Availability and Usage of Platform-Specific APIs: A First Empirical Study

Ricardo Job
IFPB
Cajazeiras, Brazil
ricardo.job@ifpb.edu.br

Andre Hora
Department of Computer Science, UFMG
Belo Horizonte, Brazil
andrehora@dcc.ufmg.br

ABSTRACT
A platform-specific API is an API implemented for a particular platform (e.g., operating system), therefore, it may not work on other platforms than the target one. In this paper, we propose a first empirical study to assess the availability and usage of platform-specific APIs. We analyze the platform-specific APIs provided by the Python Standard Library and mine their usage in 100 popular systems. We find that 21% of the Python Standard Library APIs are platform-specific and that 15% of the modules contain at least one. The platforms with the most availability restrictions are WASI (43.69%), Emscripten (43.64%), Unix (6.76%), and Windows (2.12%). Moreover, we find that platform-specific APIs are largely used in Python. We detect over 19K API usages in all 100 projects, in both production (52.6%) and test code (47.4%). We conclude by discussing practical implications for practitioners and researchers.

CCS CONCEPTS
• Software and its engineering → Software testing and debugging.

KEYWORDS
software testing, mining software repositories, test smells, Python

1 INTRODUCTION
Application Programming Interfaces (APIs) offer multiple benefits to users, such as feature reuse, productivity improvement, and reduction of development costs [9, 16, 17, 19]. A platform-specific API is an API implemented for a particular platform (e.g., operating system), therefore, it may not work on other platforms than the target one. For example, some APIs provided by the Python Standard Library have availability restrictions, such as the API os.listvolumes,1 which is available only for Windows, and the API os.chown,2 which is available only for Unix.

Nowadays, software systems are often tested on multiple platforms to increase quality and avoid bugs. This is supported by containers and modern CI/CD tools, such as GitHub Actions, which make it simple to test, for example, in the Linux, Windows, and macOS operating systems [7]. In this context, platform-specific APIs are problematic because they may break test suites on a certain platform if not properly used. Thus, a system that targets multiple platforms but uses platform-specific APIs should implement defensive code to ensure it will properly work on the desired platforms. For instance, Figure 1 presents usage examples of platform-specific APIs in three real-world projects: Tornado, Django, and Ray.

Figure 1: Usage examples of platform-specific APIs.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

MSR ’24, April 15–16, 2024, Lisbon, Portugal
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0587-8/24/04...$15.00
https://doi.org/10.1145/3643991.3644925

1 https://docs.python.org/3/library/os.html#os.listvolumes
2 https://docs.python.org/3/library/os.html#os.chown
3 https://github.com/tornadoweb/tornado/blob/f5df43f26bb4f9f519f260791754bc868defc9f24f/tornado/auth.py#L201
4 https://github.com/django/django/blob/311718e6b5f1f89ff794bba0c6d486c3410d8b/django/core/files/storage/filesystem.py#L139
We aim to assess real-world and relevant APIs and software systems, we are not aware of their real availability nor their possible impact on client systems. This knowledge can be used to better understand whether developers are aware of the API restrictions and support the creation of novel guidelines for using platform-specific APIs, like commonly adopted defensive practices. While APIs, in general, is a research topic broadly studied by prior literature (e.g., [2, 5, 8, 10–15, 18, 20–25]), to the best of our knowledge, the platform-specific APIs have never been deeply explored by the research community.

In this paper, we propose a first empirical study to assess the platform-specific APIs. Specifically, we explore (RQ1) the availability of platform-specific APIs provided by the Python Standard Library and (RQ2) their usage in 100 real-world systems. We find that 21% of the Python Standard Library APIs are platform-specific and that 15% of the modules contain at least one. The platforms with the most availability restrictions are WASI (43.6%), Emscripten (43.6%), Unix (6.76%), and Windows (2.12%). Moreover, platform-specific APIs are largely used in Python. We detect 19,288 usages of 683 platform-specific APIs in all 100 projects, in both production (52.6%) and test code (47.4%). Finally, we discuss implications for practitioners and researchers. Our results are publicly available [1].

Contributions. The contributions of the paper are threefold. First, we present the first empirical study to analyze the availability of platform-specific APIs. Second, we propose to assess the usage of platform-specific APIs in real-world systems. Third, we provide a set of implications for researchers and practitioners.

2 STUDY DESIGN

2.1 Case Study

We aim to assess real-world and relevant APIs and software systems. We selected Python due to its popularity and the rich software ecosystem with widely adopted projects to support web development, machine learning, and data analysis.

We analyze the availability of platform-specific APIs provided by Python Standard Library. This library is fundamental to building any Python application, providing features to handle text processing, file access, persistence, and networking, to name a few.

We also analyze the usage of the platform-specific APIs. For this purpose, we selected the top 100 most popular Python software systems hosted on GitHub sorted in descending order of the number of stars [3, 4]. We relied on the GitHub Search tool (GHS) [6] to find the 100 software projects. In this process, we took special care to filter out non-software projects, such as tutorials, examples, and code samples. On the median, the projects have 17,203 stars, 3,710 commits, and 8,795 APIs provided by the Python Standard Library.

Table 1: APIs provided by the Python Standard Library.

<table>
<thead>
<tr>
<th>API Level</th>
<th>Description</th>
<th>#</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Object method</td>
<td>2,940</td>
<td>33.43</td>
</tr>
<tr>
<td>Function</td>
<td>Module-level function</td>
<td>2,185</td>
<td>24.84</td>
</tr>
<tr>
<td>Data</td>
<td>Global data (variable or value)</td>
<td>1,438</td>
<td>16.35</td>
</tr>
<tr>
<td>Attribute</td>
<td>Object data attribute</td>
<td>1,073</td>
<td>12.20</td>
</tr>
<tr>
<td>Class</td>
<td>Class</td>
<td>942</td>
<td>10.71</td>
</tr>
<tr>
<td>Exception</td>
<td>Exception class</td>
<td>217</td>
<td>2.47</td>
</tr>
<tr>
<td>All</td>
<td></td>
<td>8,795</td>
<td>100.00</td>
</tr>
</tbody>
</table>

B. Detecting Platform-Specific APIs. APIs provided by the Python Standard Library may have availability notes indicating their supported and unsupported platforms. This is the standard way to indicate the API availability, for example, Availability: Windows means that an API is specific to Windows. According to the Python documentation, Availability: Unix indicates that the API is generally supported by macOS. We collected this information and classified the APIs with availability notes as platform-specific. Among the 8,795 APIs provided by the Python Standard Library, we detected 1,841 platform-specific APIs.

C. Assessing the usage of platform-specific APIs. We designed and implemented an AST-based tool to parse source code and detect the usage of platform-specific APIs. Specifically, the tool detects the presence of the selected platform-specific APIs and their location in the source code, in test or production code (a Python source file with the substring "test" on its path is classified as test, otherwise it is classified as production). We ran the proposed tool to detect platform-specific APIs in the 100 selected systems. In total, we detect over 19K usages of platform-specific APIs in all 100 projects.

2.3 Research Questions

2.3.1 RQ1: What is the availability of platform-specific APIs? First, we assess the availability of the platform-specific APIs provided by the Python Standard Library. We analyze the occurrence of platform-specific APIs by API level, module, and platform. Rationale: We

---

1. https://github.com/ray-project/ray/blob/fc98a5f286877ce7f6241961aca0c9127bee21ad/python/ray/tune/experiment/trial.py#L169
2. https://docs.python.org/3/library/os.html#os.listdrives
4. https://docs.python.org/3/library/intro.html#notes-on-availability
aim to better understand to what extent platform-specific APIs happen in Python Standard Library and identify the key entities. So far, it is not clear the extension of those APIs.

2.3.2 RQ2: What is the usage of platform-specific APIs? Next, we explore the usage of platform-specific APIs by client systems. Here, we present the usage in three distinct views: by system, API, and module. We also present the data according to their location in the source code (test or production code). 

Rationale: We aim to better understand to what extent platform-specific APIs are consumed by real-world Python systems. If this is common, it may bring to light novel discussions, for example, whether developers are aware of the API restrictions and how to mitigate possible problems caused by the usage of platform-specific APIs.

3 RESULTS

3.1 RQ1: Availability of Platform-Specific APIs

Table 2 summarizes the detected platform-specific APIs. Overall, we find 1,841 (20.93%) platform-specific APIs in the 8,795 APIs provided by the Python Standard Library. The most common API levels are methods (715 APIs), function (363 APIs), and data (308 APIs). The highest proportion happens in exceptions: 80 out of 217 (36.87%).

![Table 2: Platform-specific APIs by API level.](image)

Overall, we find platform-specific APIs in 51 out of 341 (15%) Python modules. Table 3 presents the modules with the most platform-specific APIs. The Python modules with the most platform-specific APIs are asyncio (300), os (220), and ssl (157).

![Table 3: Platform-specific APIs by module.](image)

3.2 RQ2: Usage of Platform-Specific APIs

Next, we analyze the usage of the platform-specific APIs in the 100 selected projects. Overall, we find 19,288 usages of 683 platform-specific APIs in all 100 projects. On the median, each project uses 79.5 platform-specific APIs (the first quartile is 23.3 and the third quartile is 184.3). Table 6 details the top-5 projects with the most platform-specific APIs. The Ray project has the highest usage (2,267), followed by Salt (2,098) and AIOHTTP (1,173).

Finding 3: Platform-specific APIs are largely used in Python. We find 19,288 usages of 683 platform-specific APIs in all 100 projects, in both production (52.6%) and test code (47.4%).

Table 5: Platforms with availability restrictions.

Finding 2: We find 17 different platforms with availability restrictions. The most frequent are WASI (43.69%), Emsscripten (43.64%), Unix (6.76%), and Windows (2.12%).

Finding 1: 21% of the APIs provided by the Python Standard Library are platform-specific. 15% of the modules contain at least one platform-specific API. The modules with the most platform-specific APIs are asyncio, os, and ssl.

Table 4 details the availability of the platform-specific APIs. Notice the APIs are most unavailable on the platforms WASI (43.69%) and Emsscripten (43.64%). We also find a significant number of APIs that are only available on certain operation systems: Unix (6.76%), Windows (2.12%), and Linux (1.94%). Overall, we find 17 different platforms with availability restrictions, as summarized in Table 5.

Table 4: Platform-specific APIs by platform.

Table 5: Platforms with availability restrictions.

Linux, Windows, macOS, Unix, AIX, Android
Emsscripten, WASI, pthreads, POSIX, Solaris, VxWorks
FreeBSD, BSD, DragonFlyBSD, NetBSD, OpenBSD

Finding 2: We find 17 different platforms with availability restrictions. The most frequent are WASI (43.69%), Emsscripten (43.64%), Unix (6.76%), and Windows (2.12%).

Finding 3: Platform-specific APIs are largely used in Python. We find 19,288 usages of 683 platform-specific APIs in all 100 projects, in both production (52.6%) and test code (47.4%).
Table 6: Usage of platform-specific APIs.

<table>
<thead>
<tr>
<th>Pos</th>
<th>Project</th>
<th>#</th>
<th>%</th>
<th>Test</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ray-project/ray</td>
<td>2,267</td>
<td>11.75</td>
<td>1,336</td>
<td>931</td>
</tr>
<tr>
<td>2</td>
<td>saltstack/salt</td>
<td>2,098</td>
<td>10.88</td>
<td>857</td>
<td>1,241</td>
</tr>
<tr>
<td>3</td>
<td>aio-libs/aiohttp</td>
<td>1,173</td>
<td>6.08</td>
<td>867</td>
<td>306</td>
</tr>
<tr>
<td>4</td>
<td>jina-ai/jina</td>
<td>1,162</td>
<td>6.02</td>
<td>889</td>
<td>273</td>
</tr>
<tr>
<td>5</td>
<td>ansible/ansible</td>
<td>888</td>
<td>4.60</td>
<td>308</td>
<td>580</td>
</tr>
<tr>
<td>All</td>
<td></td>
<td>19,288</td>
<td>100.00</td>
<td>9,136</td>
<td>10,152</td>
</tr>
</tbody>
</table>

Overall, the top-5 most used modules are subprocess (25.76%), asyncio (21.38%), threading (13.18%), os (10.40%), and socket (10.26%). Table 7 summarizes the most used platform-specific APIs. The most used API is asyncio.sleep (838 occurrences), which suspends the current task. Next, we have two APIs provided by the module subprocess. The API subprocess.Popen executes a child program in a new process and has 798 occurrences. The API subprocess.PIPE has 703 occurrences and it is a special value that indicates that a pipe should be opened. In Table 7, we also detail the frequency of the platform-specific APIs in test and production code. The usage of the platform-specific APIs is distinct among test and production code. For example, the API asyncio.sleep is mostly used in test code (83%), while the API os.getenv is mostly used in production code (74%). The top-3 modules in the test code are: asyncio, subprocess, and threading, while in the production code are: subprocess, asyncio, and os.

Finding 4: The most used platform-specific APIs are asyncio.sleep, subprocess.Popen, subprocess.PIPE, and os.getenv. However, there is a difference in usage in test and production code: asyncio.sleep is mostly used in tests, while os.getenv is mostly used in production code.

4 DISCUSSION AND IMPLICATIONS
Platform-specific APIs are widespread in the Python Standard Library but there is a lack of dedicated documentation. In RQ1, we found that 21% of the APIs provided by the Python Standard Library are platform-specific. Modules like asyncio, os, and ssl contain hundreds of platform-specific APIs. Moreover, we find 17 different platforms with availability restrictions, including mainstream OSs (Linux, Windows, macOS, and Unix), open-source Unix-like OSs (BSD, OpenBSD, FreeBSD, DragonFlyBSD, and NetBSD), proprietary OSs (Oracle Solaris, IBM AIX, and VxWorks), mobile OS (Android), WebAssembly platforms (Emscripten and WASI), and standards (POSIX and pthreads). Despite platform-specific APIs being widespread in the Python Standard Library, there is no documentation dedicated to their availability. The Python Standard Library documentation only provides a few notes regarding availability.14 Therefore, we recommend that dedicated documentation should be provided to better present the current state of availability and guide developers in charge of using platform-specific APIs. This kind of documentation could be auto-generated based on our dataset. Platform-specific APIs are largely used by Python systems but there is an absence of best practices and anti-patterns. In RQ2, we found over 19K usages of 683 platform-specific APIs in all 100 projects, in both production (52.6%) and test code (47.4%). We also detected that some APIs are more used in test code, while others are more adopted in production code. Given that platform-specific APIs are largely used by Python systems, developers would benefit from best practices to use them, both in test or production code. For example, the code presented in Figure 1 shows that developers may use multiple solutions (i.e., defensive coding) to call platform-specific APIs, however, it is not clear what are the possible solutions and most adopted ones. Thus, our results provide the basis for the development of novel qualitative studies on how to properly use platform-specific APIs, revealing best practices and the anti-patterns that should be avoided.

5 LIMITATION
This study focuses on the analysis of the platform-specific APIs provided by the Python Standard Library and their usage in popular projects hosted on GitHub. Therefore, our findings – as usual in empirical software engineering – may not be directly generalized to other projects or other programming languages. Further studies should be performed to better understand the platform-specific APIs in other software ecosystems.

6 RELATED WORK
APIs provide several benefits to users, such as feature reuse, productivity improvement, and reduction of development costs [9, 16, 17, 19]. API is a research topic broadly studied by prior literature, including API migration [2, 15], API deprecation [12, 18, 20–22, 25], and API evolution [5, 8, 10, 13, 14, 23, 24], to name a few. Some studies explore the compatibility issues caused by Android API evolution. In this case, researchers assess the challenges faced by Android developers to keep their applications working on multiple Android platforms [13, 14, 24]. To our knowledge, platform-specific APIs are not directly covered by the literature. Our study contributes to this research line by assessing the availability and usage of the platform-specific APIs provided by the Python Standard Library.

7 CONCLUSION AND FURTHER STEPS
In this paper, we provided an empirical study to assess the availability and usage of platform-specific APIs in Python. We analyzed the platform-specific APIs of the Python Standard Library and mined their usage in 100 popular systems. We found that 21% of the Python Standard Library APIs are platform-specific and 15% of the modules contain at least one. We also found 19,288 usages of platform-specific APIs that were detected across 100 projects, both in production (52.6%) and test code (47.4%). Lastly, we discussed practical implications for practitioners and researchers.

As future work, we plan to perform a qualitative analysis to better understand how platform-specific APIs are used in practice by developers and explore what are the best programming practices.
We also plan to analyze the availability and usage of platform-specific APIs in other programming languages and libraries.

ACKNOWLEDGMENT

This research is supported by CAPES, CNPq, and FAPEMIG.

REFERENCES


Table 7: Most used platform-specific APIs.

<table>
<thead>
<tr>
<th>Pos</th>
<th>API</th>
<th>API Level</th>
<th>All</th>
<th># %</th>
<th>Test</th>
<th># %</th>
<th>Production</th>
<th># %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>asyncio.sleep</td>
<td>function</td>
<td>838</td>
<td>4.35</td>
<td>698</td>
<td>83</td>
<td>140</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>subprocess.Popen</td>
<td>class</td>
<td>798</td>
<td>4.14</td>
<td>343</td>
<td>43</td>
<td>455</td>
<td>57</td>
</tr>
<tr>
<td>3</td>
<td>subprocess.PIPE</td>
<td>data</td>
<td>703</td>
<td>3.64</td>
<td>292</td>
<td>42</td>
<td>411</td>
<td>58</td>
</tr>
<tr>
<td>4</td>
<td>os.getenv</td>
<td>function</td>
<td>673</td>
<td>3.49</td>
<td>178</td>
<td>26</td>
<td>495</td>
<td>74</td>
</tr>
<tr>
<td>5</td>
<td>subprocess.check_output</td>
<td>function</td>
<td>615</td>
<td>3.19</td>
<td>318</td>
<td>52</td>
<td>297</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>All</td>
<td></td>
<td>19,288</td>
<td>100.00</td>
<td>9,136</td>
<td>100.00</td>
<td>10,152</td>
<td>100.00</td>
</tr>
</tbody>
</table>