IEICE TRANSACTIONS

on Information and Systems

DOI:10.1587/transinf.2023EDP7238

Publicized:2024/04/16

A PUBLICATION OF THE INFORMATION AND SYSTEMS SOCIETY

The Institute of Electronics, Information and Communication Engineers

Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER

Unveiling Python Version Compatibility Challenges in Code Snippets on Stack Overflow

Shiyu YANG[†a)], **Tetsuya KANDA**[†b)], **Daniel M. GERMAN**[††c)], *Nonmembers*, *and* **Yoshiki HIGO**[†d)], *Member*

SUMMARY Stack Overflow, a leading Q&A platform for developers, is a substantial reservoir of Python code snippets. Nevertheless, the incompatibility issues between Python versions, particularly Python 2 and Python 3, introduce substantial challenges that can potentially jeopardize the utility of these code snippets. This empirical study dives deep into the challenges of Python version inconsistencies on the interpretation and application of Python code snippets on Stack Overflow. Our empirical study exposes the prevalence of Python version compatibility issues on Stack Overflow. It further emphasizes an apparent deficiency in version-specific identification, a critical element that facilitates the identification and utilization of Python code snippets. These challenges, primarily arising from the lack of backward compatibility between Python's major versions, pose significant hurdles for developers relying on Stack Overflow for code references and learning. This study, therefore, signifies the importance of proactively addressing these compatibility issues in Python code snippets. It advocates for enhanced tools and strategies to assist developers in efficiently navigating through the Python version complexities on platforms like Stack Overflow. By highlighting these concerns and providing a potential remedy, we aim to contribute to a more efficient and effective programming experience on Stack Overflow and similar platforms.

key words: *Stack Overflow, Python, Code Snippets, Version Compatibility*

1. Introduction

Stack Overflow is a popular developer Q&A platform that provides a venue for disseminating knowledge and exchanging ideas, allowing users to post questions, provide answers, and search for content that interests them. Code snippets represent the core knowledge shared on Stack Overflow, as users regularly integrate them into their questions or answers for heightened clarity and understanding. As of August 2023, Stack Overflow has over 24 million questions, and 35 million answers [1]—a testament to its active user base and a number that continues to grow daily.

Within these questions and answers lies a treasure trove of code snippets. This abundance of available code offers developers an easy path to finding solutions to everyday programming challenges. It is commonplace for developers to copy code examples from Stack Overflow, highlighting its integral role in the development process [2].

[†]The author is with the Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565-0871, Japan

[††]The author is with the Department of Computer Science, University of Victoria, Victoria, BC V8P 5C2, Canada

a) E-mail: yangsy@ist.osaka-u.ac.jp

b) E-mail: t-kanda@ist.osaka-u.ac.jp

c) E-mail: dmg@uvic.ca

d) E-mail: higo@ist.osaka-u.ac.jp

Despite the convenience Stack Overflow offers in finding the needed code snippets, recent research has highlighted that these code snippets can be toxic [3], outdated [4, 5], or of low quality [6]. These issues can further lead to compromised software quality [4, 7], license violations [8], or migration of security vulnerabilities [9].

Numerous factors contribute to the problems associated with code snippets on Stack Overflow, with the use of outdated programming language features in these code snippets being a major concern. Programming languages are inherently fluid, constantly evolving to meet new needs and sustain longevity. To signify this evolution, popular programming languages often employ versioning, wherein more recent versions generally denote more mature forms of the language.

The issue of backward compatibility is a key consideration for many programming languages as they evolve. This principle would allow programs written in an earlier language version to be compiled and run using a later version while exhibiting the same behavior as in the previous version. This principle, however, is subverted by Python. With the release of Python 3.0, the language intentionally broke backward compatibility with its predecessor, Python 2.

This lack of backward compatibility poses significant challenges to the users of Stack Overflow. For instance, a user might find a useful Python 2 code snippet on the platform that may not be directly compatible with their Python 3 project. Consequently, the usability of Python snippets on Stack Overflow could be significantly hampered due to these compatibility issues.

We conducted an empirical study to comprehend the challenges of version compatibility within Python code snippets on Stack Overflow. Our investigation pivots around the following key research questions:

- **RQ1: How many Python code snippets have version compatibility issues in the good answers to Stack Overflow questions?**
  Answer to RQ1: About 13% of code snippets have version compatibility issues in the good answers to questions.
- **RQ2: How many of the code snippets interpretable only by Python 2 or only by Python 3 have Python version-specific identification?**
  Answer to RQ2: Only about 20% of code snippets interpretable only by Python 2 or only by Python 3 have accurate Python version-specific identification.

- **RQ3: How do users on Stack Overflow react and adapt to the introduction of new Python releases?**
  Answer to RQ3: After the release of a new Python version, responses can generally be received on Stack Overflow on the same day.

This paper serves as an extension and refinement of our previous research [10, 11]. To clarify the importance of Python version compatibility issues, we refined our research focus based on the definitions outlined in our tool paper [11]. Specifically, we shifted from analyzing all Python code snippets in the technical report [10] to concentrating solely on those in the good answers. We conducted these revised experiments using the same analytical system detailed in the tool paper to ensure methodological consistency and accuracy. Additionally, we reconstructed the dataset initially employed in the tool paper to ensure the precision of our analysis.

2. Background

2.1 Stack Overflow

Stack Overflow, a widely used online community platform, serves as an essential resource for developers looking to share and acquire programming knowledge through a question-and-answer (Q&A) format. Questions posted on the platform invite solutions from the community, which are formatted as answers. Crucial to Stack Overflow's efficacy is the code snippets embedded within these questions and answers. These provide specific contexts or demonstrate direct solutions, acting as ready-to-use references. This characteristic significantly bolsters Stack Overflow's appeal among developers.

For instance, consider a post titled "How do I check if a variable exists?[†]", as shown in Figure 1. In this post, the questioner employs a blend of textual descriptions and code snippets to articulate the question effectively, using three tags to encapsulate the question's topic succinctly. The answer presents a concise solution exemplified via a code snippet. This instance not only underscores the issue and its precise solution but also illuminates the collaborative and community-driven nature of knowledge exchange that defines Stack Overflow.

Beyond the code snippets, Stack Overflow has other distinguishing features that enhance its utility. One such feature is its community voting system, which allows users to upvote or downvote questions and answers based on their usefulness or relevance. The platform also employs a tagging system to categorize questions, streamlining the search and discovery process.

2.2 The Evolution of Python and Version Compatibility

Backward compatibility refers to the ability of software to function seamlessly when a newer version of the language

[†]https://stackoverflow.com/questions/843277

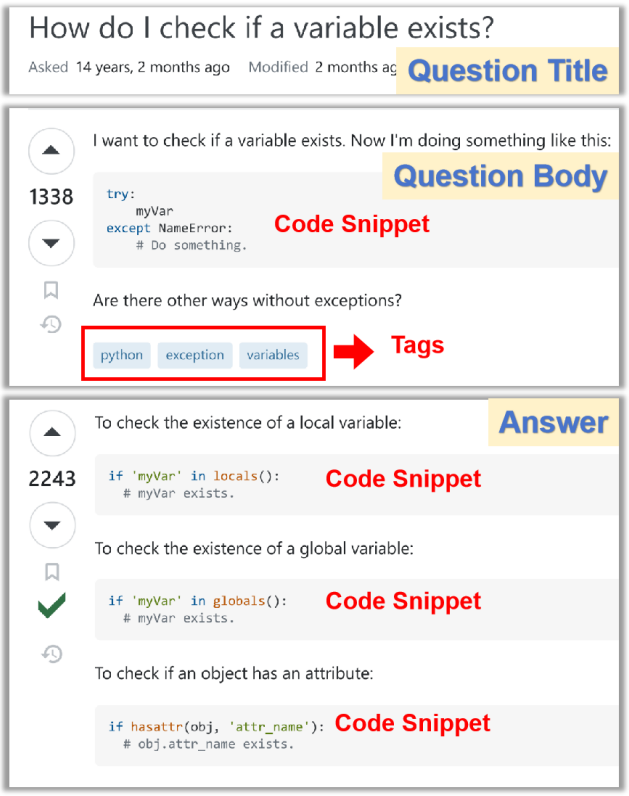


Fig. 1 An example of a Q&A post on Stack Overflow

is used without modifying the existing code. Ensuring this continuity gives developers confidence that their previously written code will not become obsolete with each new language update.

2.2.1 Development history of Python

Python 2.0 was released in October 2000, bringing various features that appealed to developers. Known for its readability and uncluttered syntax, the language comes with an array of built-in data types, such as tuples, lists, sets, and dictionaries. Beyond these core capabilities, Python also boasts a comprehensive standard library and a rich repository of user-contributed packages, facilitating rapid prototyping and effective system integration. Its powerful scripting capabilities also make it versatile for various tasks [12]. These advantages have collectively propelled Python to become one of today's most popular and widely used programming languages.

The trajectory of Python experienced a monumental shift in 2008. The launch of Python 2.6 aimed to prolong the success of the popular 2.x series, guaranteeing a continuation of support and development. Concurrently, Python 3.0 emerged as a transformative iteration focusing on modernizing and refining the language, even at the expense of

forfeiting backward compatibility.

Although Python 2 offered a rich feature set, it eventually hit a technological ceiling that hindered further advancements. This prompted the Python community to make a calculated shift, strategically phasing out Python 2 in favor of the more progressive Python 3. The community's decision reached its apex when support for Python 2.7 was officially terminated on January 1, 2020, marking the series' end-of-life [13].

This strategic bifurcation led to both gains and drawbacks. On the positive side, it enabled Python to innovate and push the boundaries, solidifying its rapid ascendance in the tech world. However, this bold move also led to a bifurcation within the Python community and posed challenges for projects transitioning from Python 2 to Python 3 [14].

2.2.2 Backward Compatibility in Python Versions

In its evolution, Python has been known to break backward compatibility almost with each new release. This policy, including rules that govern the instances when compatibility can be broken, is meticulously documented in PEP 387 [15]. Python 3.0, for instance, markedly broke backward compatibility with Python 2. Moreover, each subsequent release since Python 3.5 has seen the removal of deprecated features essential for running older Python programs, with most releases introducing minor changes that impact only a small proportion of language features.

2.2.3 Compatibility Issues of Python Snippets on Stack Overflow

The parallel paths of Python 2 and 3 heralded not only unique challenges in language enhancement but also implications for the vast reservoir of Python snippets on platforms like Stack Overflow. Historically, Python 2 was rich in capabilities but eventually encountered a juncture where it couldn't encompass newer advancements. This prompted Python's custodians to strategically sunset Python 2 in favor of Python 3, culminating in the community's decision to officially terminate support for Python 2.7 as of January 1, 2020, marking the end-of-life for this Python iteration [13].

As an integral knowledge repository, Stack Overflow is brimming with Python code snippets spanning various versions. Yet, the divergence between Python 2.x and 3.x has cast a shadow over the applicability of these snippets. Many code snippets authored during the Python 2.x epoch may not operate smoothly within the Python 3.x framework, sparking concerns about their lasting utility. This landscape accentuates the pressing need to assess the Python snippets on Stack Overflow critically. It's essential to ascertain their compatibility with contemporary Python standards, ensuring they remain valuable resources for the vast and evolving community of Python developers.

3. Study Approach

3.1 Data Collection

To unveil the challenges of version compatibility within Python code snippets on Stack Overflow, we decided to use the collection of Python code snippets from the "good answer" on the platform. As introduced in Section 2, Stack Overflow features a voting system for questions and answers. This paper defines a "good answer" as the one with the highest positive score. Note that not every question has a good answer; those without are omitted according to the data selection criteria specified in this section. Moreover, in instances where several answers to a question have the same score, all such answers are regarded as good answers. The reason for choosing good answers is that they are expected to be correct answers and most likely to be used by other users. This section diligently delineates the systematic approach we have adopted to extract Python code snippets from good answers on Stack Overflow.

Data Selection Criteria: To conduct our empirical study, we obtained Python code snippets from the good answers on Stack Overflow.

When an original questioner posts a question on Stack Overflow, tags are added to describe the topic of the question, we use these tags to distinguish questions related to Python. Additionally, the content of the posted code snippet can differ from the initial posting due to edits over time. In light of these considerations, we focused on the most up-to-date data available to users in exploring the compilability of the current code snippets. Therefore, we employed the following five criteria to pinpoint the code snippets needed for our study:

- Code snippet from good answers to questions with at least one tag labeled as "python".
- Answer scores must be positive.
- Answers that share the same positive score are all recognized as good answers.
- Questions lacking answers are omitted.
- Posting data is the latest version.

Data Source: We use SOTorrent [16] to extract Python code snippets on Stack Overflow. SOTorrent is an open dataset based on the official Stack Overflow data dump, which provides access to the version history of Stack Overflow content at the level of whole question posts and individual text or code blocks. SOTorrent has been continuously updated with many versions since its creation. When we started our study, the latest version of SOTorrent was SOTorrent20_03 as of March 15, 2020[†].

Data Extraction: Applying the established criteria to the SOTorrent dataset, we successfully extracted 1,498,133 questions. These questions led to 2,161,905 answers, among which 976,807 were good answers. Acknowledging that an answer could contain multiple code snippets, we had a

[†]https://zenodo.org/record/3746061

considerable corpus of 1,376,571 code snippets.

Data Preprocessing: During the data extraction process, we noticed that code snippets stored in the SOTorrent database could introduce formatting issues that were not originally present, such as redundant or inconsistent indentation. These formatting issues might hinder subsequent analysis efforts, prompting us to preprocess the data. This step primarily involved rectifying redundant or inconsistent indentations within the code snippets, ensuring their readiness for future parsing and analysis. Moreover, we tackled instances where the code was framed in a read-eval-print loop (REPL) mode, further enhancing the quality and uniformity of our dataset.

3.2 Code Snippet Analysis

To deeply investigate the challenges of version compatibility in Python code snippets on Stack Overflow, we utilized Python interpreters to parse code snippets in the dataset we acquired. We deployed multiple Python interpreters, specifically, one major release of Python 2 (2.7) and four major releases of Python 3 (3.5 to 3.8). The selection of these versions was driven by the availability of the Python interpreters and the timeline of the SOTorrent release we employed, which incorporates Python 3.8 as the latest version. We excluded older versions due to the challenges in preparing an execution environment and newer versions released after the dataset.

In our study, we defined a code snippet as being interpretable (or compatible) with a specific Python version if the corresponding Python interpreter could successfully parse it. To determine this, we attempted to compile the snippet using Python's native `py_compile` module. If the compilation process was successful, the code snippet was categorized as interpretable (or pass) for that particular Python version. Conversely, if the compilation failed, the snippet was classified as uninterpretable (or fail) for that version. Within the confines of our study, the previously extracted code snippets from good answers were parsed using each chosen Python interpreter version. We recorded each code snippet's outcomes (pass/fail), constructing a comprehensive overview of their compatibility across different Python versions.

In this paper, only compile errors are considered. Cases that involve run-time errors or yield different results without errors are excluded from some considerations. Firstly, compile errors are a clear and direct indicator of compatibility, making the results clearer and more manageable. Secondly, run-time errors and differences in results involve more complex factors beyond the original intent and scope of this study. Lastly, it is difficult to set up the appropriate inputs to cause the desired run-time errors.

Moreover, in our experimental environment, because the compilation process does not involve the installation of any third-party libraries, code snippets that depend on specific libraries will result in compilation errors. This setup eliminates the impact of third-party libraries on the discussion of our experimental results.

4. Results

In this section, we explore the underlying motivation, the approach taken, and the results of our three research questions concerning the challenges of version compatibility within Python code snippets on Stack Overflow.

4.1 RQ1: How many Python code snippets have version compatibility issues in the good answers to Stack Overflow questions?

Motivation: This research question aims to measure the prevalence of version compatibility issues in Python code snippets from good answers on Stack Overflow. Given that the platform is a widely used resource for developers, understanding the scope of such issues is crucial. Code snippets with compatibility issues may prevent users from reusing the code. By quantifying the extent of these version compatibility issues, this research question offers actionable insights into the prevalence and nature of version-specific problems in Python code snippets. Such insights can guide Python programmers in adopting best practices for version compatibility, inform educators in structuring their coding curricula, and provide tool developers with a clearer understanding of common issues to address in future updates, aiding them in their respective endeavors.

Approach: In addressing this research question, we parsed these Python code snippets from the good answers on Stack Overflow using Python 2.7, Python 3.5, Python 3.6, Python 3.7, and Python 3.8. And we marked whether it is interpretable in each version as described in Section 3.2. In this way, we compiled a comprehensive catalog of Python versions compatible with each snippet.

Results: Analyzing the parsing results, we found that 440,456 (32.0%) code snippets failed to be parsed by all Python versions, i.e., these code snippets are uninterpretable for all Python versions. There are several possible reasons: 1) Our study didn't cover early Python versions like 2.0 or 3.0. 2) Some "Python" tagged answers may include non-Python code. 3) Programming errors, such as syntax issues, could be present. 4) Incomplete Python code snippets. 5) Third-party libraries need to be imported. In other words, they are not caused by the Python version upgrade we would like to investigate. In addition, we found that 755,699 (54.9%) code snippets could pass the parsing for all Python versions, i.e., these code snippets were interpretable for all Python versions. This may be the case because they are not using some features that would affect the compilability of code snippets vary between Python versions.

Finally, the remaining 180,416 (13.1%) code snippets are the parts that face compatibility issues across Python 2.7 and Python 3.x. This percentage indicates a significant challenge within the Python community, suggesting that a notable fraction of shared code on Stack Overflow may be incompatible with either Python 2 or 3. For these code snippets that face compatibility issues, we found that they

can be divided into the following three categories:

Pass Python 2.7, Fail for some Python 3:
The code snippet passes Python interpreter parsing for Python 2.7 and at least one Python interpreter parsing for Python 3, but not all Python 3 versions.

Pass Python 2.7, Fail for all Python 3:
The code snippet fails for any Python 3 Python interpreter parsing but passes Python 2.7 Python interpreter parsing.

Fail for Python 2.7, Pass Python 3 (all or some):
The code snippet fails for Python 2.7 Python interpreter parsing but passes all or some Python 3 Python interpreter parsing.

Of the above categories, the results are shown in Table 1, which provides a comprehensive overview of the situation of the parts of Python code snippets that face compatibility issues extracted from good answers on Stack Overflow. The "Overall percentage" column in the table indicates the proportion of the three categories of code snippets relative to all snippets extracted from the good answers on Stack Overflow. Notably, as indicated in the table, a considerable portion of these code snippets is exclusively interpretable by Python 2. This demonstrates the persistence of legacy Python 2 code within the developer ecosystem, underscoring the challenges in migrating to Python 3 despite its growing adoption and the cessation of official support for Python 2.

These results underline the extent of version compatibility issues within Python code snippets on Stack Overflow, pointing to a pressing need for attention and potential improvements in this area.

Answer to RQ1: About 13% of code snippets have version compatibility issues in the good answers to questions.

4.2 RQ2: How many of the code snippets interpretable only by Python 2 or only by Python 3 have Python version-specific identification?

Motivation: Within Stack Overflow, question tags and the answer texts are crucial navigational tools, guiding users toward content that aligns with their specific programming needs. Stack Overflow's tagging system includes "Python 2.x" and "Python 3.x" tags, allowing users to designate their questions as specifically pertaining to Python 2 or Python 3. Similarly, mentioning the Python version in the answer texts can guide users in understanding the applicability of the given code snippet. Correctly identifying a code snippet as exclusively interpretable by either Python 2 or Python 3, through tags or textual mentions (i.e., version-specific identification) is crucial. For instance, if a code snippet is solely compatible with Python 2 and is correctly identified as Python 2, it helps users avoid inadvertently using this snippet in a Python 3 context, and vice versa. Nevertheless, the effectiveness of this approach is contingent on the accurate application of these tags and textual mentions. In our research, we aim to determine how many Python code snippets can be interpreted exclusively by Python 2 or Python 3 are accurately identified with their respective version, either through tags or textual mentions in the corresponding answers. This analysis will assess the precision of Python version-specific identification on Stack Overflow and identify potential areas for enhancement to improve the user experience on the platform.

Approach: To address this research question, we turned to the parsing results of the code snippets obtained in RQ1. We focused on the categories "Fail for Python 2.7, Pass Python 3 (all or some)" and "Fail for all Python 3, Pass Python 2.7" as they represent code snippets exclusively compatible with Python 3 and Python 2, respectively. These identified snippets were then scrutinized for the presence of corresponding "Python 2.x" or "Python 3.x" tags, or textual mentions in their associated answers, to determine whether they were for Python 2 or Python 3. This enabled us to gauge the accuracy of version-specific identification in relation to the actual version compatibility of the code snippets.

Results: The findings from our study are presented in Table 2. The first row of the table shows that out of 47,712 code snippets that are solely compatible with Python 3, only 9,981 have been correctly tagged with "Python 3.x". Furthermore, 2,356 snippets mention the Python version in the text, of which 1,944 snippets mention the correct version in the text without the "Python 3.x" tag. This corresponds to only about 25% of Python 3-specific code snippets being properly labeled. This indicates a sizable gap in version-specific identification, highlighting that nearly 75% of these snippets lack the crucial information to inform users of their compatibility solely with Python 3.

Similarly, the second row reveals an even more glaring issue: among the 72,932 code snippets that are only interpretable by Python 2, merely 9,202 have been correctly tagged as "Python 2.x". Additionally, 2,418 snippets include a text mention of the Python version, and all of these are without a corresponding version tag. This amounts to a mere 16% of Python 2-specific snippets being appropriately labeled, leaving a staggering 84% of these snippets without clear version-specific indications.

The discrepancy between the 25% accurate identifying rate for Python 3 and the 16% for Python 2 demonstrates that Python 3-specific snippets are, on average, more likely to be correctly identified than their Python 2 counterparts on Stack Overflow.

These numbers reveal a critical issue with the current state of version-specific identification on Stack Overflow. Despite the platform's wide use and the community's dependency on it for reliable coding solutions, the lack of accurate version labeling potentially misleads users and hampers effective code reuse. In summary, the data call for immediate improvements in version-specific identification practices on the platform to better guide its vast user base, especially those working in mixed-version Python environments.

Table 1 Python code snippets facing compatibility issues in Stack Overflow

Categories	#Snippets	Overall percentage
Pass Python 2.7, Fail for some Python 3	59,772	4.3%
Pass Python 2.7, Fail for all Python 3	72,932	5.3%
Fail for Python 2, Pass Python 3 (all or some)	47,712	3.5%
Total	180,416	13.1%

Answer to RQ2: Only about 20% of code snippets interpretable only by Python 2 or only by Python 3 have accurate Python version-specific identification.

4.3 RQ3: How do users on Stack Overflow react and adapt to the introduction of new Python releases?

Motivation: The evolution of Python, with its frequent version updates, greatly influences the coding practices among its community, particularly regarding version compatibility. Given the importance of Stack Overflow as a knowledge base for Python users, understanding how users react and adapt to new Python releases can provide critical insights into the challenges and solutions associated with Python version compatibility.

Approach: To investigate the response of Stack Overflow users to each Python release, we sourced the release dates of each Python version directly from Python's official website. Our approach tracks the evolution of Python code snippets corresponding to each version since its release. This allows us to gauge the platform-wide response in coding practices to each Python version update. We defined a Python code snippet "responding" to a certain Python version based on two criteria:

- The posting or the most recent modification date of the code snippet must be after the release of that Python version. While this does not guarantee that the snippet will include newly introduced language features or syntax, it increases the likelihood that the snippet is influenced by or compatible with that version.
- The code snippet should be compatible with the concerned Python version and fail to compile for all preceding Python versions, implying that it utilizes features or syntax unique to the new version.

Our study primarily focuses on Python versions 3.5 through 3.8 for this research question. Additionally, we incorporate Python 2.7 from the Python 2 series for comparative insight.

Results: To precisely assess the response of Stack Overflow users to each Python version release, we categorized code snippets corresponding to each version on a daily basis. Figure 2 presents a stacked area percentage chart for various Python versions, where the release date for each Python version is marked with a yellow dashed line. The horizontal axis represents the code snippets' post or the latest modification date, while the vertical axis indicates the percentage of each Python version in all response code snippets.

Figure 2 shows that, except for Python 3.5 (which was not assessed against previous versions), the three other Python versions (3.6, 3.7, and 3.8) elicited a rapid and substantial response within days of their release. This result not only underscores the enthusiastic attitude of the Python developer community towards new versions but also manifests the swift acceptance and incorporation of unique features offered by new versions. Particularly noteworthy is the prominence of Python 3.6 among the community. Several factors may contribute to this phenomenon. Python 3.6 introduced compelling new features like f-strings, which made string formatting more intuitive. It also received performance optimizations and benefitted from broader third-party library support, making it a particularly attractive choice. Its longer-term support has also rendered it a stable option for ongoing projects. These factors likely fueled rapid community engagement with Python 3.6, emphasizing the Python ecosystem's dynamism and willingness to embrace new advancements. Furthermore, when investigating the response time of Python 3.8, we noticed a certain delay compared to other Python versions. This delay may have various reasons, including the complexity of new features, community response time, code example availability, and version migration speed. However, since this study focused on the analysis of code snippets and did not conduct a detailed study of the full-text content of posts, there is not enough data to comprehensive explanation for this delay. An extensive analysis of the full textual content of the posts is needed to fully understand the community's reaction to the release of the new Python version. Specifically, developers are eager to experiment with and integrate new capabilities into their code, emphasizing the language's adaptability and the community's openness to change.

Answer to RQ3: After the release of a new Python version, responses can generally be received on Stack Overflow on the same day.

5. Threats to Validity

Our research faces inherent limitations and threats to validity that are worth considering.

Limitations of Python code snippets: As elaborated in Section 4, approximately 32.0% of the Python code snippets we extracted from SOTorrent failed for Python interpreter parsing for all Python versions, implying they were incom-

Table 2 The results of version annotation for code snippets in the good answers on Stack Overflow, which are only compatible with either Python 2 or Python 3.

Categories	#Snippets	#Snippets (tagged)	#Snippets (text, untagged)	Percentage (version-specific)
Fail for Python 2, Pass Python 3 (all or some)	47,712	9,981	1,944 (2,356)	25%
Fail for all Python 3, Pass Python 2	72,932	9,202	2,418 (2,418)	16%

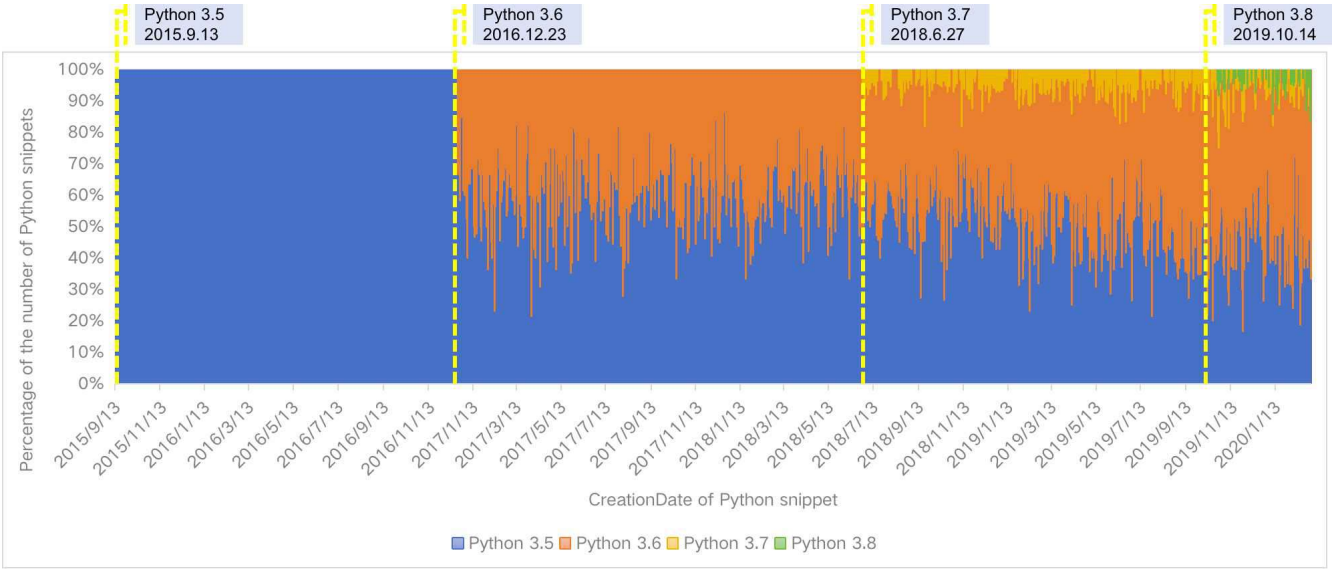


Fig. 2 Responses by Python version on Stack Overflow

patible with all Python versions. This subset of code snippets often included programming errors, pseudocode, and other issues unrelated to Python version upgrades, contributing to parsing failures. However, these snippets may also conceal features pertinent to Python version upgrades, obfuscated by errors arising from unrelated issues. Owing to technical and time constraints, we had to abandon the processing and examination of this portion of the code snippets in this study, potentially jeopardizing the validity of our results.

External validity: The generalizability of our findings presents a potential threat to external validity. Our study primarily revolved around Stack Overflow, and our insights may not extend to other Q&A platforms due to the variance in their mechanisms. Future research endeavors should encompass a broader range of Q&A platforms to mitigate this risk.

6. Related Work

Quality of Code Snippets on Q&A Platforms Within the scholarly discourse on Q&A platforms, the quality and reliability of shared code snippets have received widespread attention. The study by Wu et al. [6] underscored the critical need for improved mechanisms to evaluate the applicability of reused code, highlighting the role of such platforms in software development practices. Another investigation revealed a concerning trend of obsolete or license-violating code being circulated, underscoring a deficiency in the upkeep of these code fragments [3]. The ExampleCheck framework's exploration into API misuse within accepted answers on these forums further suggests a pressing need for more stringent solution validation processes [17]. The reliability of Java API-related snippets was specifically questioned. In the work of Zerouali et al. [18], the impermanent reliability of these snippets is brought to the fore, underscoring how library updates can swiftly invalidate previously dependable solutions. This reality underscores the ephemeral nature of code snippet accuracy. To counteract the widespread issue of reliance on outdated libraries in Java snippets on Stack Overflow, Zerouali et al. developed an automated approach for pinpointing Maven library version ranges. Their methodology illuminates the tendency within the developer community to lean on antiquated libraries and signals the need for a systematic reassessment of code snippet maintenance and the protocols for updating them. To address the challenge of outdated library usage in Java snippets on Stack Overflow, Zerouali et al. [18] developed a methodology for automatic identification of Maven library version ranges, highlighting the commonality of reliance on older libraries.

Compatibility Issues and Tools The evolution of programming languages and their corresponding updates are pivotal in influencing the longevity and relevance of code snippets shared on platforms like Stack Overflow. In this context, the work by Malloy et al. [14, 19] merits attention, as they delved into the challenges posed by Python version upgrades on code compatibility. Their contribution, a tool named *Py-*

Comply, has been developed specifically to detect the compatibility of Python code snippets with targeted versions of the Python language. Complementing this, our previous work presents *PyVerDetector*, a Chrome browser extension introduced in [11], which is ingeniously designed to tackle the issue of Python code snippet version compatibility on Stack Overflow. *PyVerDetector* aids users by automatically determining and indicating the Python version with which a given snippet of code is compatible, thus enhancing the overall utility and maintainability of code shared within the developer community.

7. Conclusion

In this paper, we conducted an empirical study investigating Python code snippet compatibility issues on Stack Overflow. The results are quite telling: Firstly, approximately 14% of code snippets in the good answers to questions exhibit version compatibility issues. This significant percentage underscores the need for more robust mechanisms or tools like PyVerDetector to assist users in accurately identifying the appropriate Python version for a given code snippet. Secondly, our findings reveal that only about 20% of code snippets interpretable exclusively by Python 2 or Python 3 have accurate Python version-specific identification. Finally, our study observes that new Python versions are quickly adopted and discussed on Stack Overflow, indicating an engaged community of users who readily respond and adapt to these changes.

These findings clearly show the Python version compatibility landscape on Stack Overflow and highlight key areas for potential improvement. We hope this research brings awareness to these issues and spurs further research and development in creating solutions that could substantially enhance the user experience on Q&A platforms like Stack Overflow. For further insights and a detailed look at the code snippets and other valuable information utilized in this study, interested readers and researchers can access our dataset on zenodo[†].

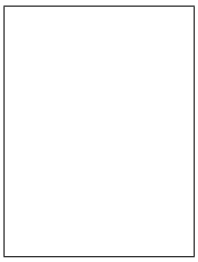
Acknowledgments

We would like to thank Davide Pizzolotto for his contributions to the tool development in our previous paper [11]. His expertise has been invaluable to our research.

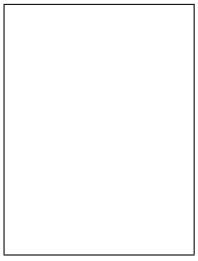
References

[1] S. Exchange, "Stack exchange data," 2023. Accessed on 2023-08.
[2] S. Baltes and S. Diehl, "Usage and attribution of stack overflow code snippets in github projects," Empirical Software Engineering, vol.24, no.3, pp.1259–1295, 2019.
[3] C. Ragkhitwetsagul, J. Krinke, M. Paixao, G. Bianco, and R. Oliveto, "Toxic code snippets on stack overflow," IEEE Transactions on Software Engineering, vol.47, no.3, pp.560–581, 2021.
[4] H. Zhang, S. Wang, T.H. Chen, Y. Zou, and A.E. Hassan, "An empirical study of obsolete answers on stack overflow," IEEE Transactions on Software Engineering, vol.47, no.4, pp.850–862, 2021.
[5] J. Zhou and R.J. Walker, "Api deprecation: a retrospective analysis and detection method for code examples on the web," Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE), pp.266–277, 2016.
[6] Y. Wu, S. Wang, C.P. Bezemer, and K. Inoue, "How do developers utilize source code from stack overflow?," Empirical Software Engineering, vol.24, no.2, pp.637–673, 2019.
[7] F. Fischer, K. Böttinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl, "Stack overflow considered harmful? the impact of copy&paste on android application security," 2017 IEEE Symposium on Security and Privacy (SP), pp.121–136, 2017.
[8] L. An, O. Mlouki, F. Khomh, and G. Antoniol, "Stack overflow: A code laundering platform?," 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), pp.283–293, 2017.
[9] M. Verdi, A. Sami, J. Akhondali, F. Khomh, G. Uddin, and A.K. Motlagh, "An empirical study of c++ vulnerabilities in crowd-sourced code examples," IEEE Transactions on Software Engineering, vol.48, no.5, pp.1497–1514, may 2022.
[10] S. Yang, T. Kanda, and K. Inoue, "The effect of python version upgrades on the compilability of code snippets posted on stack overflow," IPSJ SIG Technical Report, vol.2022-SE-211, no.28, pp.1–8, July 2022.
[11] S. Yang, T. Kanda, D. Pizzolotto, D.M. German, and Y. Higo, "Pyverdetector: A chrome extension detecting the python version of stack overflow code snippets," 2023 IEEE/ACM 31st International Conference on Program Comprehension (ICPC), pp.25–29, may 2023.
[12] R. Toal, R. Rivera, A. Schneider, and E. Choe, Programming Language Explorations, Chapman and Hall/CRC, 10 2016.
[13] "Sunsetting python 2." https://www.python.org/doc/sunset-python-2/, 2022. Accessed: 2022-06-27.
[14] B. Malloy and J. Power, "An empirical analysis of the transition from python 2 to python 3," Empirical Software Engineering, vol.24, 04 2019.
[15] B. Peterson and B. Cannon, "Backwards compatibility policy," PEP 387, Python Software Foundation, 2009.
[16] S. Baltes, C. Treude, and S. Diehl, "Sotorrent: Studying the origin, evolution, and usage of stack overflow code snippets," 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), pp.191–194, 2019.
[17] T. Zhang, G. Upadhyaya, A. Reinhardt, H. Rajan, and M. Kim, "Are code examples on an online q&a forum reliable? a study of api misuse on stack overflow," Proceedings of the 40th International Conference on Software Engineering (ICSE), p.886–896, 2018.
[18] A. Zerouali, C. Velázquez-Rodríguez, and C. De Roover, "Identifying versions of libraries used in stack overflow code snippets," 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), pp.341–345, 2021.
[19] B.A. Malloy and J.F. Power, "Quantifying the transition from python 2 to 3: An empirical study of python applications," 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pp.314–323, 2017.

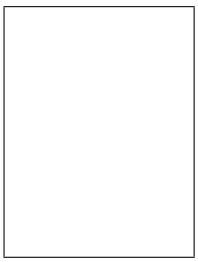
[†]https://zenodo.org/records/10790233



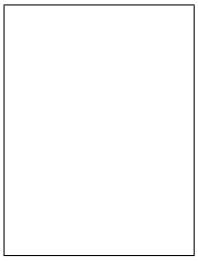
Shiyu Yang received a B.E. degree in Software Engineering from Dalian Jiaotong University in 2018 and an M.E. degree in Software Engineering from Osaka University in 2022. She is pursuing a Ph.D. degree at the Graduate School of Information Science and Technology, Osaka University. Her research interests include software engineering, especially software reuse, empirical approach, and program source code analysis. She is a member of the IEEE.



Tetsuya Kanda received his master's degree and Ph.D. degree in information science and technology from Osaka University in 2013 and 2016, respectively. At present, he is an associate professor at Osaka University. His ongoing research is centered on revealing software evolution and reuse by analyzing source code and development history.



Daniel M. German is currently a professor with the Department of Computer Science, University of Victoria. His research interests include the areas of mining software repositories, open-source software ecosystems, and the impact of intellectual property in software engineering.



Yoshiki Higo received his master's degree and Ph.D. degree in information science and technology from Osaka University in 2004 and 2006, respectively. At present, he is a professor at Osaka University. His research interests include mining software repositories, program analysis, and automated program repair.