e-Informatica Software Engineering Journal, Volume 17, Issue 1, 2023, pages: 230107, DOI: 10.37190/e-Inf230107

Empirical Study of the Evolution of Python Questions on Stack Overflow

Gopika Syam*, Sangeeta Lal*, Tao Chen**
*School of Computer Science and Mathematics, Keele University
**Department of Computer Science, Loughborough University
gopikasyam01@gmail.com, s.sangeeta@keele.ac.uk, t.t.chen@lboro.ac.uk

Abstract

Background: Python is a popular and easy-to-use programming language. It is constantly expanding, with new features and libraries being introduced daily for a broad range of applications. This dynamic expansion needs a robust support structure for developers to effectively utilise the language.
Aim: In this study we conduct an in-depth analysis focusing on several research topics to understand the theme of Python questions and identify the challenges that developers encounter, using the questions posted on Stack Overflow.
Method: We perform a quantitative and qualitative analysis of Python questions in Stack Overflow. Topic Modelling is also used to determine the most popular and difficult topics among developers.
Results: The findings of this study revealed a recent surge in questions about scientific computing libraries pandas and TensorFlow. Also, we observed that the discussion of Data Structures and Formats is more popular in the Python community, whereas areas such as Installation, Deployment, and IDE are still challenging.
Conclusion: This study can direct the research and development community to put more emphasis on tackling the actual issues that Python programmers are facing.

Keywords: Python programming, Software Development, Stack Overflow, Topic Modelling

1. Introduction

Python is a widely used, general-purpose programming language among developers. The language is constantly ranked as one of the most popular programming languages[1],[2]. Additionally, according to a recent Stack Overflow survey, the language surpassed Java, C, C++, and SQL to rank third most commonly used programming language[3]. It is a high-level, open-source, user-friendly language developed with a focus on improving code readability and development speed [1]. Python Programming language has undergone tremendous changes over years, and new language features are constantly being added to it [2].

[1]https://pypl.github.io/PYPL.html
[2]https://www.tiobe.com/tiobe-index/
[3]https://insights.stackoverflow.com/survey/2021#overview

Submitted: 21 Jan. 2023; Revised: 7 Jun. 2023; Accepted: 27 Jun. 2023; Available online: 4 Jul. 2023

Python has a large collection of libraries, frameworks, and integration support, making it incredibly useful for programmers. The language is constantly evolving, with language extensions, performance improvements, and core library updates. Because of these libraries and functionalities, it can be used in a wide variety of areas, making it popular among developers. However, developers are generally expected to be experts in these libraries and to maintain their skill set up to date, and keeping up with the pace of language development is often difficult for them. As a result, they frequently rely on programming question-and-answer sites like Stack Overflow[4] to ask for help on the challenges they encounter.

The development community commonly uses Stack Overflow, one of the major code-focused websites that is a part of the stack exchange network[5], to post challenges and exchange ideas. Stack Exchange is a network of question-and-answer websites that features discussions on various subjects in multiple fields. Some of the most well-known Stack Exchange websites include Stack Overflow[6], Super User[7], Server Fault[8], Ask Ubuntu[9], Data Base Administrator[10], Android User[11], Software Engineering[12] and others. Each of these websites features conversations on various topics and enables users to post questions and receive responses. Voting on questions and answers allows users to gain reputation points, which in turn grants them more privileges. Users with higher privileges can provide comments on questions and serve as moderators for certain sections of the website.

The most actively viewed sites in this network are Stack Overflow, Mathematics, Unix and Linux and Ask Ubuntu as of February 2023[13]. Stack Overflow has over 100 million monthly visits and has received around 21 million questions and 31 million answers to date[14], and it serves as a platform where developers can get expert advice on a variety of issues related to software development. With the help of this study, we are given a broad and realistic picture of developers' discussions about Python, helping us to learn real-world knowledge about the volume of these discussions and the topics that developers find both popular and difficult.

1.1. Motivation

Python is always expanding and evolving, with more developers adopting the language for diverse applications and new libraries and frameworks being released for a wide range of uses. Like other languages, the frameworks in Python are evolving constantly, mainly due to feature enhancements, performance improvements, and bug fixes. In spite of Python's popularity, researchers have not focused much attention on analyzing the trends and technologies of this language on Q&A websites like Stack Overflow [3]. Analyzing such Q&A platforms can provide insights that can help in making better tools and technologies for Python developers.

[4]https://stackoverflow.com/
[5]https://stackexchange.com/
[6]https://stackoverflow.com/
[7]https://superuser.com/
[8]https://serverfault.com/
[9]https://askubuntu.com/
[10]https://dba.stackexchange.com/
[11]https://android.stackexchange.com/
[12]https://softwareengineering.stackexchange.com/
[13]https://stackexchange.com/sites?view=list#traffic
[14]https://stackoverflow.co/

Python released a new version of the language in 2008 that is not backwards compatible, causing a transitional phase for Python developers. One major issue is that programs written in older versions of the language cannot be interpreted in the new version of the language without modifications. This is a challenge for developers because they must update or rewrite their applications to work with the latest language version. The study by Malloy and Power [4] discussed in detail the impact of this transition from Python 2 to Python 3 on the applications written in Python.

Table 1: Example of Python questions from Stack Overflow website

S. No	QId	Title
1.	30 667 525	ImportError: No module named sklearn.cross_validation
2.	38 987	How do I merge two dictionaries in a single expression in Python?

In addition, we can observe that developers frequently struggle with package name changes, which result in import and module not found errors, which is a roadblock in the early phases of development. Consider the Question Id (QId): 34 844 352[15] in Table 1, with 440 000 views where the user is experiencing "Import Error" with "sklearn.cross_validation" module. The issue in this question is because of the renaming and deprecation of the "cross_validation" sub-module to "model_selection" and it took approximately seven months to get an accepted answer. A lot of similar questions can be seen in Stack Overflow related to the deprecation of packages. Also, there are multiple Python questions in Stack Overflow, that took plenty of time to get a response. For example, the Question with Id: 38 987[16] in Table 1 has 2.9 million views and took almost 6 years to get an accepted answer. It can be challenging for programmers to create and maintain Python applications due to this constant evolution and dynamic characteristics, which can also affect the language's effectiveness, safety, and development time.

It is crucial to address these concerns and learn more about the actual difficulties faced by the Python development community in the real world. Some recent studies highlight the importance of analyzing issues faced by Python developers. A study by Zhang [5], investigates in detail the typical patterns and examples of issues faced in the evolution of Python APIs. This shows how Python framework evolution may result in compatibility issues in client applications. Seeing the importance of Python in the software development community, Widyasari et al. [6] recently proposed a Python bugs dataset.

The work presented in this paper is complementary to the above studies. In this work, we analyze Python questions from the Stack Overflow website to understand more about the evolution of the Python language and developers' challenges. We believe that the broad and diversified user base of the platform offers the ideal setting for investigating how various programmers use Python in their projects and what are some of the popular and challenging topics of discussion. This study will aid in establishing the extent to which current research in the Python language is deficient and understand the major causes for this by examining real-world difficulties thereby paving the way for future research in this field. We performed a multi-dimensional study of the Python questions on Stack Overflow. We analyzed the evolution of questions to understand the growth of the Python questions and answers over the period of time. We analyzed popular tags associated with

[15]https://stackoverflow.com/questions/30667525/importerror-no-module-named-sklearn-cross-validation/34844352#34844352

[16]https://stackoverflow.com/questions/38987/how-do-i-merge-two-dictionaries-in-a-single-expression

the Python questions to understand how interest in technologies changed over the period of time. Next, we identify popular topics (as well as their popularity and difficulty analysis) in Python questions to understand the common issues faced by Python developers. We believe that such analysis can be beneficial in revealing the most interesting and difficult areas of Python software development.

1.2. Goal and Research Questions

The objective of this study is to *conduct an in-depth analysis of the Python post in Stack Overflow to understand the evolution of Python and gain a better understanding of the challenges faced by developers*. We believe that the findings of this study can be utilised by practitioners, researchers, and educators in understanding the current state of Python in terms of challenges and trends and the extent to which the traditional viewpoint is different from real-world applications and issues. Furthermore, the study can also aid in identifying the areas where additional language resources and tools are required to support developers.

We begin by assessing the volume of discussion about Python and its distribution on the Stack Overflow platform to determine how popular the Python language is within the community. Then, user contributions and response times are investigated further to determine the availability of experts to address the challenges encountered by developers. We also assess the tags attached to the Python question and also identify the main topics of discussion. Finally, we categorise the topics based on several indicators to identify the popular and difficult topics. In conclusion, the following research questions are the focus of our study.

RQ 1: How have Python posts evolved throughout the years? The first research question (RQ 1) examines how the Python questions or posts have changed over time. This RQ is divided into four smaller RQs, each of which examines a distinct feature of Python posts that have been published on the Stack Overflow platform over time. The first sub-RQ (RQ 1.1) aims to gain insights into the growth of Python posts over the years by exploring the volume of questions and answers in the Stack Overflow platform. The second sub-RQ (RQ 1.2) tries to understand the amount of discussion in the platform and the satisfaction and difficulties of developers by exploring the questions with accepted answers, non-accepted answers, comment-only questions, and questions with no answers or comments The third sub-RQ (RQ 1.3) investigate how long it takes for Python questions to receive a response or an accepted response to comprehend the user's satisfaction and the availability of experts on the platform. The fourth sub-RQ (RQ 1.4) tries to understand the participation of the development community by determining the user's posting questions, answers, and accepted answers.

We observed an increase in the number of questions across the years, except in 2021, and a decrease in the percentage of questions receiving accepted answers. A huge satisfaction among the Python development community was also observed with most questions getting answered within the first hour of asking the questions.

RQ 2: How did the Python tags evolve over the years? The questions in Stack Overflow have tags assigned to them by the user. This RQ analyses these tags to determine the tags with the greatest number of questions. The findings showed that the popularity of data analytics and artificial intelligence is rising indicated by the increase in the number of questions in the "pandas" tag. It was also observed that Python developers still struggle with general programming questions as most of the question tags belonged to this category.

RQ 3: What are some of the main topics of Python discussed by Python developers? Every Python post on Stack Overflow reflects the difficulties programmers encounter when learning and creating Python applications. We extracted semantic concepts from these posts using topic modelling to better comprehend this. There were nine topics, including the general programming topic (e.g., algorithm implementation, object-oriented principles, general Python concepts, etc.), Scientific computing topic (data format handling, model implementations, big data libraries, etc.), Web development topic (mostly linked to Django and flask libraries), and others. We also explored some of the major discussion points and questions in each of these topics to understand more about the problems faced by developers in each of these topic categories.
RQ 4: What are the most popular and challenging topics discussed by developers in Stack Overflow? In this RQ we evaluated the Python topics identified in the previous RQ to determine the popular topic and the topic that is difficult to answer on Stack Overflow. Many metrics are used to categorise the topics into these categories as per prior research. The interesting observation in this RQ was that the topic of Installation, Deployment, and IDE was popular as well as challenging for developers.

In the context of this paper, the terms “Python questions” and “Python posts” are used interchangeably, indicating that they refer to the same concept. The remainder of this paper is organized as follows: Section 2 describes the study methodology used in our study and offers an in-depth discussion of the various steps involved. Section 3 provides the results and analysis for each of the research questions in this study, as well as information about the motivation, approach, and results for each. The major discussion points and insights this study offers to the research, educational, and development communities are presented in Section 4. Section 5 discusses several risks to validity, and, Section 6 discusses related works that use the Stack Overflow dataset to evaluate various aspects of programming languages, concerns encountered in Python software development, and software development challenges. Section 7 finishes the study with the conclusion and a brief discussion of future research.

2. Study methodology

Both quantitative and qualitative data analysis is used in this study to analyze different facets of Python programming [7]. This is also known as the mixed-method approach [7]. Using both of them offers valuable insights into the relationship between qualitative and quantitative data [7]. To be more explicit, the process employs several well-known statistical approaches, such as unsupervised machine learning and natural language processing to the dataset to post trends and patterns in Python posts. To support and correlate our quantitative findings, a manual analysis of a statistically significant sample of Python posts is conducted to learn more about the difficulties faced by Python developers. This practice is used by other researchers in the software engineering domain [8].

Figure 1 shows the methodology of our study. It consists of four main activities: (1) Obtaining the current Stack Overflow dataset, (2) Extracting the Python tags required for Python posts extraction, (3) Extracting Python posts or queries relating to the Python programming language (i.e., questions, answers, metadata), and (4) analyzing the metadata and textual content in the Python posts. In summary, we use the Stack Overflow dataset dump provided by *The Internet Archive*[17]. From this dataset, we extracted all the Python

[17]https://archive.org/download/stackexchange

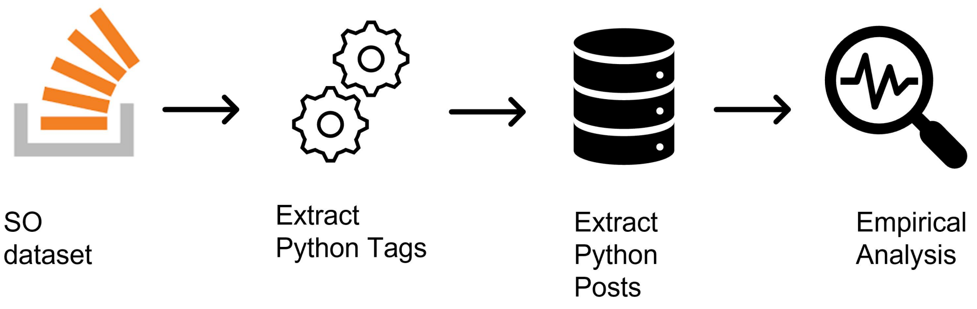


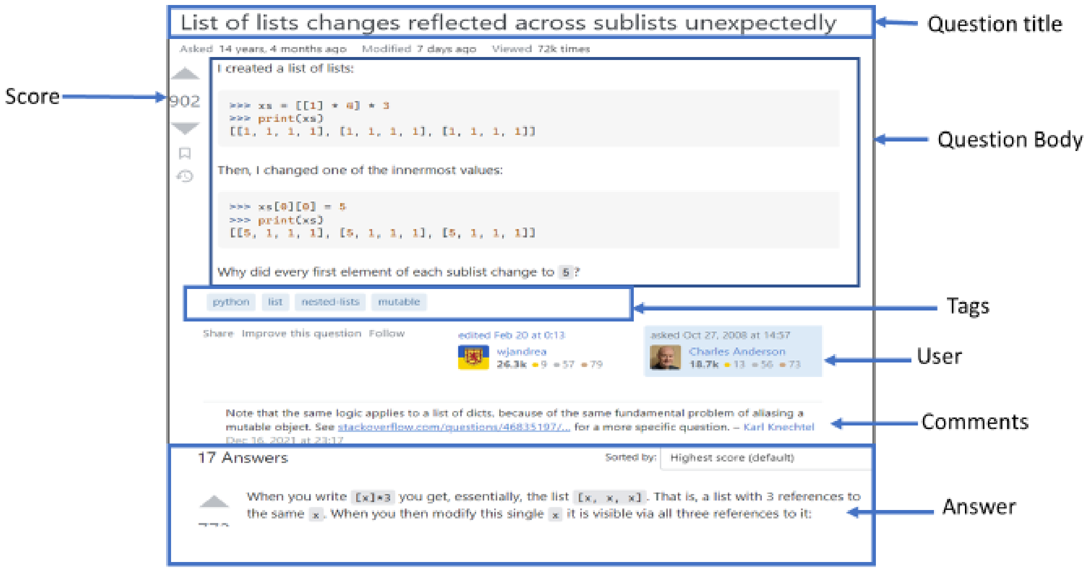
Figure 1: Overview of study methodology

posts (using tags of the post) and analyze these posts using both automated and manual processes to answer the research questions. The detailed steps related to each activity are mentioned in the following subsections.

2.1. Stack Overflow dataset extraction

The data for this study was gathered from Stack Overflow[18]. Stack Overflow is a website with over 14 million registered users that allows them to post questions and answers about many aspects of computer programming. It also allows users to edit their questions and answers, upvote or downvote questions and mark accepted answers. Users are also given badges and reputations based on the number of upvotes they receive and their meaningful contributions to the platform, which allows them to gain access to extra features like commenting and editing other people's posts[19]. The Stack Overflow data used for this study is obtained from *the internet archive*, which hosts the data of all Stack Exchange forums[20]. The platform has 8 files related to Stack Overflow: *badges.xml, comments.xml, postHistory.xml, postLinks.xml, post.xml, tags.xml, user.xml*, and *votes.xml*. The data was downloaded from this forum on June 2022. Figure 2 shows a snapshot of a post from Stack Overflow website[21]. Following is a brief description of the Stack Overflow dataset components:
Posts: There are two types of posts on Stack Overflow: (1) Questions and (2) Answers. The question post consists of a problem or challenge faced by the developer. A question has a title and body part. The title is plain text that gives the summary of the problem faced by the developer whereas the body part is a detailed description of the problem faced by the developer. The developer can put sample code, stack traces, etc. to provide more details about the problem. The Stack Overflow community can provide a response to the question. The response is called an "answer" post. There can be multiple answer posts associated with a single question post. The answer post can be of two types: Accepted answer or non-accepted answer. The accepted answer shows that the developer who asked the question considers this answer posts a solution to the question asked. Only the developer who asked the question can mark an answer as the accepted answer.
Comments: User can also provide a response to a post by posting comments on it. The users post comments on a post when they need clarification from the author of the post,

[18]https://stackoverflow.com/
[19]https://en.wikipedia.org/wiki/Stack_Overflow
[20]https://archive.org/download/stackexchange
[21]https://stackoverflow.com/questions/240178/list-of-lists-changes-reflected-across-sublists-unexpectedly



List of lists changes reflected across sublists unexpectedly

Asked 14 years, 4 months ago Modified 7 days ago Viewed 72k times

902

I created a list of lists:

```
>>> xs = [[1] * 4] * 3
>>> print(xs)
[[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]]
```

Then, I changed one of the innermost values:

```
>>> xs[0][0] = 5
>>> print(xs)
[[5, 1, 1, 1], [5, 1, 1, 1], [5, 1, 1, 1]]
```

Why did every first element of each sublist change to `5`?

python list nested-lists mutable

Share Improve this question Follow edited Feb 20 at 0:13 wjandrea 26.3k 9 57 79 asked Oct 27, 2008 at 14:57 Charles Anderson 18.7k 13 56 73

Note that the same logic applies to a list of dicts, because of the same fundamental problem of aliasing a mutable object. See stackoverflow.com/questions/46835197/... for a more specific question. – Karl Knechtel Dec 16, 2021 at 23:17

17 Answers Sorted by: Highest score (default)

When you write `[x]*3` you get, essentially, the list `[x, x, x]`. That is, a list with 3 references to the same `x`. When you then modify this single `x` it is visible via all three references to it:

Figure 2: A snapshot of a post from Stack Overflow

they want to leave constructive criticism that is helpful for the author to improve the post, or when they want to leave some minor information about the post.

Tags: A developer is required to associate one to five tags with the question (while creating any new question). These tags are useful in categorizing the questions. It helps the developer community on Stack Overflow to search questions that belong to a specific category. In Figure 2, the questions have four tags – "Python", "list", "nested-list", and "mutable"

Score: The Stack Overflow community can upvote or downvote a post based on their analysis of how useful a particular post is for the community. The *score* metric shows the total number of upvotes minus the total number of downvotes. In Figure 2 the score of the question is 902.

View Count: This metric shows how many times a question post is viewed. It is associated with only question posts. In Figure 2 the question is viewed 72k times.

Favorite count: This metric shows how many times the developers on the Stack Overflow community marked a post as their favourite. The metric is only associated with question posts.

User details: The post also has information about the user who asked the questions, answered the question, commented on the post and edited the post.

2.2. Python tags and posts extraction

The dataset contains information about all the posts on the Stack Overflow website. For this study, we focused our attention on discussions related to Python. Hence, we need to extract all the relevant posts for this study from the obtained *internet archive* dataset. Python is a popular programming language, therefore manually collecting Python questions from this vast dataset is challenging. So, to identify all the Python-related questions, we followed the steps mentioned below:

First, we created a list of all Python-related tags. We used data exchange explorer[22] and extracted the top 5000 questions with the highest score with the tag <python>. We used

[22]https://data.stackexchange.com/stackoverflow/query/edit/1600435#resultSets

the "score" account as a criterion because it is used by other researchers [8]. Additionally, the score count provides more authentic criteria for the usefulness of a question than the view count. Because when a user search for a question, they may view many questions before finding a question that resolves their problem. This increases the view count of many questions even when it does not provide useful answers. However, the score shows the number of upvotes that a question received. A user/moderator will only upvote a question they think is useful.

All the extracted posts were manually analyzed by the first author of the paper. The first author then created an initial list of Python tags by analyzing the various tags associated with these questions. This involved researching the tags and their relation to Python language. For example, the tag <eventlet> was further researched and its online documentation[23] helped to identify the relation of this tag to Python. During this process, There were some tags for which the first author was sure that they are related to Python. Such tags are put into "**Python-related**" tag list. The tags for which the first author had any confusion were put into another list called the "**doubtful tag**" list. The "**Python-related**" tag list was reviewed by the second author to identify whether these tags are Python related or not. During this review, all the disagreements between the first author and the second author are resolved through discussion. However, if the agreement is not reached for any tag, such tags are moved to the "doubtful-tag" list. Using this process, we identified 169 tags.

Second, we performed a second review of all the tags present in the "doubtful-tag" list. We noticed that most of the tags in this are either too generic or not popular. For each tag in the list, we extracted the top 30 highest-score question posts. The first author reviewed all 30 questions for each tag and either moved the tag to the "Python-related" tag list (if at least 90% of the 30 questions analyzed for a tag were related to Python) or kept it in the "doubtful-tag" list (in case of confusion). Both of these lists were again verified by the second author. We kept a tag in the "Python-related" tag only if both the first and the second author agreed on a tag to be Python-related. If there is a disagreement with respect to any tag it is moved to "doubtful-tag-list". For example, we analyze 30 question posts containing the tag <tuples>. We noticed that the out of these 30 question posts 21 questions were related to Python, 3 were related to C#, 4 were related to C++, and 1 was related to Java and typescript. So, this tag was kept in the "doubtful-tag" list. On the other hand, tags like <psycopg2> and <pep8> were added to the "Python-related" tag list after observing that all 30 questions manually checked were related to Python. Using this process, we added 5 more tags to the "Python-related" tag list and collected a total of **174** tags. These tags had 100% agreement between the first author and the second author because all the tags in which there was a disagreement were removed from this and moved to the "doubtful-tag-list" (see Table 2).

Third, previous studies on the Stack Overflow dataset mentioned the use of the title of questions (in addition to tags) for extracting the desired questions [8]. Hence, we explored titles of questions to extract more Python questions. For this, we extracted the 50 questions with the highest scores that have the word "Python" in the title and did not include any of the tags from our "Python-related" tag list. The content of these 50 questions was analyzed manually by the first author to determine their relevance to Python. Out of the 50 questions, 20 were related to Python while the remaining 30 were related to other programming languages. These questions' posts were further reviewed by the second

[23]https://eventlet.net/

Table 2: Final tags included in our Python-related tag list

List of Tags
Python, python-module, python-class, python-datamodel, pandas, python-3.x, python-packaging, matplotlib, python-internals, pip, python-import, pyenv, python-venv, pypi, mysql-python, zen-of-python, python-c-api, python-multithreading, python-2.x, python-unicode, setup.py, python-os, pyc, Django, django-models, psycopg2, python-wheel, ipython, python-typing, python-requests, cherrypie, cpython, django-queryset, pandas-loc, python-multiprocessing, python-2.5, python-2.7, python-zipfile, python-datetime, pandas-groupby, ironpython, pytest, python-3.6, flask, django-views, numpy-ndarray, numpy, python-imaging-library, python-2.6, beautifulsoup, timedelta, urllib2, scipy, seaborn, pythonpath, django-orm, pyyaml, python-nonlocal, python-unittest, pylint, pyflakes, pychecker, python-3.3, django-admin, pywin32, gunicorn, pytorch, mypy, pickle, django-shell, shutil, fnmatch, django-templates, wxpython, numpy-einsum, imaplib, flask-sqlalchemy, pygtk, pyspark, python-mock, flake8, django-media, django-authentication, python-asyncio, python-3.5, pycurl, python-3.4, django-signals, itertools, scikit-learn, scrapy, python-logging, eventlet, django-2.0, tkinter, openpyxl, pprint, django-testing, pyperclip, smtplib, popen, pypdf2, pypdf, configparser, django-validation, django-urls, python-sphinx, pycrypto, python-control, pymongo, django-manage.py, manage.py, pandas-explode, nonetype, django-migrations, paramiko, python-idle, python-dataclasses, gevent, django-rest-framework, nltk, pytz, difflib, distutils, py2exe, python-poetry, django-custom-manager, python-attrs, google-api-python-client, python-closures, pyarrow, django-aggregation, django-staticfiles, pyinstaller, django-class-based-views, python-2to3, mod-python, pydev, django-modeltranslation, imghdr, pyqt, magicmock, python-docx, pathlib, numpy-slicing, Theano, python-nose, django-q, django-celery, python-rq, fillna, pyqt4, django-1.10, python-click, h5py, pytables, pygame, jython, jpype, pycharm, jupyter-notebook, anaconda, jupyter, pypy, tensorflow, keras, conda, opencv, pep8, argparse, f-string, timeit, xlrd, mplcursors

author. We noticed that most of the questions were not related to Python but rather were demanding Python functionality in other programming languages. We opted not to include these questions in our final extracted questions because of the significant number of false positives. A sample from these questions is shown in Table 3.

Table 3: Illustrates the questions with a Python tag in the title, but are related to other languages

Programming language	Title	Question Id
R	Does R have an assert statement as in Python?	2 233 584
Java	Java plotting library like Python's matplotlib	958 806
C++	C++ algorithm like Python's "groupby"	12 335 860
Node.js	Node equivalent of "python -m SimpleHTTPServer"?	22 513 544
Javascript	Is there a javascript equivalent of Python's __getattr__ method?	1 529 496

Fourth, we looked at 30 questions with the highest score that had the word "Python" in their body but neither consist of any of the identified tags from our "Python-related" tag list nor the word "Python" in their title. Each of these questions was manually verified to confirm their relation to Python. Analysis revealed that all 30 questions containing the word "Python" in the body were not actually related to Python programming language. Nine of the questions were general inquiries about programming concepts, coding frameworks, keyboards, and other topics, while 21 questions pertained to other programming languages.

For instance, a list comprehension method in ruby, similar to the one in Python is requested in Question Id: 52 624[24]. Due to the enormous number of false positives, we decided to exclude the question with "Python" in the body alone from our analysis. A similar approach is followed by other researchers as well [8].

Table 4: Summary of collected data

Item	Value
Total number of Python questions	2 449 567
Question with Accepted Answers	1 231 993
Percentage of questions with accepted answers	50.29%
Timestamp of the first question	2008-08-02 03:35:56
Timestamp of the last question	2022-06-05 06:38:44
Total tags related to Python	174

Fifth, we extract all the question posts consisting of at least one tag from our "Python-related" tag list which has 174 tags. Using this process we extracted 2 449 567 questions. We observed that The number of questions extracted by our approach is comparable to the number of questions extracted in other studies [3]. The details about our extracted dataset are mentioned in Table 4.

2.3. Results analysis

We perform both quantitative and qualitative studies of Python posts. For quantitative analysis, we use tools such as SQL queries and code scripts to extract the information from the dataset. Whereas, for the qualitative study we perform manual analysis of the dataset. Our quantitative strategy includes running database queries and performing topic modelling analysis with an unsupervised machine learning algorithm. The qualitative approach, on the other hand, involves authors manually analysing statistically significant sample sets of data.

Table 5 shows the details about each research question and the methodology (quantitative or qualitative) we followed to answer it. For instance, in RQ 3, we first use a quantitative technique to identify the topics using LDA analysis, and then we use a qualitative strategy to analyse a sample of questions within each topic given by LDA to correctly label it. In Section 3 we provide more details about the research questions and the corresponding results obtained.

Table 5: An outline of RQs and the respective research approach that we used to answer each RQ

RQ Categories	RQ details	Research Approach
Questions Metadata	RQ 1	Quantitative
Tags	RQ 2	Qualitative and quantitative
Topics	RQ 3 RQ 4	Qualitative and quantitative

[24]https://stackoverflow.com/questions/310426/list-comprehension-in-ruby

3. Empirical study design and results

In this section, we provide details of all the research questions. We describe the motivation, approach, and findings of all the research questions.

3.1. RQ 1: How have Python posts evolved throughout the years?

The first research question (RQ 1) examines how the Python questions or posts have changed over time. This RQ consists of four sub-RQs. Each question focuses on a different aspect like the number of Python questions posted annually (RQ 1.1), the trend of successful, unsuccessful, ordinary, and comment-only questions (RQ 1.2), the time to get an accepted answer to Python questions (RQ 1.3), and the number of users who post questions and answers (RQ 1.4).

3.1.1. RQ 1.1: How did the Python post change over the years?

Motivation: This RQ determines the popularity of Python over time by analysing the number of Python questions posted between 2008 and 2022. This will be useful in understanding how the language has evolved over time and to what extent developers struggle with it. Determining the increasing or decreasing trend in the number of questions posted, for example, will help Stack Overflow, Kaggle, and other similar website teams to assign moderators based on this pattern. In this RQ, we use all the extracted Python questions as well as all the questions with accepted answers based on the identified set of tags. A question with an accepted answer typically indicates whether it was helpful. This will be beneficial for assessing Python developers' performance and satisfaction.
Approach: We group all the questions by year and saved them as a key-value pair in the Python dictionary. A line chart is drawn that shows the yearly trend of all the Python questions posted between 2008 and 2022. Another dictionary is used to store the set of questions with accepted answers based on the year. A bar graph is created that shows the yearly trend of total questions posted in the year and the number of questions having accepted answers.
Results: We extracted 2 449 567 Python questions from which 1 231 993 (50.29%) questions had an accepted answer. Table 4 provides the summary of the collected data. Figure 3 shows the yearly breakdown of the number of Python questions and Python questions with accepted answers. In this chart, for each year the blue bar is representing the total number of questions and the green bar is representing the questions with accepted answers. It is interesting to note that the number of questions posted each year is increasing (except for the years 2021 and 2022). This shows and complements the increase in the popularity of Python questions and further motivates this study.

A slight decline can be seen in the number of questions posted in the year 2021. There can be many reasons for that. One reason could be the COVID-19 pandemic. Because of this many sectors faced a decline in the number of jobs posted. Maybe developers working in Python also faced similar issues[25]. Georgiou [9] reported a decline in the number of questions posted during COVID. Another reason can be because of the rise of popularity

[25]https://www.zdnet.com/article/developer-jobs-demand-for-programming-language-python-falls-amid-pandemic/

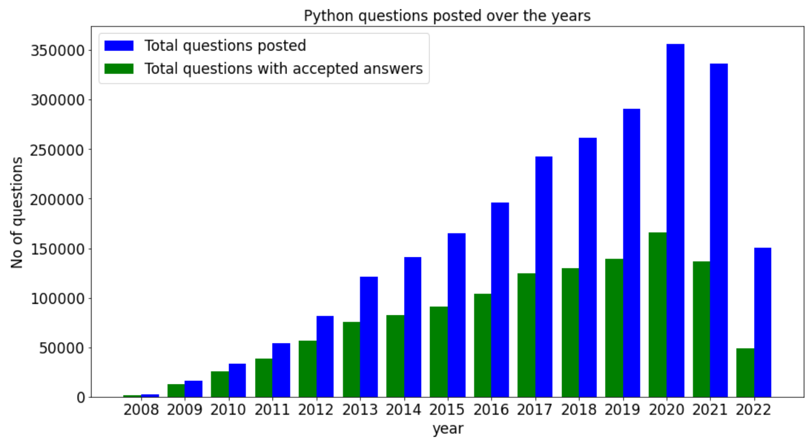


Figure 3: The number of total questions and questions with accepted answers over the years

of Java because of the advancement in Android Apps, SmartTV, Desktop Apps, and many more[26].

In our analysis, we consider data from 2008 to 2022. We notice a decreasing trend percentage of questions with accepted answers, i.e, 2008 has the highest percentage of accepted answers (81.49%) and 2022 has the lowest percentage of accepted answers (32.42%). We believe that this happened because older questions have a higher probability of getting accepted answers in later years.

3.1.2. RQ 1.2: What are the trends of successful, ordinary, comment-only, and unsuccessful questions?

Motivation: In this RQ we analyze successful, ordinary, comments-only, and unsuccessful questions. On Stack Overflow, users can post questions, answers, and comments on the Stack Overflow website. If the user (who asked the question) is satisfied with an answer, she can mark it as accepted. According to the study by Pinto [10] a question with an acceptable answer might be deemed a **successful** question because it indicates the user's satisfaction. An **ordinary** question, on the other hand, has answers but none of them are marked as accepted. An unsuccessful query is one that has no answers.

We frequently observe that the comments section under each question is also an excellent discussion area where users could get some tips or answers. Therefore, to decide the volume of discussion, we can analyse questions with comments and no answers. We called such questions as **comment-only** questions. Questions that have no answers or comments are considered **unsuccessful**. In this RQ, we aim to identify the trend of successful, ordinary, comments-only, and unsuccessful questions to have a better understanding of the satisfaction and difficulties faced by Python developers.

Approach: Out of all the questions, Python-related ones were filtered out based on the identified set of tags. The question with an accepted answer was extracted separately as a key-value pair into a Python dictionary based on their year. All questions with an "answercount" field greater than one and without accepted answers were then aggregated by year as "ordinary" questions. Out of the remaining questions, the ones with "commentcount"

[26]https://gettotext.com/python-java-and-sql-the-most-in-demand-programming-languages-in-2022/#:~:text=Perhaps%20the%20main%20reason%20for%20this%20precipitous%20drop,greatly%20influence%20the%20salary%20you%20can%20ask%20for

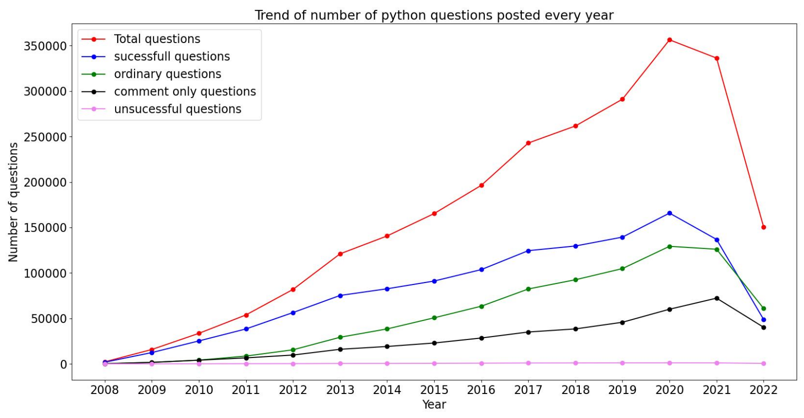


Figure 4: Question distribution over the years

greater than one was then grouped based on year into the dictionary. All the remaining questions were considered unsuccessful and were grouped together.

Results: Figure 4 shows the results obtained for this RQ. We extracted a total of 2 449 567 Python question posts. Out of these, 1 231 993 (50.29%) questions were successful (got an acceptable answer) and 807 735 (32.97%) questions were ordinary questions (i.e., had at least one answer), 400 760 (16.36%) questions are comment-only (i.e., only comments posted for these questions), and the remaining 9079 (0.37%) questions were unsuccessful (i.e., questions neither received any answers nor any comments). Figure 4 also shows that the number of successful questions outnumbers the ordinary, comment-only, and unsuccessful questions over the years (except in 2022). A decline in the year 2022 may be because we don't have complete data from the year 2022 as our experimental dataset was extracted in June 2022. The results show that a large number of questions received an accepted answer or at least one answer. This shows high satisfaction among Python developers. It is also interesting to show that the number of unsuccessful questions is very low over the years, i.e., $\leq$0.57%.

When comparing the results to the earlier study on Java [11] with Stack Overflow data, it can be seen that the total number of questions posted in Java is not always increasing and became more stable during 2013, 2014, followed by a fall in 2015. However, Python questions were consistently increasing during that time period, demonstrating how the language grew in popularity and eventually surpassed Java questions by 2017.

3.1.3. RQ 1.3: How much time it takes to get an accepted answer for Python questions?

Motivation: In this RQ, we analyze how much time it takes Python questions to get the first answer as well as the accepted answer. This is important as insight into the amount of time it takes to get an accepted answer or first response for Python questions can help in providing more evidence about the satisfaction among the Python community and support the results of prior RQs.

Approach To calculate the time for receiving the first answer and accepted answer, we collected a statistically significant sample of questions. We collected 16 529 questions from our experimental dataset with a 99% confidence level and 1% confidence interval [12, 13]. We created this sample using random sampling without replacement. We kept the confidence interval (i.e., margin of error) to 1%, this means that the true results might differ by ±1% from the results obtained. We selected a confidence level equal to 99%. The confidence level measures the uncertainty regarding how accurately a sample reflects the population being studied within a chosen confidence interval. A confidence level of 99% means that we are 99% certain that the results obtained using the sample match that of the actual population. We used the following formula for calculating the sample size[27]:

$$\text{Unlimited population } (n) = \frac{z^2 \times \hat{p}(1-\hat{p})}{\epsilon^2} \tag{1}$$

$$\text{Finite population } (n') = \frac{n}{1 + \frac{z^2 \times \hat{p}(1-\hat{p})}{\epsilon^2 N}} \tag{2}$$

Here, n – sample size obtained using an infinite population,
n' – sample size for finite population,
z is the z score,
ϵ is the margin of error or confidence interval,
N is the population size,
$\hat{p}$ is the population proportion.

Sample size calculation:

$$\text{Unlimited population } (n) = \frac{2.58^2 \times 0.5(1-0.5)}{0.01^2} = 16\,641 \tag{3}$$

$$\text{Finite population } (n') = \frac{16\,641}{1 + \frac{2.58^2 \times 0.5(1-05)}{0.01^2 \times 2\,449\,567}} = 16\,529 \tag{4}$$

In statistics, If the population is large often the information is inferred by studying a sample of that population. This is a well-known approach for large datasets and has been used by many other researchers in the software engineering domain [12, 13]. Hence, we believe that it is effective in our study as well. For each of these questions in the sample, we calculate the time for receiving the first answer and accepted answer with the help of the "CreationDate" field associated with each post in the Stack Overflow dataset.
Results: From the sample data, we looked at the time at which the answer post was created for each question. The findings showed that most of the responses to the Python questions occur within the first hour. The histogram on Figure 5 shows the distribution of time in hours from when a question is posted until the answer is obtained. This indicates that most of the Python questions receive an answer within a short period implying significant satisfaction within the developer community which further supports the developer satisfaction as identified in the prior RQ.

We also looked at the time at which an accepted answer is obtained. The histogram on Figure 6 shows the distribution of Python questions and reception of an accepted answer. The graph shows that most of the Python questions receive an accepted answer within the first hour. This indicates that there are experts in the Stack Overflow community to provide answers to the challenges faced by the Python software development community.

[27]https://www.calculator.net/sample-size-calculator.html

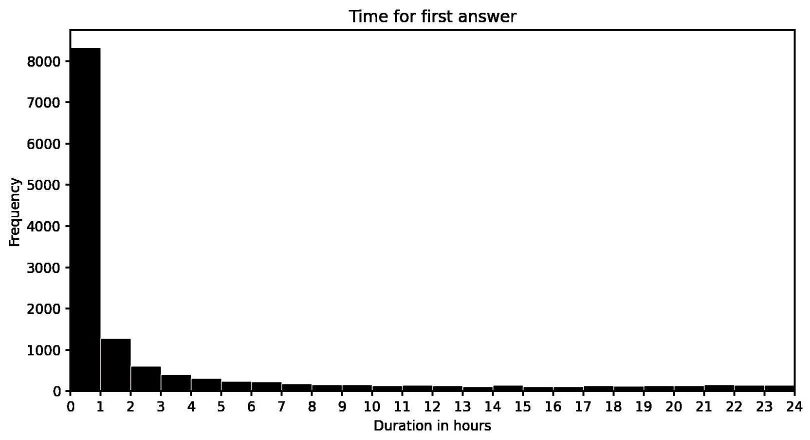


Figure 5: Time distribution of when a user asked a question and received the first answer

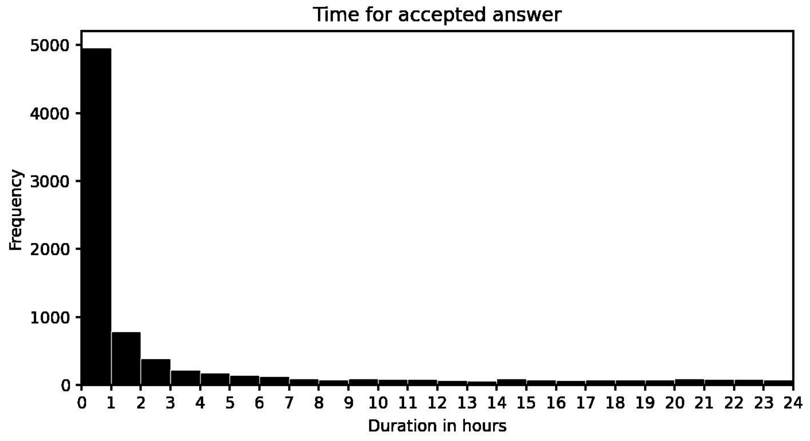


Figure 6: Time distribution of when a user asked a question and received an accepted answer

3.1.4. RQ 1.4: What is the distribution of Python questions and answers among developers?

Motivation: In this RQ, we analyze the Stack Overflow community participation in Python questions and answers. This analysis can provide insights into the size of the Python development community on Stack Overflow as well as whether only a few members of the community are responsible for asking and answering Python questions.
Approach: To determine the users posting questions, answers and accepted answers. we collected a statistically significant sample of questions. We collected 16 529 questions from our experimental dataset with a 99% confidence level and 1% confidence interval [12, 13]. To identify the user posting questions and answers we use the "OwnerUserId" field. We filter out users who posted questions, accepted answers, and non-accepted answers.
Results: The results showed that 15 361 unique users posted Python questions and 16 381 unique users posted Python answers in our sample data. Among these 16 381 users, 5419 unique users post accepted answers and 10 962 users post non-accepted answers. Table 6 shows a detailed distribution of the type of posts made by the unique users. Also, among all the answers posted, most of the users posted non-accepted answers.

Table 6: Distribution of Python question and answers among developers

Category	Count	Percentage
Users who post only questions	13 377	87.08%
Users who post questions and accepted answers	967	6.29%
Users who post questions and non accepted answers	1169	7.61%
Total users posting questions	15 361	
Users who post only accepted answers	4452	82.15%
Users who post accepted answer and questions	967	17.84%
Users who post accepted and non accepted answers	1484	27.38%
Total users posting accepted answers	5419	
Users who post only non accepted answers	8461	77.18%
Users who post non accepted answers and questions	1169	10.66%
Users who post non accepted and accepted answers	1484	13.53%
Total users posting non accepted answers	10 962	

Summary for RQ 1: We notice an increasing trend in the number of Python questions posted each year (except in the year 2021). We also notice a decreasing trend in the percentage of questions getting accepted answers. This finding shows a need for support for Python developers as the number of Python questions is increasing over the years. We notice a high satisfaction among the Python developers as only 0.37% of questions are unsuccessful and a large percentage of questions receive an accepted answer or first answer within the first hour after asking the question.

3.2. RQ 2: How did the Python tags evolve over the years?

Motivation: Tags are an essential part of Stack Overflow questions since they help organise them into distinct categories. Using tags to classify questions will also help to bring the attention of experts to each question[28]. Tags can also be used to filter the questions that we are interested in. In this RQ we aim to learn more about the various tags linked to Python questions. Analyzing the tags linked with questions will assist in determining the number of questions associated with each tag as well as the areas where Python developers experience the most difficulties.
Approach: This RQ was carried out in two stages. First, we extracted all of the tags associated with all of the retrieved Python questions and estimated the number of questions posted against each of them. This data was grouped yearly to determine how many questions were posted each year that use these tags. The top ten tags in this data were used to create a line chart to determine the trend of these tags over time.

Second, we used all tags identified as part of Section 2.2 and group them into different categories. Thirty questions were manually verified on the website by the first author for each of these tags in order to classify them into relevant groups. This was conducted to identify the complex and challenging areas of Python by determining the areas where developers are posting more questions. Table 7 provides a full list of the groups and the

[28]https://stackoverflow.help/en/articles/5611195-overview-of-tags

Table 7: Grouping detail of identified Python tags to extract questions

Group	Tags	% of Total Python Questions
Web development	django, django-models, cherrypy, django-queryset, flask, django-views, urllib2, django-orm, django-admin, gunicorn, django-shell, django-templates, flake8, django-media, django-authentication, django-signals, django-2.0, django-urls, django-manage.py, manage.py, django-migrations, django-rest-framework, django-custom-manager, django-aggregation, django-staticfiles, django-class-based-views, django-modeltranslation, django-q, django-1.10	17.8%
Scientific computing	pandas, pandas-loc, pandas-groupby, numpy-ndarray, numpy, python-imaging-library, beautifulsoup, scipy, pytorch, numpy-einsum, pyspark, scikit-learn, scrapy, pandas-explode, nltk, google-api-python-client, pyarrow, numpy-slicing, theano, fillna, h5py, pytables, tensorflow, keras, opencv, mplcursors	16%
General programming	python-module, python-class, python-datamodel, python-packaging, python-internals, python-c-api, python-typing, cpython, python-datetime, ironpython, timedelta, python-nonlocal, fnmatch, imaplib, itertools, pprint, pyperclip, smtplib, python-control, nonetype, python-dataclasses, pytz, difflib, python-attrs, python-closures, mod-python, python-click, jython, jpype, pypy, f-string, timeit	19.6%
OS, multithreading and multiprocessing	python-multithreading, python-os, python-multiprocessing, pyyaml, python-asyncio, popen, configparser, pathlib, django-celery, python-rq, argparse	6.75%
Installation, deployments and IDE	pip, python-import, pyenv, python-venv, pypi, setup.py, pyc, ipython, pythonpath, pywin32, python-idle, distutils, py2exe, python-poetry, pyinstaller, pydev, pycharm, jupyter-notebook, anaconda, jupyter, conda	12.9%
Testing and documentation	zen-of-python, pytest, python-unittest, pylint, pyflakes, pychecker, mypy, python-mock, python-logging, django-testing, django-validation, python-sphinx, imghdr, magicmock, python-docx, python-nose, pep8	10.4%
File handling and formats	python-unicode, python-wheel, python-zipfile, shutil, openpyxl, pypdf2, pypdf, xlrd	4.91%
Database and interaction	mysql-python, psycopg2, flask-sqlalchemy, pymongo	2.45%
Network and serialisationing	python-requests, pickle, pycurl, eventlet, paramiko, gevent	3.68%
Graphs and plotting	matplotlib, seaborn	1.23%
GUI support	wxpython, pygtk, tkinter, pyqt, pyqt4	3.07%
Others	pycrypto, pygame	1.23%

associated tags for each group. This data is used to plot a pie chart (i.e., Figure 7) to better comprehend the challenging areas for Python developers.

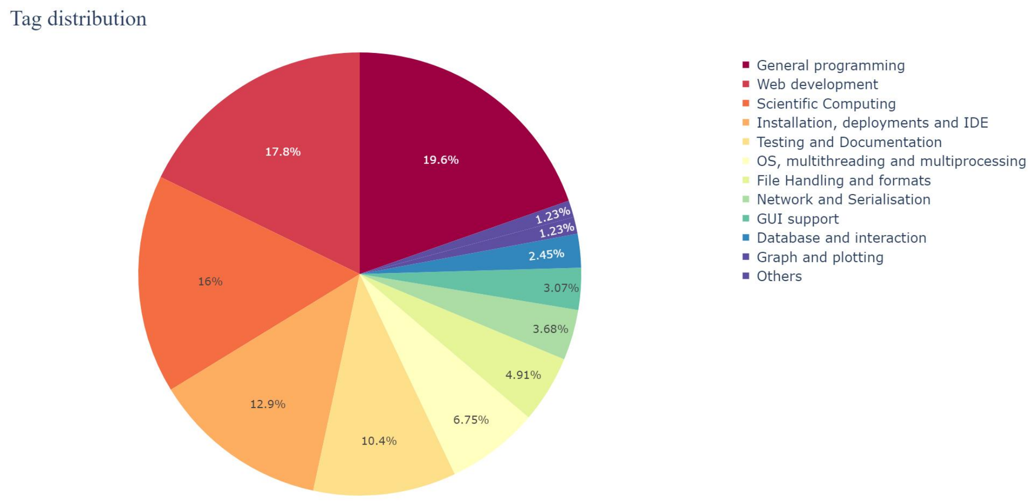


Figure 7: Tag distribution

Results: Our "Python-related" tag lists consist of 174 unique tags. We computed the total number of questions posted for each of these tags. The tag "Python" has the most questions submitted against it, with a value of 1 960 201 (80.02 per cent of all questions). Python-3.x, with a value of 314 799 (12.85%) appears the second most frequently in the dataset. This research focuses on Python and hence it is normal to have a high-frequency question consisting of Python tags. Here, we are more interested in identifying what other tags occur frequently in Python questions. Hence, we removed the tags like Python, Python-3.x, Python-2.x, Python-2.5, Python-2.7, Python-3.6, Python-2.6, Python-3.3, Python-3.5, Python-3.4, and Python-2to3 from our analysis (for this RQ), so that we can focus our analysis on other tags.

We computed the total number of questions posted for the rest of the tags and extracted the top 10 tags based on the count of the number of questions posted for each tag. We found that Django (291 137 posts or 11.88%), pandas (246 501 posts or 10.06%), NumPy (102 519 posts or 4.18%), dataframe (85 074 posts or 3.47%), TensorFlow (76 974 posts or 3.14%), list (68 603 posts or 2.8%), OpenCV (67 350 posts or 2.74%), matplotlib (65 393 posts or 2.66%), flask (50 403 posts or 2.05%), and dictionary (46 061 posts or 1.88%) are the top ten tags based on the number of questions. When examining the yearly top ten tags separately, it was discovered that these tags consistently appeared in the top ten spots for most of the years. The line chart in Figure 8 shows the yearly distribution of the number of questions posted against the top ten tags.

The graph shows a rapid rise in Django questions between 2008 and 2020, followed by a slight decrease from 2020 to 2021, which may be attributed to a drop in questions during that time as seen in the RQ 1. Tags such as list, OpenCV, dictionary, NumPy, matplotlib, and dataframe were steady from 2008 to 2010 and then showed an increase till 2021. However, from 2008 to 2010, tags like TensorFlow, Flask, and Pandas did not appear in the top 10. However, since 2010, there have been many more questions in pandas, and by 2020, they had surpassed the number of questions in Django. The study by McKinney [14] discusses in detail how "pandas" is one of the greatest tools for quickly handling and manipulating

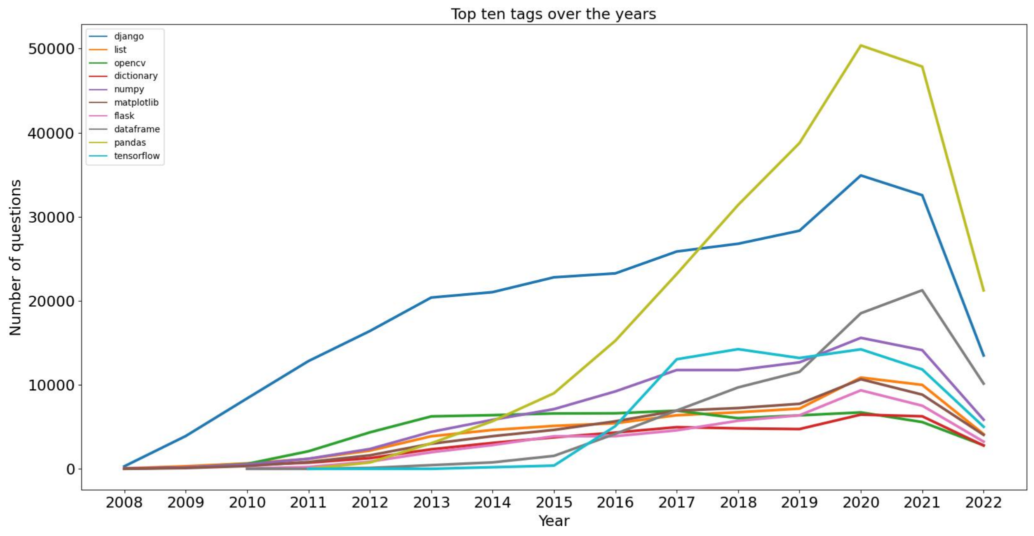


Figure 8: Top ten Python tags over the years

data, making it an essential tool for data analytics. The rise in popularity of data analytics and artificial intelligence since 2010 may be responsible for this spike in questions[29].

Questions on TensorFlow were consistent until 2011 when they suddenly increased in 2015, overtaking OpenCV, NumPy, and matplotlib to become the tag with the third-most questions posted in 2017. This may be the effect of the library being open-sourced by Google in 2015[30]. Additionally, the library is ranked first in the list of the most popular deep learning libraries in 2017 created by data incubators using data from GitHub, Stack Overflow, and Google searches[31]. Beginning in 2010, the top ten list also included the tag "flask", which continued to see a surge in question volume until 2020. The number of questions linked with the tag "dataframe" increased from 2017 to become the tag with the third most questions posted in 2021.

To ascertain which categories the tags belong to, we categorised them as shown in Table7. The majority of tags were linked to general programming (32 of 174 tags or 19.6%), and this included tags relating to data types (e.g., nonetype), classes (e.g., Python-dataclasses), time calculation and iteration tools (e.g., timedelta, itertools), and many more. Web development is the second category with the greatest amount of tags (29 of 174 tags, or 17.8%), with most of the tags being associated with the well-known Python web frameworks Django and Flask.

Scientific Computing is the third category (26 of 174 tags, or 16%), and it includes tags for computer vision and natural language processing (OpenCV, PyTorch, nltk), machine learning and artificial intelligence (scikit-learn), data manipulation and mathematical operations (pandas, NumPy). The next category, Installation, deployments, and IDE (21 of 174 tags or 12.9%), comprised tags for installation (pip, pyinstaller, etc.) and integrated development environments (IDEs), such as pycharm, jupyter-notebook, etc. The category Testing and Documentation (17 of 174 or 10.4%) featured tags for testing (such as pytest

[29]https://en.wikibooks.org/wiki/Data_Science:_An_Introduction/A_History_of_Data_Science#Chapter_Summary

[30]https://hub.packtpub.com/tensorflow-always-tops-machine-learning-artificial-intelligence-tool-surveys/

[31]https://www.thedataincubator.com/blog/2017/10/12/ranking-popular-deep-learning-libraries-for-data-science/

and Python-Unittest), error checking (such as Pyflakes and PyChecker), and documentation (such as Python-DocX and Zen-of-Python).

OS, multithreading, and multiprocessing tags (11 of 174 tags or 6.75%) include Python-asyncio, Python-os, etc., while File Handling and formats tags (8 of 174 tags or 4.91%) include pypdf, openpyxl, and others. Eventlet, Paramiko and other tags are categorized in the Network and Serialization category (6 of 174 tags, or 3.68%), whereas tkinter, pyqt, and other tags are found in the GUI Support category (5 of 174 tags, or 3.07%), and mysql-python, pymongo, and other tags are found in the Database and Interaction category (4 of 174 tags, or 2.45%). The least amount of tags (2 of 174 tags, or 1.23%) was found in the categories Graph and plotting with tags matplotlib and seaborn, and Others with the tags pycrypto and pygame.

Summary for RQ 2: The findings showed that the popularity of data analytics is rising indicated by the increase in the number of questions in the “pandas” tag. It was also observed that Python developers still struggle with general programming questions as most of the question tags belonged to this category.

3.3. RQ 3: What are some of the main topics of Python discussed by Python developers?

Motivation: Stack Overflow allows developers to use natural language to describe their challenges or propose a solution to problems. Every language has features and capabilities that are unique to it. For instance, Python developers must be proficient in a wide range of Python libraries, including Django, pandas, timedelta, and many others. As a result, depending on the features of each language and its applications, the issues that developers discuss may differ substantially. For this reason, it’s critical to comprehend the main Python discussion points and the challenging areas that developers face daily.

In this RQ, we use the Stack Overflow dataset to identify the important Python topics that are commonly discussed by Python developers. Identifying the primary topics of discussion in the Python community will aid in determining the most interesting areas for Python developers. This will contribute to the enhancement of Python’s tools, design, and documentation. The information from this RQ will also be valuable to Python-related publications because it identifies areas in which the Python community is most interested and has many issues. Researchers can use this to find possible study areas in the future.
Approach: To answer this RQ, we use **topic modelling** to determine the major Python discussion topics. Using the word distributions in the document collection, topic modelling is an unsupervised machine learning algorithm, that attempts to find any hidden patterns of word frequency. Based on the patterns, it provides a collection of topics that contain collections of words that co-occur in the document set [15].

We use the following methods to identify the main topics discussed by the Python community. **First**, we extract the entire Python posts from Stack Overflow and use several pre-processing techniques to extract relevant keywords. **Second**, topic modelling approaches are used to extract the topics from the pre-processed data. **Third**, to learn more about the context around the discovered topics, the extracted topics and a sample of questions are analysed, and a name is given to it based on the context. Figure 9 shows the approach followed for topic modelling. Following is the detailed description of each step.

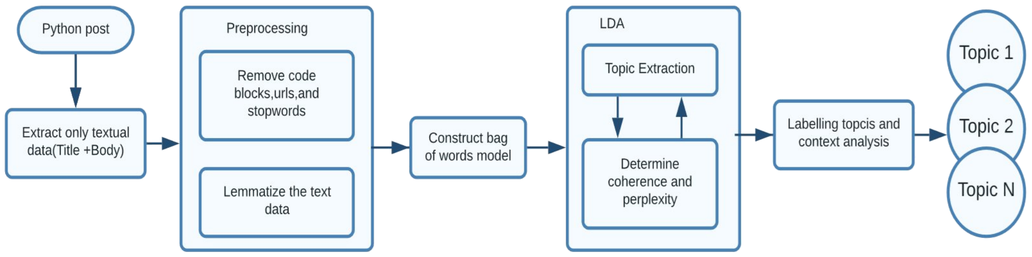

Python post
Extract only textual data(Title +Body)
Preprocessing
Remove code blocks,urls,and stopwords
Lemmatize the text data
Construct bag of words model
LDA
Topic Extraction
Determine coherence and perplexity
Labelling topcis and context analysis
Topic 1
Topic 2
Topic N

Figure 9: Topic modelling approach

1. Data Extraction and Pre-processing: The Stack Overflow dataset has a large number of questions and hence, we created a random sample of question posts with a 99% confidence level and a 1% confidence interval. We extracted 16 529 question posts. We extracted both title and body of these questions for analysis. In the initial round of pre-processing, we eliminate all code snippets from the text. As the code blocks do not add meaningful words for topic modelling, this step is critical to remove extraneous noise from the data. We exclude all URLs and HTML tags from the textual data because these are not required to generate keywords.

The next step is to eliminate any punctuation and expand any word contractions (for example, a change I'm to I am). Multiple white spaces, numerals, and words that are less than three characters in length are then eliminated from the text. Then we exclude the English stop words offered by NLTK as well as a few custom stop words like "thanks", "question", "answer", etc. Then, to assist with topic modelling, the words are lemmatized to their root form using the NLP Python module spaCy [16] Then, a bag of word models is constructed for Topic modelling to efficiently determine the word occurrences in all of the questions [17].

2. Topic Modelling: The topic model is a statistical model that is commonly used in natural language processing to determine the primary topics in a document collection[32]. A popular algorithm for topic modelling Latent Dirichlet Allocation (LDA) [18] is used to answer this research question. LDA is a probabilistic model created from a set of documents, where each document is represented as a mix of topics, and each topic is characterized by the word distribution. Using LDA for topic modelling can be seen in multiple prior research which use Stack Overflow dataset in a similar context and has proved to produce effective results [3, 8, 13, 19, 20].

The LDA algorithm receives the corpus prepared using the bag-of-words technique. The number of topics, or "k", is another key parameter needed by the LDA algorithm. Determining an ideal k value is critical in LDA since a low number provides a broad selection of topics while a high value provides more detailed topics that are essentially noise. To determine the ideal k value, we considered a parameter called topic coherence. The semantic relationship between the terms in a topic is measured by the topic's coherence, and a greater coherence value produces better results [21]. So, we ran the LDA iteratively with the value of k ranging from 2 to 100 and with a step size of two. The coherence values for each cycle were examined, to find the ideal k value. We identified k value 20 as giving the best results.

3. Determining the topics: After the model with the optimal coherence score has been determined, research is conducted to ascertain each LDA topic's nature and meaning. As each word in the corpus is given a probability of belonging to a certain topic by the

[32]https://en.wikipedia.org/wiki/Topic_model

LDA method, the distribution of topics across words is employed for topic labelling. To comprehend the topic, the top word with the highest probability is utilized. Each document is also assigned a probability that indicates how closely it relates to the topic. A sample of these documents was manually identified to better support the topic determination. These techniques helped in assigning an appropriate label to each topic.

Results: The LDA algorithm does not provide meaningful topic names for the results that it generates. Hence, two of the authors perform a manual examination of these 20 topics and associated keywords to find meaningful topic names. For each topic, we looked at the terms that are present in the topic and performed a discussion to assign a meaningful name. During this analysis, we notice that several topics contained similar words with a high probability, thus they were merged to better represent the topic. Using this process, we finally left with nine topics: *general programming, scientific computing, web development, file handling and formats, data structures and formats, graph and plotting, operating systems, multi-threading and multiprocessing, installation, IDE and deployments, and GUI support.* The nine topics, the number of documents that correspond to each topic, and the top keywords linked with each topic are all displayed in Table 8.

Examining the top words in each topic was beneficial in identifying the topic name. However, more in-depth analysis is needed to determine the rationale or the problems

Table 8: Topics generated by LDA with the frequency of occurrence and relevant keywords

Topic	Question count	Percentage %	Top keywords
General programming	3826	23.14%	input, output, time, differ, result, length, function, binari, random, variabl, random, class, local, default, method, argument, multipl, actual, refer, check
Scientific computing	2602	15.74%	panda, datafram, model, test, train, dataset, tensorflow, keras, predict, fit, featur, pyspark, spark, accuraci, scrapi
Web development	2478	14.99%	django, page, form, view, html, server, web, flask, post, applic, load, http, site, app, respon
Data Structures and Formats	2421	14.64%	arrai, list, vector, object, float, integ, data, iter, json, text, dictionari, tupl, dict, type
Installation, deployments, and IDE	1396	8.44%	instal, import, modul, version, packag, pip, path, environ, librari pycharm, configur
OS, multithreading, and multiprocessing	1172	7.09%	process, start, task, script, command, job, thread, execut, port, worker, client, socket
Graphs and plotting	1099	6.64%	imag, plot, matplotlib, background, color, label, figur, displai
File Handling and formats	922	5.57%	file, line, open, csv, format, directori, zip
GUI Support	613	3.7%	item, button, click, window, tkinter, press, option, page, menu

encountered by the developers. Hence, two of the authors perform a manual analysis of a statistically significant set of questions. We use a confidence level of 95% and an interval of 5%. Using these parameters, we created a sample size of 376 posts. Two of the authors annotated the dataset with their understanding of the rationale behind each topic. Once, completed both of the authors exchange the annotated dataset with each other. During the review, the conflict was resolved using a discussion between the annotator and the reviewer. There were 17 posts for which no agreement was received. Hence, an agreement percentage of 95.2% is achieved.

3.3.1. General programming

This topic covers queries about general programming, for example, Object-Oriented (OO) principles, using Python with other programming languages, a general queries about implementing logic. Table 9 shows some example questions relating to this topic. One of the major concerns that we noted in this topic was implementing OO principles in Python. For example, question 1 (QId: 65 676 114) in Table 9 shows a query where the developer is asking a question about "importing class from different scripts'. Importing classes or global variables from different files requires an understanding of the OO concepts, an incorrect initialization will lead to an error. The second main query in this was about using Python with other programming languages, for example, javascript, ruby, C++, SQL, etc. question 2 (QId: 26 071 943) in Table 9 shows a query where the developer is asking a question about how to use Ruby and Python together. This reveals that even though many individuals are familiar with Python programming concepts, creating practical applications that integrate with libraries of other languages is still challenging. Developers use Python in various applications and hence several times they need to use Python with other programming languages. Providing appropriate support to the developers when they are using Python with other programming languages can be useful. We also observe several queries related to the logic of code with string, lists dictionaries, time-date operation, etc. Question 3 (QId: 21 162 471) in Table 9 shows an example of a query where a developer seeking help with reversing a string.

Table 9: Example of question belonging to the topic "General Programming"

S. No	QId	Title
1	65 676 114	How to **import class** from different script, where the class uses a global variable at initialization?
2	26 071 943	control stdin and stdout of a **ruby** program in python
3	21 162 471	**Search and replace string** with reverse

In summary, from analysis of this topic, we find that developers face difficulty in applying and using OO principles, using Python with other programming languages, and implementing logic with data structures like strings, lists, dictionaries, etc. To help developers with these more tutorials can be written to use the OO principle in Python. Developers need support in terms of API when they are integrating Python with other programming languages. Hence, the current API can be improved so that it becomes easier for developers to use Python with other programming languages. This is also the **biggest topic** if we see the size of the topic, i.e., the number of questions in each topic, and hence by addressing queries related to this topic needs of a large number of developers can be addressed.

3.3.2. Scientific computing

This topic covers queries about advanced computing concepts that solve complex challenges, such as artificial intelligence, machine learning, neural networks, computer vision, etc. Most of the conversations in this area concerned handling data in the proper format for implementing various models, big data libraries, and general data science concepts. Some examples of questions belonging to this category are shown in Table 10. The majority of the inquiries were on how to efficiently handle and operate data using Panda's library. For example, in example 1 (QId: 41 814 828) in Table 10, the developer is seeking help with reading nested JSON in Pandas data frames. In addition to the pandas' package, there are numerous more queries about data preparation before model implementation. For example, refer to example 3 (QId: 65 255 029) in Table 10 where the developer is asking a query about converting data to glove or word2vec format. A lot of developers asked for help utilizing PySpark to operate on data, such as in example 4 (QId:70 520 386) in Table 10, the developer is asking a question about extracting a specific column in Pyspark. Another prominent topic of discussion in this field is the theories of many machine learning models, algorithms, performance measurements, and scientific computing concepts. For example, consider example 5 (QId: 59 793 818) in Table 10, where the developer is seeking help in understanding the negative cross-validation score function. We also notice some questions related to data crawling (QId: 34 542 381).

Table 10: Example of question belonging to the topic "Scientific Computing"

S. No	QId	Title
1	41 814 828	Trouble reading nested JSON in **pandas** dateframe
2	49 517 830	Incorrect regex identification using **pandas**
3	65 255 029	How can I convert a **dataset** to glove or word2vec format?
4	70 520 386	Extract specified string from a column in **PySpark dataframe**
5	59 793 818	What does negative **cross_val_score()** mean?

In summary, from this topic, we observe that developers face issues in data extraction and data manipulation when using libraries such as Pandas and pyspark. The research community should provide tools to the developer community so that they can efficiently and easily process data. Joy et al. [22] performed a detailed study about questions related to Pandas on Stack Overflow and identified five major categories in which developers face difficulty. Among them, the data frame-based issues are at the top position which matches the finding of our paper. They also need help in understanding the output of machine learning libraries. Hence, these libraries can be improved to help developers in interpreting the results. For example, if a model is outputting negative cross-validation then the developer can be informed that there is some error in the code and the developer needs to correct it.

3.3.3. Web development

This topic primarily discusses various aspects of web frameworks and the challenges that developers encounter in web development. The two primary Python web frameworks, Django and Flask, were the major topics of discussion. Table 11 contains a few examples of questions in this category. The majority of Django discussions focused on user profiles, forms, and models, among other topics. For instance, example 1 (Question Id: 12 526 065) shows a new Django developer who needs assistance in setting up a user profile for a web application. While seeking help on Stack Overflow, the developer also criticizes the lack of documentation for this concept.

Table 11: Example of question belonging to the topic "Web Development"

S. No	QId	Title
1	12 526 065	**Django:** create new user and profile
2	52 264 764	how to access field using **django** signals
2	53 243 629	How to retrieve logged user id in **flask**

Additionally, a developer in example 2 is having trouble using Django signals to access fields. The developer claims that the documentation was insufficient to fix the problem. Also, a lot of developers had queries about the Flask framework. For example, a developer is requesting information about a flask library to retrieve the user id after user login in Example 3 (53 243 629) of Table 11.

In conclusion, we can see from this topic that developers struggle to understand ideas from existing documentation, which needs to be improved. An alternative approach might involve adding more courses that focus on building web applications from start using Django and Flask by providing in-depth examples that can assist developers.

3.3.4. Data structures and formats

This topic covers queries about operations on data structures like lists, dictionaries, arrays, strings, vectors, etc. In this topic, we noted three main developer concerns, 1) efficient use of these data structures, 2) conversion from one data structure type to another type, and 3) using nested data structures. Table 12 shows some examples of questions that belong to this topic. Question 1 (QId: 5 324 217), in Table 12 shows an example where the developer is asking about the processing of nested lists. Question 3 in this table, shows an example of queries where developers are seeking advice about turning the list into a dictionary. In large programs, it often happens that developers need to convert from one data structure to another hence simple functions that can help developers in making such conversions can be useful. In question 4 (QId: 25 982 638), the developer is asking a question about finding, a fast way to convert a list to an array.

Table 12: Example of question belonging to the topic "Data Structure and Formats"

S. No	QId	Title
1	53 242 179	Comparing multiple **lists inside of lists**
2	39 663 866	Manipulating **dictionaries within lists**
3	58 340 665	**Turning list into dictionary**
3	25 982 638	H5py: **fast way** to save list of list of numpy.ndarray?

In summary, from analysis of this topic, we observe that the developer faces difficulty using nested data structures and converting from one data structure to another. The research community and the Python developer community need to design some methods or APIs that make such kinds of operations easier. In addition, we observe that the developers need help in optimizing their code. Code optimization is an important problem and it is noticed that developer face difficulty in it [8]. Permua et al. [8], also reported that developers face difficulty in code optimization. They reported issues related to code optimization and refactoring such as simplifying switch-case statements or loops, duplicate code removal, loops, compacting logic, etc. Our finding provides another dimension to these results by suggesting that developers seek help in code optimization when converting from one data structure to another.

3.3.5. Installation, deployments, and IDE

This topic covers issues related to installation, import errors, deployments, and problems with Integrated Development Environments (IDEs). We notice three main concerns in this topic, 1) issues with the Virtual environment, 2) resource access or import error, and, 3) issues with pip/conda install. Table 13 shows a few examples of questions related to this topic. The first major issue that we identified was related to Python *virtual environment*. A virtual environment is created on top of the "base" Python environment. It is used to isolate from the package environment of the base environment. We noticed two major concerns of the developers first, issues with package installation in the virtual environment and the wrong listing of packages in the virtual environment. Question 1 (QId: 41 801 382) shows a query where the developer is asking questions about installing packages in a specific virtual environment. The second major concern was resource access or import error, i.e., the developer was facing issues related to access to a certain resource especially when they integrated Python with other libraries/technologies such as docker, flask, AWS, google app engine, cython, WIN API, psycopg2, etc. Question 2 (QId: 45 266 200), shows an example question where the developer is asking for help to access a resource in Python when it is integrated with AWS. The third main concern we notice was related to the installation of packages using pip or conda. We notice that developers face lots of difficulty in installing packages because of different versions of Python and pip or different operating systems (e.g., Linux, windows, etc.). Table 13 shows an example (question 3, QId: 30 993 086) where the developer is getting an error about the installation of pip.

Table 13: Example of question belonging to the topic "Installation, deployments, and IDE"

S.No	QId	Title
1.	41 801 382	How to install packages into **specific virtualenv** created by conda
2.	45 266 200	Python in AWS Lambda: "module "requests" has no attribute "get"
3.	30 993 086	pip3: command not found but python3-pip is already installed

In summary, from the analysis of this topic, we notice issues related to the installation of packages and libraries. The research community and the Python development community need to provide some easy methods so that developers can easily install packages without the need to worry about the exact version. Because from the analysis, it is observed that the Python packages are not very stable, and using a different version can give errors. Hence, there is a need to make Python more robust by making it backwards-compatible.

3.3.6. OS, multi-threading, and multiprocessing

This topic covers discussion about OS process execution, subprocess management, job scheduling, client-server issues, database/SQL issues, and multitasking/multithreading.

We observed that the major concerns in this topic are 1) Operation with the "celery" framework, 2) Execution of threads, 3) Running multiple processes/subprocesses, 4) Database/SQL-related issues, and 5) Job scheduling. Table 14 shows some example questions in this topic category. The first major area of discussion in this topic category was on celery framework. Celery is an open-source asynchronous task queue or job queue which is based on distributed message passing[33]. An example is Question Id: 61 221 609 in Table 14 which discusses techniques for running many processes simultaneously. Similarly, Question Id: 53 044 978 is also on the Celery framework which has no answers. These questions do not have a satisfactory response, which could be an indication of a lack of Celery experts in the Stack Overflow platform.

Table 14: Example of question belonging to the topic "OS, multi-threading, and multiprocessing"

S. No	QId	Title
1	61 221 609	How to get multiple **celery** task results at the same time?
2	53 044 978	How to use a **Celery** worker
3	32 565 819	Python socket programming with **threads**
4	10 604 958	Run and get output from a **background process**
5	13 060 427	sorting and selecting **data**
6	41 604 289	urllib2 <urlopen error [Errno 61] **Connection refused**>

Another major area of discussion within this topic is the execution of processes. For example, Question Id: 10 604 958 discusses strategies for getting output from background processes, and the developer also edited the post later to indicate that the responses offered were insufficient to resolve the issue. We also noticed many questions related to thread execution, an example is Question Id: 32 565 819 about socket programming with Python threads. This question also did not receive an accepted answer. Additionally, we observed that there are numerous questions regarding the scheduling of programs and execution of jobs at various times. We notice client server-related issues in this topic where the developer asks for issues related to making connections between client servers or processing/managing data between these two. Question Id: 41 604 289 shows, an example where the developer is getting a connection refused error when trying to connect to the server. We also notice several SQL/database-related issues, for example, Question Id: 13 060 427, shows a query from the developer where she is asking how to use Python and SQL together.

In summary, from analysis of this topic, we notice issues related to distributed systems, threads, processes/subprocesses, Database/SQL-related, and Job scheduling. We also notice a lack of satisfactory response in this topic area which indicates the need for more community experts to answer OS-related queries. The research community needs to work and provide an appropriate tool to the developers that make it easier for the Python developer such advanced operations.

3.3.7. Graphs and plotting

This section includes discussions about plotting and adding colours to graphs and it covers 6.64% of the total number of documents in the analysis. The major areas of discussion we observed in this category were on libraries like "matplotlib" and "plotly". Additionally,

[33]https://docs.celeryq.dev/en/stable/getting-started/introduction.html

there were a few questions about the "seaborn" library. Table 15 gives an overview of a few questions in this topic category.

Queries on the matplotlib library were most commonly seen within this topic. One example is Question Id: 22 923 402 in Table 15 where the developer is struggling with enlarging the axis size. Enlarging the size of the axis, labels and different colour schemes is crucial in graph plotting. Another example is Question Id: 64 568 227 in Table 15 where a developer struggles with adding grids to the background of a plot. This question did not receive an accepted answer.

Another major area of discussion in this category was the plotly library. For instance, consider the question with Id:64 949 228 (Table: 15), where a developer seeks help to draw a horizontal line in the graph in a different coordinate position. The developer also gives a sample output obtained for the same problem with the matplotlib library. This could indicate that many developers struggle with some basic operations in plotly, and improving the documentation with more examples could help to address this issue. Another example is Question Id: 70 889 046 (Table 15) where the developer asks about colouring Sunburst rings based on different conditions. The question did not receive any answers suggesting a lack of experts in the Stack Overflow platform.

Table 15: Example of question belonging to the topic "Graphs and plotting"

S. No	QId	Title
1	22 923 402	**matplotlib:** enlarge axis-scale label
2	64 568 227	How to add a grid graph as a **background** of one graph plot?
3	64 949 228	Plot horizontal lines in **plotly**
4	70 889 046	How do I colour Sunburst rings in **Plotly** based on different conditions

In summary, from our analysis of this topic, we found that developers face difficulties in some of the basic concepts of axis size, labels and colour schemes in both plotly and matplotlib libraries. Also, we noticed a lack of experts in the Stack Overflow platform to provide answers regarding the plotly library. Improving the documentation on these libraries with more examples will make it easier for developers to understand the fundamentals of plotting. Also, we observed a lack of adequate community experts to answer questions related to the plotly library.

3.3.8. File handling and formats

This topic addresses queries pertaining to various file operations and the handling of multiple files. We noticed that some of the common areas of discussion in this category are managing multiple files simultaneously, errors in file handling, file operations like read, write, compress, etc., and altering the content of files. Some examples of questions belonging to this category are given in Table 16.

The majority of the questions were about operating with different file formats. Consider the example Question Id: 45 710 477 in the Table 16 where a developer is trying to use values from one Python file to another. There are many similar questions on Stack Overflow where developers struggle with handling two or more files. Another major area of discussion is regarding errors while handling files. One such example is 2 (Question Id: 62 133 256) in Table 16. Here, the developer gets an Out of Memory error while operating on a large file and seeks help on Stack Overflow. But the question did not receive any response from

Table 16: Example of question belonging to the topic “File Handling and Formats”

S. No	QId	Title
1	45 710 477	Importing variables from **another file** in Python
2	62 133 256	numpy loadtxt for **large text files**
3	19 277 816	python logging: how to get **compressed** .gz log file using TimedRotating-FileHandler
4	55 481 848	How to **modify the contents** of text file?

the platform. Often, the search for and taking of corrective measures to correct errors in software development is extremely time-consuming and has a significant effect on the cost. This is explained in detail in the study [23].

The other major area of discussion was on compressing files, as in Example 3 (Question Id: 19 277 816) in Table 16 where a developer wants to compress the output file to a particular format. The question did not receive any answers from the platform. Another area of discussion was altering the contents of the file. For instance, example 4 (Question Id: 55 481 848) in Table 16 is about replacing commas in a file where a file seek operation can resolve the issue.

In conclusion, from our analysis of this topic, we noticed issues related to multiple file management, errors in file handling, file operations like read, write, and compression, and modification of file content. We could also observe a lack of response from the Stack Overflow platform in many of these questions, which suggests that more experts are needed to answer queries. Also, errors in file operations need to be addressed to reduce the impact on software development costs and time. So, the developer community can work on documenting some of the common errors and mitigation steps, especially in the case of large files.

3.3.9. GUI Support

This was the topic category with the fewest documents (i.e., 3.7%) and includes issues related to the graphical user interface (GUI). Most of the questions were on popular Python GUI libraries such as “pyqt”, “tkinter”, and “wxpython”. Table 17 contains a few examples of questions belonging to this category. Considering the library “tkinter”, an example question is 1 (Question Id: 54 122 578) in Table 17 where a developer struggles with labels in the window displayed. In example 2 (54 122 578) of Table 17 a developer keeps running into errors. The question did not receive any accepted answer. Another major discussion area was regarding the “pyqt” library. For instance, consider example 3 (Question Id: 22 299 612) from Table 17 where a developer encounters trouble displaying colour icons in the menu.

Table 17: Example of question belonging to the topic “GUI Support”

S. No	QId	Title
1	54 122 578	**Tkinter** labels not showing in the pop-up window
2	54 122 578	**Tkinter** variable classes not defined
3	22 299 612	How to display colour icons in the menu, **PyQt**

We observed that many developers struggle with the basics of graphical user interface development. Although these questions received satisfactory responses from the platform, measures can be taken by the educator community to create more courses in Python GUI libraries.

Summary for RQ 3: Our analysis of the Python posts reveals that there are nine major topics: *General programming, Web development, Scientific Computing, Data Structures and Formats, Installation deployments IDE, OS multi-threading and multiprocessing, Graphs and plotting, File Handling and Formats*, and *GUI Support.* Each of the topics has some main issues, however, we notice some major concerns like integrating Python with tools and languages, problems associated with understanding and changing between different data structures, issues related to Django and Flask, issues in installing packages using Pip/conda, issues related to the subprocess, multi-threading, etc.

3.4. RQ 4: What are the most popular and challenging topics discussed by developers in Stack Overflow?

Motivation: In this RQ we aim to identify the type of Python questions on Stack Overflow that developers find interesting and challenging. This RQ is based on the results from RQ 4, and the analysis is carried out in a more fine-grained manner to comprehend the Python topics that are thought to be the most well-liked and difficult by the Python community. This will be useful to obtain more detailed and focused insights on the patterns of Python questions, problems, and demands on supporting activities. Additionally, identifying the popular and difficult topics might enable organizations like Kaggle and Stack Overflow to assign and hire additional experts in those areas.

Approach: As in earlier research, we utilise three complementary popularity measurements to determine a topic's popularity [8, 13, 24–27]. First, the average number of registered and unregistered users who have viewed the post. More views suggest that more developers have encountered similar difficulties, hence the topic is more popular. Overall, this metric measures the Python community's interest based on how frequently the post is viewed.

Second, the average number of posts identified as favourites by the community. The favourite count measure illustrates how useful the question was. Additionally, the higher favourite count suggests that more individuals have encountered the same problems and found the solution useful. Overall, this metric assesses the popularity of questions depending on how useful they are.

Third is the average score for each of the posts. On Stack Overflow, users can give a question an upvote if they think it is useful. The score is then calculated using these upvotes. A higher score indicates that more developers find the subject intriguing and have faced similar problems. Therefore, we utilise this as a measure of each question's perceived community value.

As in previous research, two metrics are employed to determine the difficulty of each topic [8, 28]. First, the number or percentage of posts without an accepted answer is evaluated. Only the person who asked the question has the capability to mark an answer as an accepted answer when they are satisfied with the solution. Therefore, a question with an accepted answer is an indication of user satisfaction. Also, the fact that so many questions have been made without any accepted answers shows how difficult the question in the topic is to answer [13, 29]. Second, we look at the question in a topic for which there are no answers. Sometimes, even when they obtain a suitable response, users who ask questions neglect to designate the response as accepted. Therefore, considering the queries for which there is no answer, it may be inferred that neither experts nor developers were able to provide an answer, making the question difficult [8, 28].

Results: The results are subdivided into two sections topic popularity and topic difficulty.

Topic Popularity: Table 18 shows that the most popular topics are Data structures and Formats and Installation, Deployments, and IDEs, while File Handling and Formats are the least popular. Even though there are fewer questions related to Installation, deployments, and IDE than General programming, the average view count for the former is larger than the latter. This implies that many developers are constantly seeking and viewing the installation and IDE-related questions. The most popular topic is Data structures and Formats, according to the other two metrics average score and average favourite count. This demonstrates that the solution to issues based on data structures was more helpful and satisfactory to the developers.

Table 18: Popularity of Python topics

Topic	Count	Average View Count	Average Score	Average Favorite Count
General programming	3826	2082.792	2.048	0.605
Scientific computing	2602	1883.971	1.640	0.500
Web development	2478	2057.166	1.982	0.499
Data Structures and Formats	2421	6082.245	4.495	0.980
Installation, deployments, and IDE	1396	5707.434	4.590	1.113
OS, multithreading, and multiprocessing	1172	1964.17	1.732	0.479
Graphs and plotting	1099	2253.111	1.626	0.501
File Handling and formats	922	1785.682	1.085	0.284
GUI Support	613	1908.889	1.334	0.368

The question with the highest score and favourite count in the entire dataset is "What does the 'yield' keyword do?" with a score of 10 150 and 6431 users selecting it as their favourite post, evidences the lack of documentation. It is interesting to note that the findings of this study suggest that questions about Installation, deployments, IDE, and Data structures and formats provide the most challenges to Python programmers, as illustrated by the higher view and favourite counts of these topics. However, questions about Data Structures and Formats receive more satisfactory answers from the experts on the Stack Overflow platform. The rising popularity of these issues can be addressed by creating documentation on every functionality and library that are more comprehensive and accessible.

Topic Difficulty: Table 19 shows the difficulty level of various topics. We categorize the topics into four levels of difficulty using the metric percentage of questions without answers (PQ-WAnA) and percentage of questions without an accepted answer (PQ-WAcA). Following are the levels we created (level 1 is the most difficult and level 4 is the least difficult):

level 1 = [PQ-WAnA $\geq$ 0.229 and PQ-WAcA $\geq$ 0.621],
level 2 = [0.180 $\geq$ PQ-WAnA $\leq$ 0.228 and 0.529 $\geq$ PQ-WAcA $\leq$ 0.620],
level 3 = [0.162 $\geq$ PQ-WAnA $\leq$ 0.179 and 0.482 $\geq$ PQ-WAcA $\leq$ 0.528],
level 4 = [PQ-WAnA $\leq$ 0.162 and PQ-WAcA $\leq$ 0.481].

Table 19: Difficulty of Python topics

Level	Topic	Count	Without accepted answer %	Without answer %
3	General programming	3826	0.481%	0.162%
3	Scientific computing	2602	0.461%	0.1575%
3	Web development	2478	0.515%	0.186%
4	Data Structures and Formats	2421	0.422%	0.111%
1	Installation, deployments, and IDE	1396	0.621%	0.229%
2	OS, multithreading, and multiprocessing	1172	0.534%	0.189%
2	Graphs and plotting	1099	0.529%	0.224%
3	File Handling and formats	922	0.517%	0.162%
3	GUI Support	613	0.513%	0.197%

Table 19 reveals that the Installation, Deployment, and IDE issue has the most questions without any accepted answers or answers. We observed that questions related to Installation, IDE and deployment were also in the popular topic category. This implies that a large number of individuals are looking for these questions and not getting satisfying answers. The lack of answers may be caused by poorly constructed or ambiguous questions that lack specific details about the issue at hand. This, however, emphasizes the significance of the need for efficient resources and support on package installation and IDE-related issues in Python.

To avoid the difficulties faced by developers in the early stages of development, it is essential to address these issues. Another noteworthy finding is that the issue of data structures and formats is both popular and less complex. This topic has the lowest percentage of questions without accepted answers and answers. This can suggest that, even when developers encounter data structure and format-related concerns, the community can supply adequate and acceptable answers.

Correlation between topic popularity and difficulty: Similar to the previous studies [8], we performed a correlation analysis between the popularity and difficulty metrics. We compute the Pearson correlation between six pairs of metrics, 1) average view count and the percentage of questions without accepted answers, 2) average view count and the percentage of questions without answers, 3) average Score and the percentage of questions without accepted answers, 4) average score and the percentage of questions without answer, 5) average favorite count and the percentage of questions without accepted answers, and 6) average favorite count and the percentage of questions without answer. Figure 10, shows the scatter plots for all six pairs. This figure shows a weak correlation between popularity and the "percentage of the question without any answer" difficulty metrics. We do not observe much correlation between popularity metrics and the other difficulty metrics (i.e., percentage of questions without accepted answers). This figure shows two outlier topics, i.e., "installation deployment and IDE" topics and "Data structures and formats topics. The installation deployment and IDE topic has high popularity and difficulty whereas the "Data structures and formats" has high popularity but low difficulty, i.e., this topic is popular but receiving satisfactory answers from the community.

To obtain a deeper analysis of the correlation, we analyze Pearson and Spearman correlation coefficient between popularity and difficulty metrics. Table 20 shows the Pearson correlation value obtained for each of the pairs. We obtained a positive correlation between % of questions without accepted answers and all the three popularity metrics whereas

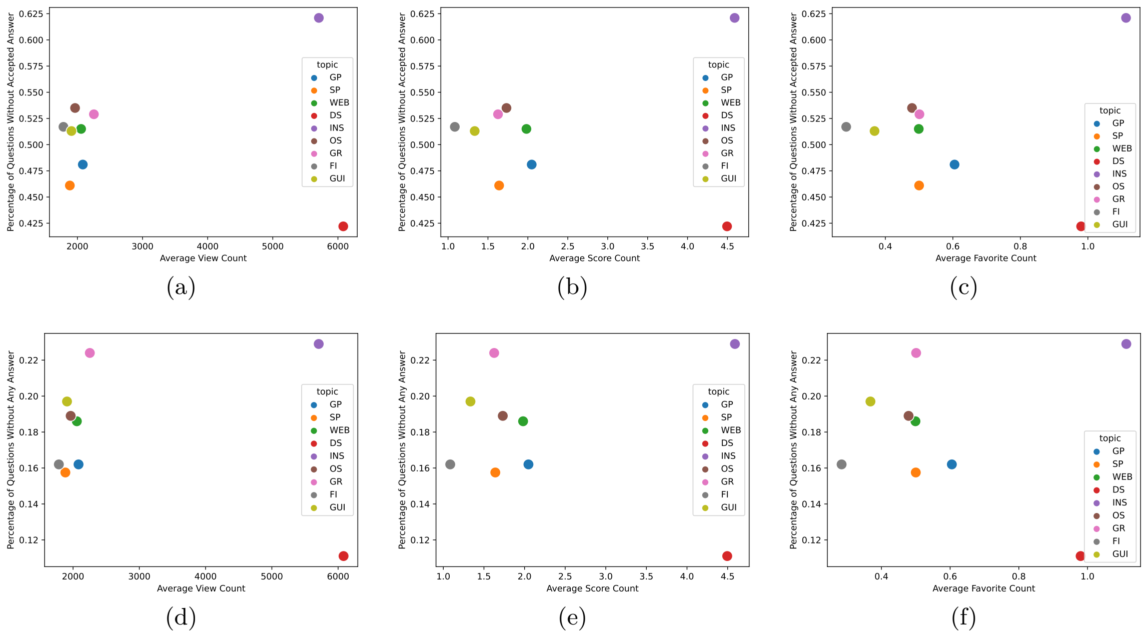


Figure 10: Scatter plots showing the correlation between popularity and difficulty matrix for all the nine topics: GP: General Programming, SP: Scientific Computing, WEB: Web Development, DS: Data Structure, INS: Installation Deployment and IDE, OS: OS, Multi-threading and multiprocessing, GR: Graphs and Plotting, FI: File handling and Formats, GUI: GUI support

a negative correlation between % questions without answers and all the three popularity metrics. However, none of this correlation is statistically significant as indicated by the *p*-values. Next, we perform Spearman correlation analysis. Table 21 shows that there is a low positive correlation between the average view count and percentage of questions without any answer using Spearman correlation. However, this is not statistically significant as indicated by the *p*-values. To further explore the reason *why some questions are not getting any answers*, we manually analyzed a sample of unanswered questions.

Table 20: Pearson Correlation between Popularity and Difficulty Metrics, Cor: correlation

	Average view count (cor/*p*-value)	Average score (cor/*p*-value)	Average favorite count (cor/*p*-value)
% w/o any answer	−0.164/0.671	−0.133/0.732	−0.04/0.91
% w/o accepted answer	0.071/0.855	0.104/0.788	0.164/0.67

Analysis of questions without any answer: In the next step, we analyzed questions without any answer for the topic "*installation, deployment, and IDE*". We extracted a statistically significant sample of 172 questions. This sample represents a confidence level of 95% and an interval of 5% from a total of 308 unanswered questions from this topic. Two of the authors manually analyzed these questions. From our analysis, we observed that there were very few questions that did not get any comments from the Stack Overflow community. Those consist of low-quality questions or questions which needed more details.

Table 21: Spearman Coefficient between Popularity and Difficulty Metrics, Cor: correlation

	Average siew sount (cor/p-value)	Average score (cor/p-value)	Average favorite count (cor/p-value)
% w/o any answer	0.175/0.651	−0.033/0.93	0.033/0.93
% w/o accepted answer	0.033/0.932	−0.033/0.93	−0.083/0.83

For most of the questions, the Stack Overflow community provided help in the form of comments where they discussed potential solutions, providing links to other related Stack Overflow questions, or other useful links (e.g., bug issues reports on GitHub, etc.). We notice in that there were many questions where the users were trying to integrate Python with other technologies or IDEs like C/C++, Cython, Pycharm, Spyder, AWS, and Docker, and hence were facing installation issues. We also notice several questions where users were facing issues in installing Python packages on the Linux environment.

Summary for RQ 4: Our analysis reveals that *Data structures and formats* and *Installation, Deployments, and IDEs* are among the most popular topics. *Installation, Deployment, and IDE* was found to be the most difficult topic. Our analysis of unanswered questions reveals that many unanswered questions are not difficult but they did have useful comments and some of them were not clear.

4. Discussions and takeaways

According to the findings of the study, Stack Overflow is an extremely popular forum for developers to ask questions and engage in discussions about the difficulties they face when programming in Python. In most circumstances, developers can ask for help in a variety of Python libraries and applications and receive a suitable response. To further our understanding of the difficulties faced by Python developers, this quantitative research is supplemented with a manual analysis of a statistically significant sample of posts. The results also assist educators in updating their course content to accommodate real-world situations and practical language applications. In this section, we provide details of findings that can help the community.

4.1. Research community takeaways

The findings of this study demonstrate the difficulties that Python developers face in real-world projects. These findings highlight the gap between the academic elements of Python and its practical real-world application. This provides an opportunity for the research community to better investigate and contribute to this field.
Major topics: According to the findings of this study, the major topics discussed by the Python community are general programming, scientific computing, web development, data structures and formats, installation, deployment, IDE, operating system, multi-threading and multiprocessing, file handling and formats, graph and plotting and GUI support. We notice that the general programming topic has the biggest size, i.e., a large number of questions are asked under this topic. This topic consists of questions like OO principal,

integrating Python with other languages, and implementing simple data structures like lists, dictionaries, etc. The research community should work on developing these concepts further so that it's easier for the software developers to understand. For example, some kinds of transformations or functions can be developed that make it easier to change from one data structure to another.

Universal platform for installation: We notice several questions related to installation issues. Hence, some more research is needed that can make it easier for Python developers to install the packages in an easier manner. From the questions, we notice that there are several ways in which software developers can install packages, for example, using pip, and conda. Also, there are differences in different operating systems like Linux and Windows. Hence, some unified tools are needed that can make package installation easier. Hence, some research is needed to develop a universal package installation system.

OS, multi-threading, and Multiprocessing issues: One of the most problematic subjects for the Python community were OS, multi-threading, and Multiprocessing. A manual examination of the sample questions indicated that several of them had no accepted answer. Although this issue has received considerable attention from the research community [30, 31], it remains a challenging problem in the Python community.

Python multipurpose language: We notice that Python is a multi-purpose language because it is being used in many applications like web development, graph plotting, scientific application, database connections, and as an introductory programming language. Because the software developers are working on so many applications, all these areas have their individual issues as well. Hence, more detailed research is needed that can address the list of issues faced by developers in each of these domains. The are some introductory studies in these areas, for example, analysis of popular topics in Pandas library [22]. More such detailed studies are needed that can help in uncovering specific issues faced in each domain.

4.2. Educator community takeaway

The educational community can leverage the topics obtained from the topic modelling results as a foundation for designing and developing Python courses. The analysis of the topic of general programming revealed that developers find it difficult to integrate various libraries and packages into real-world applications while using general Python concepts. Educators can take this into account and develop courses that focus on **practical applications and demonstrations of appthelying various concepts at an advanced level**. Furthermore, many programmers encounter various errors while coding, therefore **tutorials can be created** with more examples and separate sections for illustrating potential errors, corrective measures, and frequently asked questions in community-based platforms like Stack Overflow. Additionally, more courses on challenging topics like installation, deployments, IDE and OS, multi-threading, and multiprocessing can be developed. More tutorial needs to be developed to understand the different kinds of **data structures** like lists, strings, dictionaries, etc. The **OO principles** also need to discuss in detail.

4.3. Tool/IDE or package development community takeaway

The tool/IDE of the Python package development community plays a vital role in Python software development. Hence there are specific findings that the vendor community can utilize to improve the tools/IDEs/Packages.

Outputting more meaningful or clear error messages: It is noticed from some of the questions that when there is some error in the code it takes lots of time for software developers to figure out the reason for the error because the error message was very generic. Hence, it becomes difficult to debug the code. Providing more clear or more meaningful error messages can help the software development community.
Fast and efficient implementation: We notice several questions where developers were searching for fast or more efficient implementations of data structures or loops. Providing developers with more efficient implementations can be beneficial.
Easier integration with other programming languages: We notice several questions where software developers reported difficulties in the integration of Python with other programming languages or tools like C, C++, SQL, AWS, etc. Hence, the vendor community should work on making these interactions easier and developing APIs that are easier and more robust to use.
Installation of libraries using pip/conda: There are several queries from the software developers that about the installation of libraries using pip or conda. Software developers were facing difficulties in installing packages on different versions of pip. They were also facing difficulty when they had to install multiple packages and they had a dependency on some specific version or system. Hence, the Python development community should focus on making these installation commands more robust or easier so that software developers can easily install the required packages. These installation needs to be tested on various platforms like Windows, Linux, and MacOS Several developers reported issues with package installation in virtual environments or accessing the already installed packages in virtual environments. The tool development community should also work on making the virtual environment creation more robust.
Backward compatibility of the software packages: Software developers report several issues arising because of non-backwards compatibility of functions. They report errors where a previously working code broke because of an update in the packages. Hence, the Python package development community should focus on making the packages backwards compatible.

4.4. Developer community

The findings of this study can be used by Python developers to take a much more disciplined approach to designing diverse Python applications. Organizations can make use of the findings to ensure that the development team receives the proper training on the required Python tools and resources. These findings should be used by the development community to create better tutorials and documentation to reduce any early barriers for new developers. Our investigation revealed that installation, deployment, and IDE were the most difficult topics, and the development community can create manuals and guides to support Python users.

Senior programmers can assist junior programmers more with initial environment setup and package installations. OS, multithreading, and multiprocessing are among the topics with a large number of posts without accepted answers. Software managers can take this into account and allocate more time and resources to finish these tasks.

Additionally, the current documentation for the libraries and modules in these subjects can be improved. On the other side, growing the amount of material can make it more difficult to locate relevant references. Therefore, many of the challenges mentioned in those questions can be addressed by an automated system that recommends appropriate documentation.

5. Threats to validity

In this section, we discuss various threats that can potentially impact the results of our study. We divide this section into three parts: internal validity, external validity and construct validity [32].

5.1. Internal validity

These are validity concerns that may have an impact on our results. The experimental dataset for this study was generated by extracting and analyzing questions from the Stack Overflow website that consists of at least one of the 174 tags that were identified in our "Python-related" tag list (refer to Section 2.2). These tags were determined by manual analysis of questions and after online research into how the tags relate to Python. To further remove the possibility of bias or false positive/false negative in the selection of tags, two of the authors verified the list of tags. Both of these authors had a detailed discussion in case of disagreement. A tag is included in the list only if both of the authors agree to include that tag otherwise the tag is removed from the list. We extracted all the question that consists of at least one of these tags. Using this approach, we extracted 2 449 567 questions, which is comparable to previous approaches that worked on Python questions posts analysis on Stack Overflow [3]. We decided not to include questions that consist of the "Python" in either title or body but do not consist of any of the identified tags from our "Python-related" tag list. Even though this approach will reduce the total number of questions in our experimental dataset, it avoids the possibility of having a large number of false positives as mentioned in the previous approaches [8]. Our approach ensures that we analyze posts that discuss the issues related to Python developers. Additionally, just like in prior studies [8, 20, 33], our analysis is restricted to only looking at the most recent version of the post.

The inclusion of duplicate questions is another validity issue. There are questions on Stack Overflow that seek answers on similar topics or areas, and we haven't used any strategies to separate such questions. The outcomes of this study could be impacted by this. In the future, we intend to broaden our investigation by comparing the outcomes after eliminating duplicates. We also do not analyze the comments associated with Python posts. Comments on questions consist of temporary post-it notes and only privileged users can post comments on a question [8].

In RQ 1.3, we notice that all the questions got answered within 24 hours. We consider a sample of all Pythom questions in this paper. All of those questions in this sample got the answer within 24 hours. However, if we consider the real world there can be a question that took much more time to get the answer. Our dataset included 2,400,000 questions and 26,000,000 answers. To answer the research questions (RQ 1.3, RQ 1.4) we had to run two queries each on all of these questions and one single query was taking around 50 seconds. So, processing the entire set of questions for all the answers was not feasible and would have taken years with our processing capability. So, we had to take a sample set of questions (16 529) to explore these questions with a 99% confidence level that the results given by the query on these 16 529 questions are a representation of the entire population. We took this approach to determine the sample as it was used in many previous papers [12, 13]. We did all the cautions while selecting the sample but still, there can be some bias in the results because of the specific sample that we used. We plan to do more detailed research on a bigger dataset to mitigate this issue.

In addition, in our analysis of popular and difficult questions, we considered questions that are not answered as difficult. In Stack Overflow, questions might be too long or too short, a duplicate of other, ambiguous, and with insufficient details. On Stack Overflow, we frequently see comments asking for more information on questions. These could be some of the reasons it went unanswered or did not have an accepted answer. This might have an impact on the results of our analysis of difficult Python Stack Overflow questions.

In this study, we chose a k value of 20 as the ideal number of topics for topic modelling. The LDA model's ability to provide high-quality topics is directly impacted by this value. It is known that determining an ideal number of "k" is tough. To mitigate this risk, experimentation is carried out with a range of k values ranging from 2 to 100 in increments of two, and topic coherence is calculated for each k value, as in many previous studies [8, 34]. Further, using a regular expression, any code snippets are eliminated from the text before being fed into the LDA algorithm to get better results from the LDA model. This methodology has also been used in numerous earlier research using the Stack Overflow dataset [20].

To reduce any further threats to internal validity, built-in Python modules such as "nltk" and "sklearn" are employed for data processing. Another risk to validity is the manual labelling of the topics provided by the LDA model. The group names produced as part of prior tag analysis RQ 3 (3.2) were used to categorise the topics, and they were not changed after the topic model results are obtained, to eliminate any unconscious biases that might have affected the approach. In addition, a manual examination of the generated topics was performed, and similar topics were merged, when necessary, with relevant labels assigned [13].

While analyzing the results of the LDA algorithm, we notice that some of the posts were incorrectly assigned to a different topic. In our analysis of 376 posts, we found 51 (%13.56) false positive posts, i.e., questions that were suited better for other categories. In these false positives, we notice questions from the topics general programming, web, and GUI which were appearing in other topics.

5.2. External validity

This section includes the threats to the generalisation of the findings obtained from this study. This study gathered data for Python question analysis from Stack Overflow, one of the largest sites on the Stack Exchange network featuring conversations on a wide range of programming topics. However, there are numerous other communities where developers can share their problems and ask questions. Stack Overflow is a popular site featuring a variety of domain expertise. But the study can be further improved by including discussions from other platforms or surveying actual Python developers about the difficulties they face. The study's findings also provide snapshots of data that future research can use to assess how the field has changed and how well Stack Overflow is doing. Finally, the study's findings highlight the issues and difficulties that Python developers faced at the time the study was conducted.

5.3. Construct validity

Construct validity examines how theory and observation relate, or whether the measured values correspond to the actual values. The topics that the LDA algorithm produces may not accurately reflect the posts related to those topics. To mitigate this risk, all keywords linked with each subject were analysed, and a random collection of documents or questions associated with each topic were verified on the Stack Overflow website before giving a label to each topic to appropriately depict the topic. Additionally, several metrics are employed as part of RQ 4 to assess the popularity and difficulties of key Python topics, and they could pose a risk to the construct validity. However, these measures are employed since they have been widely utilised in many previous studies [8, 13].

6. Related work

In this section, we present an overview of several studies that are closely related to this research. There is a lot of existing work that evaluates various aspects of different programming languages using the Stack Overflow dataset. This section is organised into three main research lines. The studies that examine programming languages using the Stack Overflow dataset are covered in the first category. The second category focuses on studies that analyse the Stack Overflow dataset to identify various software development challenges, and the third category focuses on identifying issues faced by the Python software development community.

6.1. Analysis of Stack Overflow for programming-related questions

In several studies, the Stack Overflow dataset is investigated to examine various aspects of programming languages. The sociological change that the Java developer community has seen is examined in one study which looks at social dynamics with a focus on user altruism and popularity, both internal and external material cross-referencing, and the topics that have generated significant discussions within the Java community over time [11]. The study uses topic discovery to uncover the main discussion topics, and graph mining to depict social interaction within the community.

Another study uses the Stack Overflow dataset and relevant GitHub activity to investigate well-known programming languages like Go, Swift, and Rust [35]. Several research questions are framed to determine the main topics covered, their evolution over time, difficulty level, resource availability, and the relationship between language growth and developer activities.

Other main work uses Python and R to do survival analysis on a Stack Overflow dataset [36]. The emphasis is on predicting the response time for the first answers and accepted answers, where Python had the highest answer rate and R had the highest acceptance answer rate.

Most of the studies mentioned above-analyzed data on Stack Overflow of various programming languages but Python. To the best of our knowledge, only Tahmooresi [3] worked on analyzing Python questions on Stack Overflow. Our study extends and complements their work in several ways. They analyzed topics and tags on Stack Overflow and proposed a tool to detect similar libraries for Python. However, our study examines a variety of factors, including language evolution, themes, user engagement and satisfaction, tags, topics, and topic popularity and difficulty.

6.2. Analysis of Stack Overflow for software development, maintenance, and other aspects

One main study investigates the Stack Overflow dataset to understand the evolution, practices, and challenges faced by programmers in refactoring [8]. The study examines how refactoring discussions have progressed over time, what are some of the refactoring areas that developers discuss, and which queries are the most popular and challenging. The study looks at the most used tags and terminology in these questions, along with refactor-specific terminologies, as well as the most popular, problematic, and unsolved questions. The goal of the study is to identify areas where refactoring research is lacking, as well as to comprehend why developers' perceptions and actual use of these concepts differ.

Another study used the Stack Overflow dataset to better understand the issues that developers encounter in quantum software engineering (QSE) [12]. They also use GitHub issue reports to learn more about the problems that arise in real-world quantum computing projects. The research focuses on the types of questions about QSE that are posted on the website, QSE topics that are discussed in technical forums, and QSE topics that are highlighted in issue reports of quantum computing projects. The research was able to give insight into future quantum computing potential, uncover several QSE-specific issues experienced by developers, and highlight some of the most popular and difficult quantum computing topics.

An in-depth study uses data from six Q&A Stack Exchange websites to conduct an empirical analysis on logging-related questions [20]. The study looks at the trend of logging questions with accepted answers, the number of answers posted for each question, and the time it takes to receive accepted responses to these questions using statistical analysis. For each of the six websites, the logging questions are also examined in terms of programming languages, with an emphasis on the most common logging questions across programming languages and the distribution of logging questions among programming languages. Content analysis of logging questions is also conducted, with an emphasis on identifying significant topics in the title and description of these questions, as well as analysing the tags associated with these questions. The study shed light on developers' various logging requirements and will aid in the development of better logging tools.

The Stack Overflow dataset is also used in a study on chatbot development issues [13]. This study examines the Stack Overflow posts to learn more about the topics that chatbot developers are interested in and the challenges they face. Topic modelling is used to determine the most popular chatbot-related subjects addressed in the forum and the popularity and difficulty of these topics are also evaluated by the researchers. The report also outlined some conclusions that can be made from it to assist the chatbot community in concentrating on the most pressing issues.

The research discussed above looked into different aspects of software development in various languages. However, none of them concentrated on a comprehensive study of Python language and the various software development issues in the language. Unlike the previous research, this work focuses on analysing the several aspects and challenges of employing Python in software development.

6.3. Analysis of issues faced by Python software development community

The study by Peng [37], performs an empirical investigation on thirty-five prominent Python projects to analyse the use of language features in the projects and develops PYSCAN, an automatic language feature recognizer. The study identifies the top five language characteristics and finds that different domains focus on different language features, with exception handling and nested functions being used most differently. An in-depth examination of these features is performed to summarise their application areas and benefits.

Another study related to Python performs an in-depth analysis of the addition of mutation to Python source code [38]. The mutation method is challenging in dynamically typed languages since the mutant cannot be checked before run time and the applicability of many mutation operators used in other languages is unknown in these dynamically typed languages. The study offers a collection of mutation operators suitable for Python, along with guidelines for choosing operators and a thorough explanation of each. The Python tool for testing mutations, MutPy, is used to implement and analyse each of the discussed operators.

In another major study, a quantitative analysis of Python's performance overheads is carried out with an emphasis on hardware features and language runtime [39]. Additionally, a thorough analysis of how the runtime interacts with the processor's underlying microarchitecture reveal that CPython and PyPy exhibit limited instruction-level parallelism. The paper also discusses several key observations regarding Python's performance optimization possibilities.

All of the above studies concentrate on analysing a single aspect of Python, such as the addition of mutation operations, performance overheads, and language features. The main distinction between our study and others is that our investigation is not constrained to a single language characteristic or paradigm. We attempt to understand the difficulties encountered in the real world by developers while using the language in their applications, as well as some of the areas of Python that are popular and challenging for them.

7. Conclusion

In this empirical study, we conducted a quantitative and qualitative analysis of Python questions posted in Stack Overflow. The quantitative approach involved applying various statistical measures to the extracted data while qualitative analysis involved content analysis of a statistically significant sample of Python questions. Seven research questions were identified, aiming to gain a deeper understanding of the novel challenges encountered by the developers while using Python language in various applications.

The result of this study shows that Stack Overflow is an immensely popular platform where developers seek help and there has been a growing increase in the number of Python questions on the platform from 2008 with a slight drop in 2021. Secondly, analysis of Python tags revealed that the number of questions posted against pandas and TensorFlow is increasing, indicating that scientific computing is becoming a more popular area among Python developers. Thirdly, topic modelling revealed that most of the questions are posted against general programming on concepts like object-oriented principles, algorithm implementation, error management, functional programming, time and date operations, and general Python concepts. Scientific computing, Web development, and Data structures and formats were other topics that developers need assistance with. Finally, the topic of Installation, deployments, and IDE had more view count and was more challenging for developers. Data structures and Formats were identified as the least challenging and most popular topics, with a higher average score and more satisfactory response from the community. This study opens doors for Python researchers and practitioners to further comprehend Python development challenges. From the results obtained from this study, a series of actionable takeaways are highlighted for relevant stakeholders that can improve the field of Python programming and enhance developer productivity.

In the future, we plan to extend topic modelling on the entire dataset to perform a comparison with the results of this study and to analyze how Python topics evolve. At present, we analyzed popular topics based on three metrics, i.e., average view count, average score, and average favourite count. In the future, we plan to consider the size of a topic (#number of questions) as one of the popularity metrics. Also, a structured survey with junior and senior Python developers can be conducted, exploring their general and specific challenges encountered while using Python. This survey can complement and evaluate this Stack Overflow study to provide the software engineering community with a more comprehensive view of Python development practices and difficulties. Finally, the study can be expanded to investigate developers' discussions on other forums and repositories to evaluate the commits and defect reports to obtain further insights regarding numerous problems faced by Python developers.

Funding

This work is not supported by any external funding.

Acknowledgement

We thank Harshit Gujral for helping us with the LDA code.

References

[1] M. Lutz, *Programming Python*. Beijing, Boston, Farnham, Sebastopol, Tokyo: O'Reilly Media, Inc., 2001.
[2] K. Chowdhary, "On the evolution of programming languages," *arXiv preprint arXiv:2007.02699*, 2020.
[3] H. Tahmooresi, A. Heydarnoori, and A. Aghamohammadi, "An analysis of Python's topics, trends, and technologies through mining Stack Overflow discussions," *arXiv preprint arXiv:2004.06280*, 2020.
[4] B.A. Malloy and J.F. Power, "An empirical analysis of the transition from Python 2 to Python 3," *Empirical Software Engineering*, Vol. 24, No. 2, 2019, pp. 751–778.
[5] Z. Zhang, H. Zhu, M. Wen, Y. Tao, Y. Liu et al., "How do Python framework APIs evolve? An exploratory study," in *27th international conference on software analysis, evolution and reengineering (saner)*. IEEE, 2020, pp. 81–92.
[6] R. Widyasari, S.Q. Sim, C. Lok, H. Qi, J. Phan et al., "BugsInPy: A database of existing bugs in Python programs to enable controlled testing and debugging studies," in *Proceedings of the 28th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, 2020, pp. 1556–1560.
[7] A. Tashakkori, C. Teddlie, and C.B. Teddlie, *Mixed methodology: Combining qualitative and quantitative approaches*, Vol. 46. Thousand Oaks, California: Sage, 1998.
[8] A. Peruma, S. Simmons, E.A. AlOmar, C.D. Newman, M.W. Mkaouer et al., "How do I refactor this? An empirical study on refactoring trends and topics in Stack Overflow," *Empirical Software Engineering*, Vol. 27, No. 1, 2022, pp. 1–43.
[9] K. Georgiou, N. Mittas, A. Chatzigeorgiou, and L. Angelis, "An empirical study of COVID-19 related posts on Stack Overflow: Topics and technologies," *Journal of Systems and Software*, Vol. 182, 2021, p. 111089.
[10] G. Pinto, F. Castor, and Y.D. Liu, "Mining questions about software energy consumption," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 22–31.

[11] G. Blanco, R. Pérez-López, F. Fdez-Riverola, and A.M.G. Lourenço, "Understanding the social evolution of the Java community in Stack Overflow: A 10-year study of developer interactions," *Future Generation Computer Systems*, Vol. 105, 2020, pp. 446–454.

[12] H. Li, F. Khomh, M. Openja et al., "Understanding quantum software engineering challenges an empirical study on Stack Exchange forums and GitHub issues," in *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2021, pp. 343–354.

[13] A. Abdellatif, D. Costa, K. Badran, R. Abdalkareem, and E. Shihab, "Challenges in chatbot development: A study of Stack Overflow posts," in *Proceedings of the 17th international conference on mining software repositories*, 2020, pp. 174–185.

[14] W. McKinney et al., "pandas: A foundational Python library for data analysis and statistics," *Python for High Performance and Scientific Computing*, Vol. 14, No. 9, 2011, pp. 1–9.

[15] C. Jacobi, W. Van Atteveldt, and K. Welbers, "Quantitative analysis of large amounts of journalistic texts using topic modelling," *Digital Journalism*, Vol. 4, No. 1, 2016, pp. 89–106.

[16] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," *To Appear*, Vol. 7, No. 1, 2017, pp. 411–420.

[17] Y. Zhang, R. Jin, and Z.H. Zhou, "Understanding bag-of-words model: A statistical framework," *International Journal of Machine Learning and Cybernetics*, Vol. 1, No. 1, 2010, pp. 43–52.

[18] D.M. Blei, A.Y. Ng, and M.I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, Vol. 3, No. 4/5, 2003, pp. 993–1022.

[19] R.H. Ali and E. Linstead, "Modeling topic exhaustion for programming languages on Stack Overflow," in *SEKE*, 2020, pp. 400–405.

[20] H. Gujral, A. Sharma, S. Lal, and L. Kumar, "A three dimensional empirical study of logging questions from six popular Q & A websites," *e-Informatica Software Engineering Journal*, Vol. 13, No. 1, 2019.

[21] M. Röder, A. Both, and A. Hinneburg, "Exploring the space of topic coherence measures," in *Proceedings of the eighth ACM international conference on Web search and data mining*, 2015, pp. 399–408.

[22] S.K.S. Joy, F. Ahmed, A.H. Mahamud, and N.C. Mandal, "An empirical studies on how the developers discussed about pandas topics," *arXiv preprint arXiv:2210.03519*, 2022.

[23] J.C. Westland, "The cost of errors in software development: Evidence from industry," *Journal of Systems and Software*, Vol. 62, No. 1, 2002, pp. 1–9.

[24] S. Ahmed and M. Bagherzadeh, "What do concurrency developers ask about? A large-scale study using Stack Overflow," in *Proceedings of the 12th ACM/IEEE international symposium on empirical software engineering and measurement*, 2018, pp. 1–10.

[25] K. Bajaj, K. Pattabiraman, and A. Mesbah, "Mining questions asked by web developers," in *Proceedings of the 11th Working conference on mining software repositories*, 2014, pp. 112–121.

[26] M. Bagherzadeh and R. Khatchadourian, "Going big: A large-scale study on what big data developers ask," in *Proceedings of the 2019 27th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, 2019, pp. 432–442.

[27] S. Nadi, S. Krüger, M. Mezini, and E. Bodden, "Jumping through hoops: Why do Java developers struggle with cryptography APIs?" in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 935–946.

[28] X.L. Yang, D. Lo, X. Xia, Z.Y. Wan, and J.L. Sun, "What security questions do developers ask? A large-scale study of Stack Overflow posts," *Journal of Computer Science and Technology*, Vol. 31, No. 5, 2016, pp. 910–924.

[29] C. Rosen and E. Shihab, "What are mobile developers asking about? A large scale study using Stack Overflow," *Empirical Software Engineering*, Vol. 21, No. 3, 2016, pp. 1192–1223.

[30] A. Malakhov, D. Liu, A. Gorshkov, and T. Wilmarth, "Composable multi-threading and multi-processing for numeric libraries," in *Proceedings of the 17th Python in Science Conference, Austin, TX, USA*, 2018, pp. 9–15.

[31] Q. Nguyen, *Mastering Concurrency in Python: Create faster programs using concurrency, asynchronous, multithreading, and parallel programming.* Birningham, UK: Packt Publishing Ltd, 2018.

[32] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell et al., *Experimentation in software engineering.* Springer Science and Business Media, 2012.

[33] H. Gujral, S. Lal, and H. Li, "An exploratory semantic analysis of logging questions," *Journal of Software: Evolution and Process*, Vol. 33, No. 7, 2021, p. e2361.

[34] M. Alshangiti, H. Sapkota, P.K. Murukannaiah, X. Liu, and Q. Yu, "Why is developing machine learning applications challenging? A study on Stack Overflow posts," in *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM).* IEEE, 2019, pp. 1–11.

[35] P. Chakraborty, R. Shahriyar, A. Iqbal, and G. Uddin, "How do developers discuss and support new programming languages in technical Q&A site? An empirical study of Go, Swift, and Rust in Stack Overflow," *Information and Software Technology*, Vol. 137, 2021, p. 106603.

[36] L. Lord, J. Sell, F. Bagirov, and M. Newman, "Survival analysis within Stack Overflow: Python and R," in *4th International Conference on Big Data Innovations and Applications (Innovate-Data).* IEEE Computer Society, 2018, pp. 51–59.

[37] Y. Peng, Y. Zhang, and M. Hu, "An empirical study for common language features used in Python projects," in *International Conference on Software Analysis, Evolution and Reengineering (SANER).* IEEE, 2021, pp. 24–35.

[38] A. Derezińska and K. Hałas, "Analysis of mutation operators for the Python language," in *Proceedings of the Ninth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX. June 30–July 4, 2014, Brunów, Poland.* Springer, 2014, pp. 155–164.

[39] M. Ismail and G.E. Suh, "Quantitative overhead analysis for Python," in *International Symposium on Workload Characterization (IISWC).* IEEE, 2018, pp. 36–47.