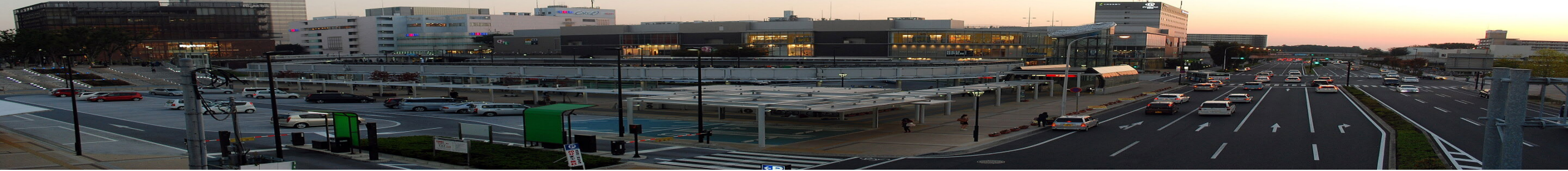


The 35th IEEE International Symposium on Software Reliability Engineering (ISSRE 2024)



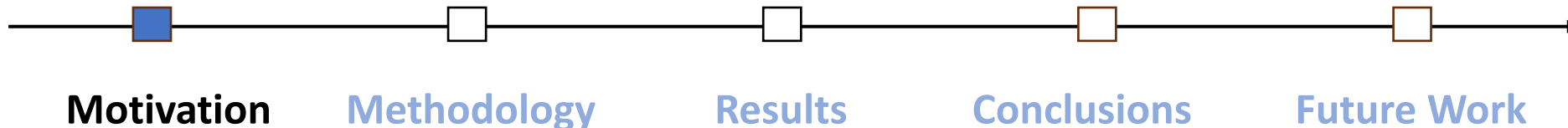
Coding Pitfalls: Demystifying the Potential API Compatibility Risk of Variadic Parameters in Python

Shuai Zhang, Gangqiang He, Guanping Xiao



Nanjing University of Aeronautics and Astronautics, China

ISSRE 2024, Tsukuba, Japan, October 29, 2024



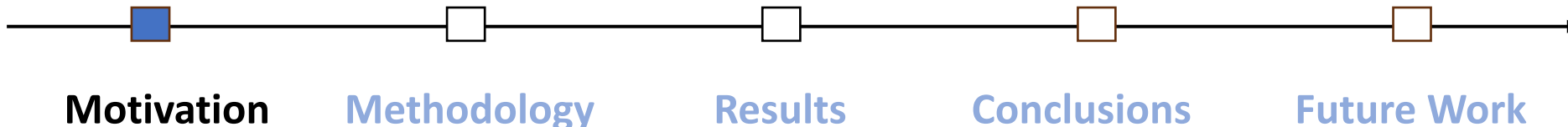
Python has been consistently ranked at the top of the TIOBE index and boasts a rich ecosystem.



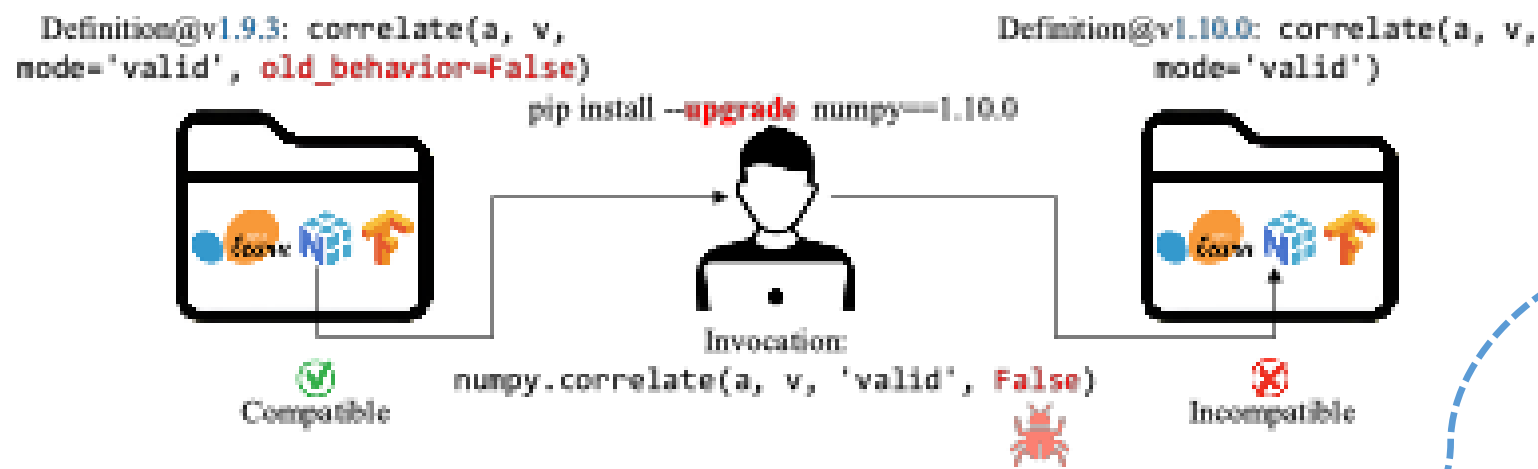
Oct 2024	Oct 2023	Change	Programming Language	Ratings	Change
1	1		 Python	21.90%	+7.08%
2	3	▲	 C++	11.60%	+0.93%
3	4	▲	 Java	10.51%	+1.59%
4	2	▼	 C	8.38%	-3.70%



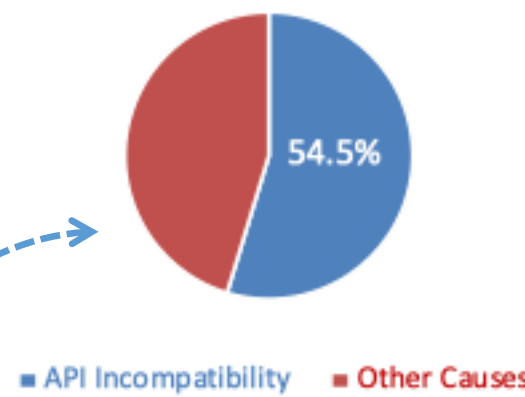
Python is used by a wide range of professionals, from data scientists to software engineers and academic researchers, enabling efficient development and innovation.



Evolution of Python Third-Party Library Versions Brings Compatibility Issues



Distribution of Root Causes of DL Compatibility Issues



J. Wang, G. Xiao, S. Zhang, H. Lei, Y. Liu, and Y. Sui, "Compatibility issues in deep learning systems: Problems and opportunities," in ESEC/FSE. ACM, 2023, pp. 476–488.



Python library developers often use variadic parameters to ensure backward compatibility and enhance API extensibility, ensuring existing code remains functional.

```
1 # API Definition in library SciPy 0.19.1
2 def pdist(X, metric = 'education', p = None, w = None, V = None, VI = None):
3     ...
```

<https://github.com/scipy/scipy/blob/v0.19.1/scipy/spatial/distance.py#L1128>

```
1 # API Definition in library SciPy 1.0.0
2 def pdist(X, metric = 'education', *args, **kwargs):
3     ...
```

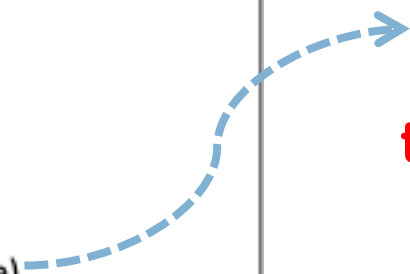
<https://github.com/scipy/scipy/blob/v1.0.0/scipy/spatial/distance.py#L1383>



```
1. # API Definitions in Tornado 6.0
2. # API HTTPRequest's parameter definition does not support callback and **kwargs
3. class HTTPRequest(object):
4.     def __init__(self, url, method, headers, body, auth_username, auth_password,
auth_mode, connect_timeout, request_timeout, if_modified_since, follow_redirects,
max_redirects, user_agent, use_gzip, network_interface, streaming_callback, header_callback,
prepare_curl_callback, proxy_host, proxy_port, proxy_username, proxy_password,
proxy_auth_mode, allow_nonstandard_methods, validate_cert, ca_certs, allow_ipv6, client_key,
client_cert, body_producer, expect_100_continue, decompress_response, ssl_options):
5.     ...
6.
7. # API fetch passes **kwargs internally to API HTTPRequest and removes callback since 6.0
8. def fetch(self, request, raise_error=True, **kwargs):
9.     ...
10.    if not isinstance(request, HTTPRequest):
11.        request = HTTPRequest(url=request, **kwargs)
12.    ...
13.
14. # Test Case
15. import tornado.httpclient
16. import tornado.ioloop
17. async def fetch_url():
18.     http_client = tornado.httpclient.AsyncHTTPClient()
19.     response = await http_client.fetch('http://example.com', callback=None)
20. tornado.ioloop.IOLoop.current().run_sync(fetch_url)
```

Improper use of variadic parameters can unintentionally cause compatibility issues.

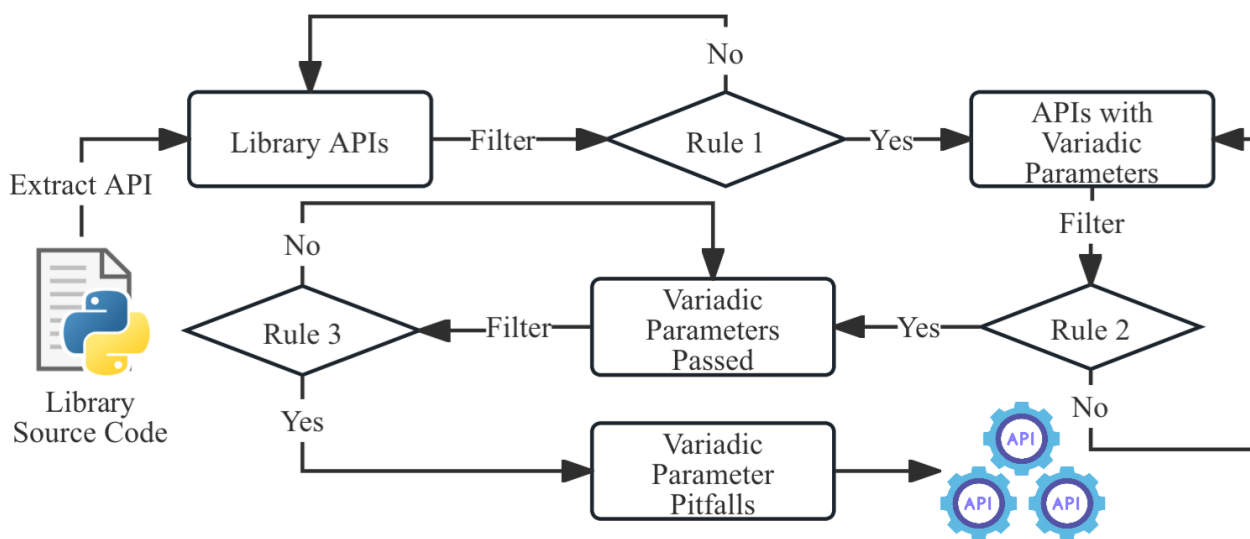
Variadic Parameter Pitfalls (VPPs—where an API with variadic parameters passes them to other APIs that don't support them.)





Contributions:

1. Performed the first study to explore variadic parameters in Python and proposed the definition of variadic parameter pitfalls.
2. Developed a tool called VPPDetector to detect variadic parameter pitfalls in Python library.
3. Provided practical development recommendations.



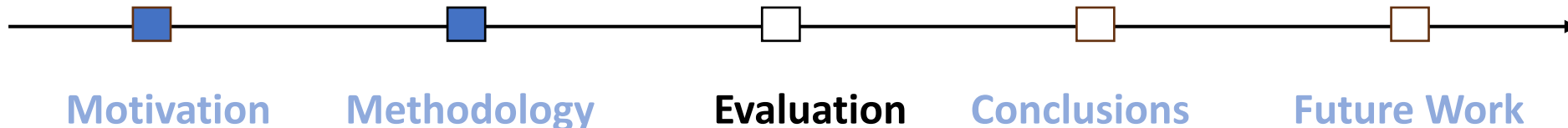
Overview of VPPDetector approach

<https://github.com/RAISE-Group/VPPDetector>

Rule1. The API's parameter definition includes variadic parameters, i.e., *args and/or **kwargs.

Rule2. Within the implementation of an API filtered by rule 1, there exists a case where variadic parameters are passed to other APIs.

Rule3. The called API, which receives variadic parameters (as per rule 2), does not define variadic parameters in its parameter definition.



Dataset Selection Criteria



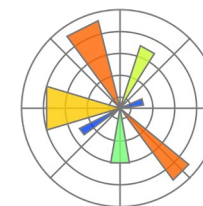
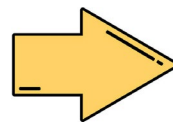
- GitHub Stars



- PyPI Download Counts

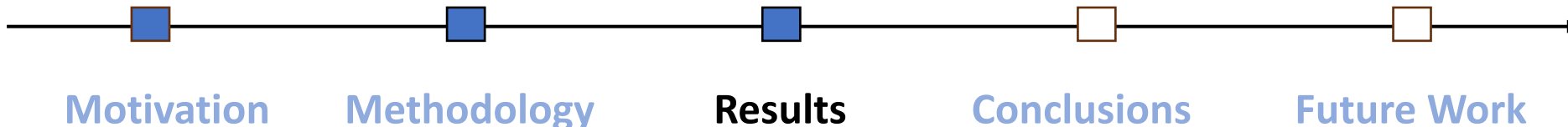


- API Documentation Detail



S. Zhang, G. Xiao, J. Wang, H. Lei, Y. Liu, and Z. Zheng, "Pcart: Automated repair of python api parameter compatibility issues," arXiv preprint arXiv:2406.03839, 2024.

33 Popular Python Libraries



Distribution of Variadic Parameter Usage in the Top 10 Libraries by Pitfall Rate

Library	Version Num	Avg. APIs with Variadic Params	Avg. Variadic Params Passed	Avg. Variadic Param Pitfalls	Avg. Variadic Param Pitfall Rate
Pillow	75	62	25	9	35.74%
Requests	107	43	39	7	18.12%
aiohttp	216	63	49	8	14.37%
Faker	292	21	17	2	10.23%
Gensim	48	55	46	3	8.58%
Click	53	30	27	2	6.73%
scikit-learn	42	178	119	7	6.44%
XGBoost	31	25	18	1	6.43%
SciPy	66	388	310	19	6.22%
TensorFlow	76	886	19	37	5.76%

$$\begin{aligned}
 & Avg.VariadicParamPitfallRate \\
 &= \frac{1}{n} \sum_{i=1}^n \frac{VariadicParamPitfalls_i}{VariadicParamsPassed_i}
 \end{aligned}$$



Distribution of Variadic Parameter Usage in the Top 10 Libraries by Pitfall Rate

Library	Version Num	Avg. APIs with Variadic Params	Avg. Variadic Params Passed	Avg. Variadic Param Pitfalls	Avg. Variadic Param Pitfall Rate
Pillow	75	62	25	9	35.74%
Requests	107	43	39	7	18.12%
aiohttp	216	63	49	8	14.37%
Faker	292	21	17	2	10.23%
Gensim	48	55	46	3	8.58%
Click	53	30	27	2	6.73%
scikit-learn	42	178	119	7	6.44%
XGBoost	31	25	18	1	6.43%
SciPy	66	388	310	19	6.22%
TensorFlow	76	886	19	37	5.76%

$$\begin{aligned}
 & Avg.VariadicParamPitfallRate \\
 &= \frac{1}{n} \sum_{i=1}^n \frac{VariadicParamPitfalls_i}{VariadicParamsPassed_i}
 \end{aligned}$$



Motivation

Methodology

Results

Conclusions

Future Work

Matplotlib
3.4.3



Matplotlib
3.5.0

zTest.py

```
1 '''
2 API Definition in matplotlib 3.4.3
3 def Shadow(patch, ox, oy, props, **kwargs)
4
5 API Definition in matplotlib 3.5.0
6 def Shadow(pathx, ox, oy, **kwargs)
7 '''
8
9 import matplotlib.pyplot as plt
10 import matplotlib.patches as patches
11 fig, ax = plt.subplots()
12 rect = patches.Rectangle((0.2, 0.2), 0.4, 0.4, linewidth=2, edgecolor='blue', facecolor='cyan')
13 ax.add_patch(rect)
14 shadow = patches.Shadow(patch=rect, ox=0.05, oy=-0.05, props = None)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
⊙ (matplotlib3.5.0) → ~ python zTest.py
Traceback (most recent call last):
  File "zTest.py", line 6, in <module>
    shadow = patches.Shadow(patch=rect, ox=0.05, oy=-0.05, props = None)
  File "/dataset/zhang/anaconda3/envs/matplotlib3.5.0/lib/python3.8/site-packages/matplotlib/patches.py", line 663, in __init__
    self.update({'facecolor': color, 'edgecolor': color, 'alpha': 0.5,
  File "/dataset/zhang/anaconda3/envs/matplotlib3.5.0/lib/python3.8/site-packages/matplotlib/artist.py", line 1064, in update
    raise AttributeError(f"{type(self).__name__!r} object "
AttributeError: 'Shadow' object has no property 'props'
```

<https://github.com/matplotlib/matplotlib/blob/v3.5.0/lib/matplotlib/patches.py#L637>



Conclusions

To mitigate VPPs, our recommendations are as follows:

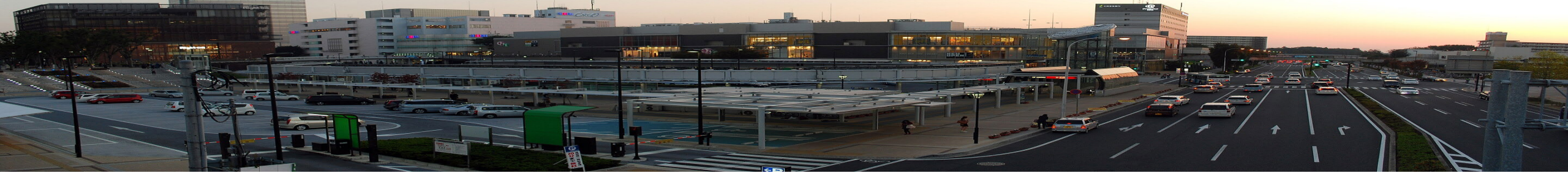
- **Recommendation 1.** When passing variadic parameters to an API, developers should ensure that the called API's parameter definition includes variadic parameters.
- **Recommendation 2.** Parse the variadic parameters first to identify those that will be used, and then pass only those parameters to an API, rather than simply passing variadic parameters.



Future Work

- Enhance the detection accuracy of VPPDetector.
 - VPPDetector currently supports detecting API variadic parameter pitfalls within a single file but cannot track them if the called API receiving variadic parameters is defined in another file.
- Analyze the patterns of variadic parameters and VPPs during Python library API evolution, as well as developers' intentions behind using these parameters.

The 35th IEEE International Symposium on Software Reliability Engineering (ISSRE 2024)



Thanks!

Q&A

Shuai Zhang

Email: shuaizhang@nuaa.edu.cn