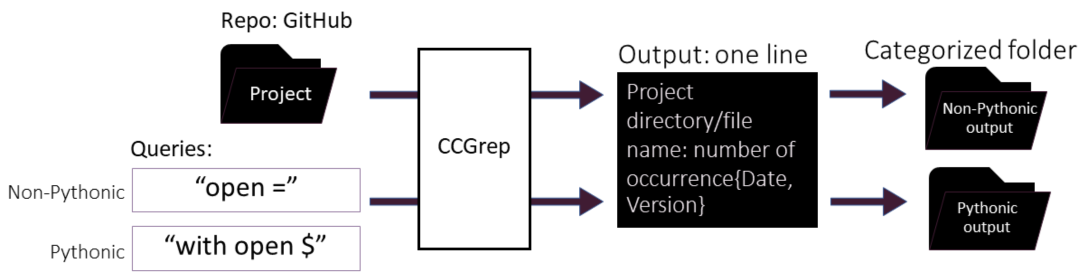


Fig. 3: Data Preparation

output files to CSV files to facilitate the visualization, i.e., plotting by reading from the CSV files.

IV. Results and Discussion

In this section, we present the results and then answer each of the research questions.

a) (RQ$_1$): **Do developers adopt the idiom for file reading statements over time?***:* The following figures (see Figure 4) show the occurrences of non-idiomatic and idiomatic style of `with open` file reading statement in each package of the selected projects per year, in the form of a scatter plot. The x-axis denotes the release date of the project and the y-axis is the name of packages in the project. The red dots represent the occurrences of non-idiomatic code and the green dots represent the occurrences of the idiom. 3 out of the 4 projects contained the idiom so we will discuss only the 3 projects here.

Figure 4a shows the occurrences of the usage of file reading statement in Beaker project. In this project, the developers introduce only the idiom. In the graph, the idiom initially appears in the `doc` page at the beginning (2013). Then, the number of packages that adopt the idiom slightly increase (during 2014 to 2019). At the end, there were 8 packages in which the idioms were found including `doc`, `tools`, `server`, `labcontroller`, `inittest`, `Misc`, `documentation` and `client`.

Figure 4b is the visualization for DFHack project. In this project, the developers introduce both idiomatic and non-idiomatic file reading statements (green is idiomatic and red is non-idiomatic). At the beginning of the project, the non-idiomatic codes were introduced. Then, they started to disappear in the middle to the end, while the idiom began to be introduced a bit later in the middle to the end of the project with an incremental trend through time.

Figure 4c shows the usage of file reading statement in IPython. The developers of this project present both idiomatic and non-idiomatic codes all over the project. The non-idiomatic code mostly appeared at the beginning of the project. Then, it occurred less in the middle to the last release. On the other hand, the idiom started to be introduced into the project mostly from the mid of 2011 until the last release of the project (end of 2014) with an increasing trend.

Summary: We found that 3 out of the 4 selected projects adopt the `with open` idiom. However, the amount of adoption differs. One project only contains the idiom without the non-idiomatic counterpart. The other two projects have non-idiomatic code more at the beginning and idiomatic code more from the middle to the last release.

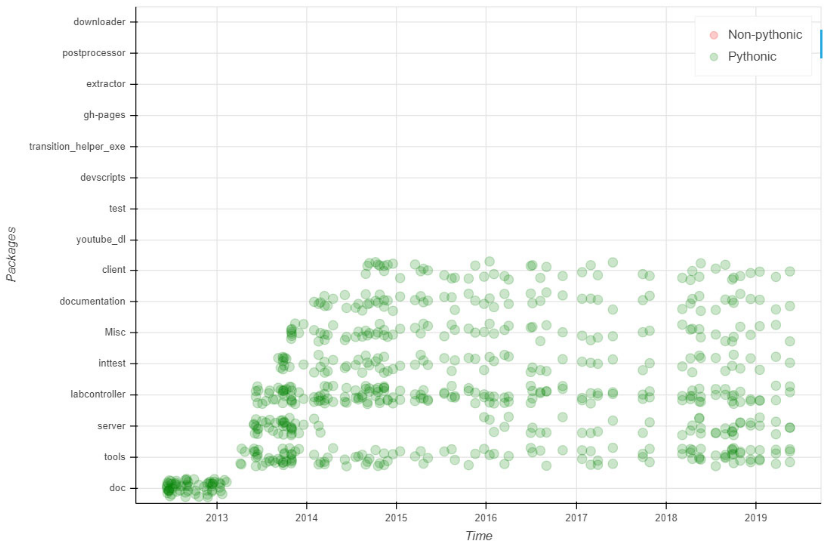
b) (RQ$_2$): **Do the developers fix their non-idiomatic file reading statements?***:* The bar graphs in Figure 5 shows the occurrences of the idiom and non-idiom usage of the file reading statement in each file in the first and the last version of the project in IPython project. We only discuss the findings in the IPython project here since it shows an interesting trend of idiom adoption. The other project results can be found from the study website[6]. The number of found idioms is represented in the y-axis and the file name is represented in the x-axis. The red bar shows the amount of non-idiomatic code found and the green bar shows the idiomatic code found.

From the two graphs, we can see that the non-idiomatic codes first appear in the project's first release, then they are mostly removed from the files during the project evolution. The code, however, occurs in some other files in the last version instead. Interestingly, we can clearly observe from the visualization that, in the last revision, the `with open` idiom occurs in almost half of the files in the project.

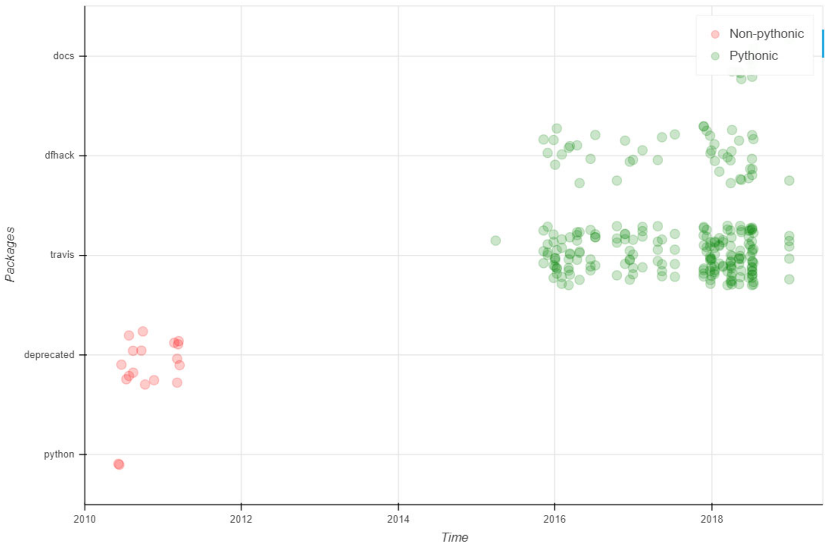
Thus, for future work, we could investigate other popular Pythonic idioms (i.e., if-statement) and how these idioms evolve over time as well. We also could enlarge the diversity of analyzed projects to see what difference among different types of projects. Another interesting work is to build a more interactive tool to visualize evolution results.

Summary: We found that two projects (Beaker and DFHack) contain removals of non-idiomatic code along with the inclusions of the `with open` idiom. Nonetheless, there are some projects that also introduce more non-idiomatic file reading statements (TShock and IPython).

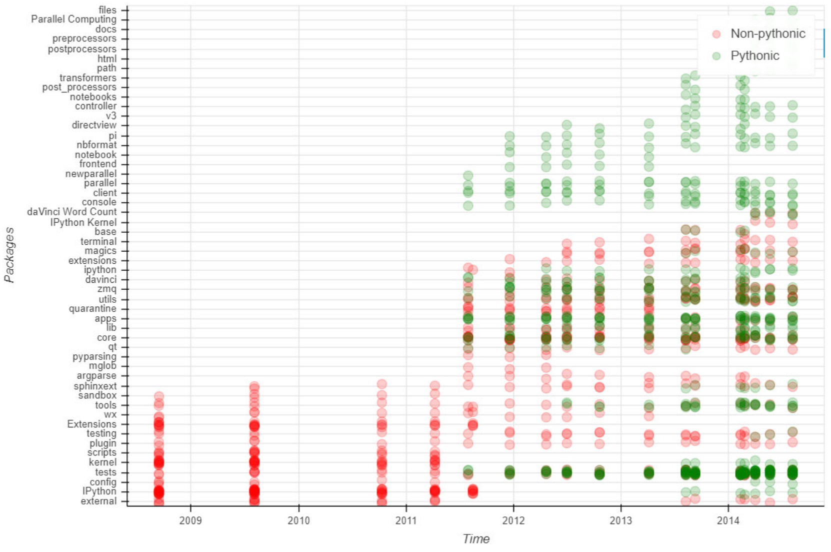
[6]Study website: https://muict-seru.github.io/iwesep19-idioms/



(a) Beaker

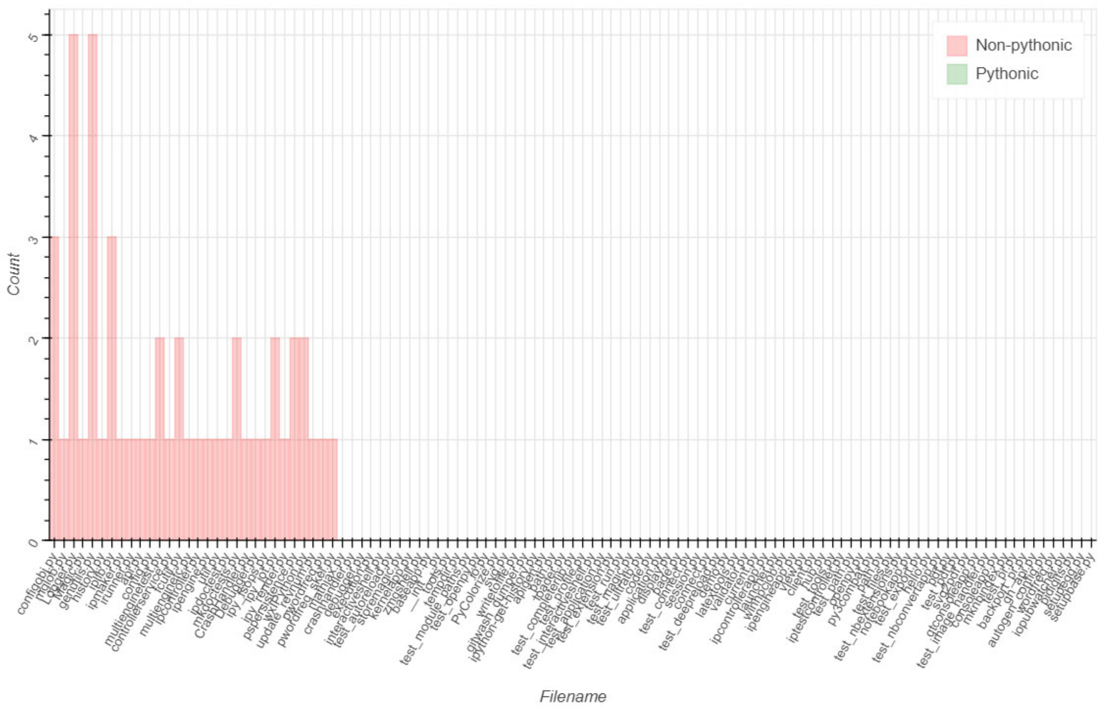


(b) DFHack

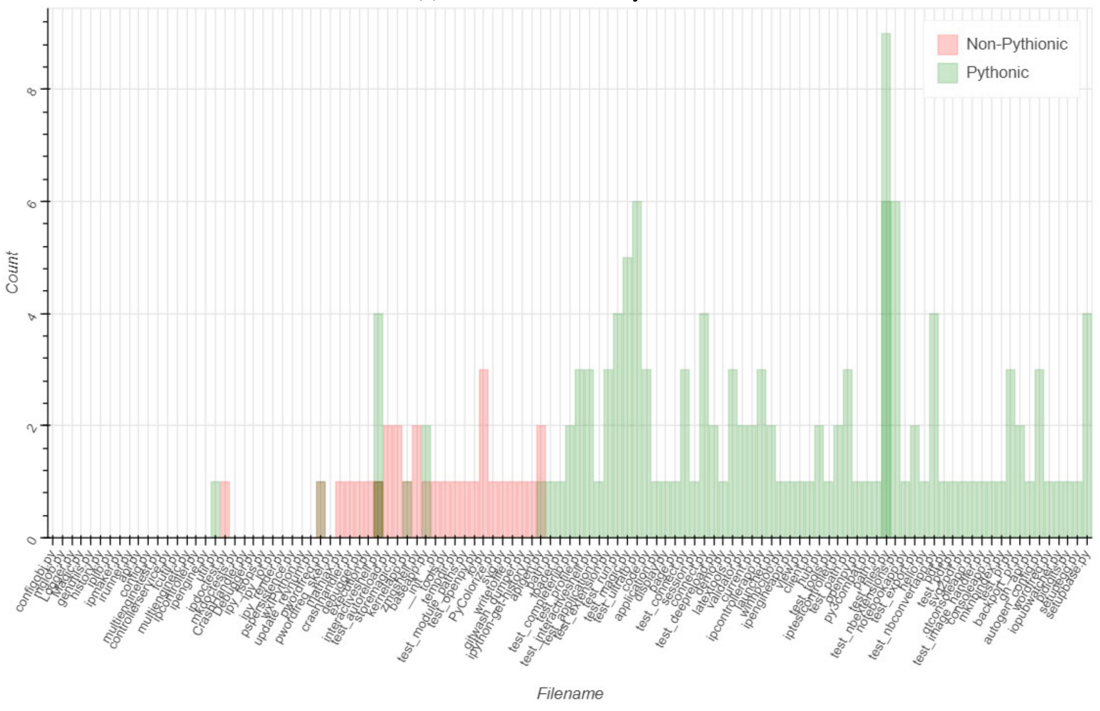


(c) IPython

Fig. 4: Occurrences of the `with open` idiom and its non-idiomatic counterpart



(a) First Version of IPython



(b) Latest Version of IPython

Fig. 5: The comparison between the first and the last release of IPython

V. Threats to Validity

In this section, we discuss potential threats to validity, which are divided into construct validity and generalizability, of our work as follows. For construct validity, in the data preparation process, we rely on one code clone detector tool, CCGrep, to extract the code snippets. The tool may produce some false positive and false negative results. Moreover, we did not check if the disappearing non-Pythonic code is actually replaced by Python idioms. This detailed investigation will be done in the future work. Another threat is the generalization of the results. In our study, we only focus on one type of idiom: `with open`. This idiom is important to study since it is one of the most widely used idioms in open source software projects [2]. The findings may be different for other Python idioms. Lastly, our findings are based on the analysis of four Python projects. They may not be generalized to other projects.

A. Related work

There are a few studies about language features. Parnin et al. [10] study the adoption and use of Java generics in 40 programs. Dyer et al. [3] studied 31,000 open source Java projects to look for the usage of new Java features such as enhanced-for loops or pre-defined annotations.

Though few studies focus on the visualization of the evolution of programming idioms, there exist many tools proposed for visualizing code clones. Tairas et al. [13] described a new approach to display visualization of clone detection results that takes advantage of the ability to extend the Visualiser plugin known as *AJDT Visualiser*. Livieri et al. [9] presented a tool named *D-CCFinder* which is a distributed approach at large-scale code clone analysis. Another tool called *SoftGUESS* was proposed by Adar and Kim [1] representing a novel way of looking at code-clones in the context of many system features.

Research in recent years extends to the whole spectrum of clone management [12]. To interrelate the system's structure with the clone detection results, Hauptmann et al. [5] suggested using edge bundle views for visualization. To better understand the evolution of code clones, Hanjalić [4] propoesd *ClonEvol* which helps developers analyze the evolution of code clones. Honda et al. [6] built a system named *CCEvovis* that visualizes the evolved code clones across multiple versions of a program to support maintenance of code clones.

VI. Conclusion

This paper presents a novel method on visualization and analysis of adoption and evolution of the popular `with open` Pythonic idiom. The analysis is performed on four open source software projects with multiple releases over several years.

The two visualizations of the evolution of non-idiomatic and idiomatic file reading statements and the amount of appearing and disappearing idioms reveal that the Python developers increasingly adopt the `with open` idiom in their projects over time, which can be seen in the first visualization. Moreover, the developers also removed the non-idiomatic version of the file reading statements and added more idiomatic code in the projects. This phenomenon can be observed from the second visualization that compares the adoption of non-idiomatic and idiomatic code between the first and the latest release.

This study is among the first to employ visualization techniques for studying Pythonic idioms. It may be useful for future studies on coding idioms and software visualization.

Acknowledgment

This research project was partially supported by Faculty of Information and Communication Technology, Mahidol University and JSPS KAKENHI Grant Numbers 18H04094, JP18KT0013 and 17H00731.

References

[1] E. Adar and M. Kim. Softguess: Visualization and exploration of code clones in context. In *ICSE'07*, pages 762–766, 2007.

[2] C. V. Alexandru, J. J. Merchante, S. Panichella, S. Proksch, H. C. Gall, and G. Robles. On the usage of pythonic idioms. In *Onward! '18*, pages 1–11, 2018.

[3] R. Dyer, H. Rajan, H. A. Nguyen, and T. N. Nguyen. Mining billions of ast nodes to study actual and potential usage of java language features. In *ICSE '14*, pages 779–790, 2014.

[4] A. Hanjalić. Clonevol: Visualizing software evolution with code clones. In *VISSOFT '13*, pages 1–4, 2013.

[5] B. Hauptmann, V. Bauer, and M. Junker. Using edge bundle views for clone visualization. In *IWSC '12*, pages 86–87, 2012.

[6] H. Honda, S. Tokui, K. Yokoi, E. Choi, N. Yoshida, and K. Inoue. CCEvovis: A clone evolution visualization system for software maintenance. In *ICPC '19*, pages 122–125, 2019.

[7] T. S. Kenneth Reitz. Code style: Idiom. In *The Hitchhiker's Guide to Python: Best Practices for Development 1st Edition*, 2018.

[8] J. Knupp. *Writing Idiomatic Python*. 2013.

[9] S. Livieri, Y. Higo, M. Matushita, and K. Inoue. Very-large scale code clone analysis and visualization of open source programs using distributed CCFinder: D-CCFinder. In *ICSE '07*, pages 106–115, 2007.

[10] C. Parnin, C. Bird, and E. Murphy-Hill. Adoption and use of java generics. *Empirical Software Engineering*, 18(6):1047–1089, Dec 2013.

[11] U. Rey and J. Carlos. From Python to Pythonic : Searching for Python idioms in GitHub. In *Seminar Series on Advanced Techniques & Tools for Software Evolution*, pages 1–4, 2017.

[12] C. K. Roy, M. F. Zibran, and R. Koschke. The vision of software clone management: Past, present, and future (keynote paper). In *CSMR-WCRE '14*, pages 18–33, 2014.

[13] R. Tairas, J. Gray, and I. Baxter. Visualization of clone detection results. In *EXT '06*, pages 50–54, 2006.

[14] K. I. Yuya, Miyamoto. Code clone detector like grep command tool: CCGrep (In Japanese). In *IPSJ/SIGSE Software Engineering Symposium (SES2019)*, 2019.