Master's Thesis

Empirical Study on Extracting Practical Code Scenarios from Python Textbooks

Hathaichanok Damrongsiri

Program of Information Science and Engineering
Graduate School of Science and Technology
Nara Institute of Science and Technology

Supervisor Professor Kenichi Matsumoto
Software Engineering Lab. (Division of Information Science)

Submitted January 30, 2024

A Master's Thesis
submitted to Graduate School of Science and Technology,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Master of Engineering

Hathaichanok Damrongsiri

Thesis Committee:

Supervisor Kenichi Matsumoto
(Professor, Division of Information Science)
Shoji Kasahara
(Professor, Division of Information Science)
Takashi Ishio
(Professor, Department of Media Architecture, Future University Hakodate)
Raula Gaikovina Kula
(Associate Professor, Division of Information Science)
Kazumasa Shimari
(Assistant Professor, Division of Information Science)

Empirical Study on Extracting Practical Code Scenarios from Python Textbooks*

Hathaichanok Damrongsiri

Abstract

Python serves as a versatile programming language due to its diverse audience. This versatility, however, has a double-edged nature as it implies a multitude of learning scenarios, which can quickly become overwhelming for students. To explore these scenarios in an educational context, the investigation focuses on their usage in Python textbooks. The approach involves the manual curation of 1,017 chapter titles from 76 Python textbooks. The results indicate that coding scenarios are prevalent, making up approximately 39.5% of the content compared to other topics. The scenario content is classified into four types: Application Programming Interfaces, Data and Processing, Graphical User Interfaces, and others. Additionally, a list of 19 Python libraries used in these various scenarios is identified. To assist educators and students, PyEdu, a Python Educational Scenario Visualizer, is showcased, utilizing the results of the empirical study.

Keywords:

Python Programming, Educational Materials, Teaching Methods

*Master's Thesis, Graduate School of Science and Technology, Nara Institute of Science and Technology, January 30, 2024.

Contents

List of Figures

List of Tables

1 Introduction

Python is regarded as one of the most popular and versatile programming languages, with much of its success being accredited to its easy-to-learn and flexible features compared to C++ and Java [2]. It is commonly regarded as the most suitable language for novice programmers to learn [3]. According to a 2021 Anaconda report*, 63% of respondents stated that they used Python frequently or always, while 71% of educators reported teaching machine learning and data science with Python, which has gained popularity due to its ease of use and gentle learning curve. Meanwhile, 88% of students reported being taught Python in preparation for entering the data science and machine learning field.

The success of Python could be attributed to its vast usage by varying audiences. Python plays a vital role in various engineering tasks such as data science, machine learning, and cloud computing [4, 5]. For example, it is used for machine learning, data science, statistics, and even web programming. This can also be viewed as a double-edged sword, implying a multitude of learning scenarios for a student. Prior work [6, 7] has shown that there is more than one way to write Python, and there may even be a Pythonic way to write idiomatic code that is faster and more efficient [8, 9, 10]. Complementing these studies, the goal of this paper is to identify and extract different learning scenarios that may arise from teaching Python in textbooks.

Additionally, prior works [11, 12, 13, 14] cite textbooks as a useful source from which to extract practice examples. Building upon this foundation, the aim extends beyond extracting mere code snippets, focusing instead on garnering a more comprehensive range of examples. Hence, a learning scenario is defined as:

> *a collection of code snippets, libraries, and context presented in an educational manner to understand a specific usage scenario.*

*`https://www.anaconda.com/state-of-data-science-2021`

Table of Contents **Page**

Figure 1.1: An excerpt from the Table of Contents extracted from *A Beginners Guide to Python 3 Programming* [1]

It is assumed that authors of textbooks present their work in an educational manner, distinctly different from the format of question-and-answer resources like Stack Overflow. Likewise, in contrast to blogs and online tutorials, practical scenarios in textbooks often contain much richer information, as they typically encompass an entire chapter. This approach extends beyond merely pulling out examples from a textbook. It proposes a methodology that uses the Table of Contents (ToC) of a textbook to identify and extract learning scenarios. For instance, Figure 1.1 displays the ToC of the Python textbook, *A Beginners Guide to Python 3 Programming* Hunt [1].

By analyzing the chapter titles, one can infer the textbook's content. The book begins with an introduction, followed by fundamental topics in the subsequent chapters. As it progresses, the complexity increases, introducing advanced concepts like iterables and complex data structures such as lists. Finally, the textbook ends with a practical application: Implementing a TicTacToe game. In this context, the TicTacToe game can be viewed as a learning scenario that allows learners to apply various concepts acquired throughout the textbook.

In this paper, the focus of exploration is to understand how textbooks apply Python concepts in practical scenarios, emphasizing the examination at the level of entire chapters. The approach relies on utilizing the Table of Contents within these textbooks to effectively locate and infer practical applications.

To guide this research, three research questions have been posed:

- **(RQ1) To what extent are chapters that describe learning scenarios prevalent in Python textbooks?**
  Motivation: The motivation is to identify the varied contents of a Python textbook. Addressing RQ1 enables the isolation and analysis of diverse learning scenarios employed in the books.
  Results: From the analysis of the 76 textbooks, five categories were emerged: Introduction, Fundamentals, Software process, Scenario, and Abstract titles. It was observed that chapters focusing on practical scenarios, categorized as Scenario, are particularly prevalent, accounting for 39.5% of the content. This trend is especially notable in textbooks focused on Machine learning ($\mathbb{ML}$), Web development ($\mathbb{WD}$), and DevOps ($\mathbb{DO}$).

- **(RQ2) What are the characteristics of chapters that contain learning scenarios?**
  Motivation: The aim is to understand whether and how frequently teachers use applications in a single textbook. Insights can be used to characterize the scenario itself and determine which scenarios are used to target the audience for whom the textbook was written.
  Results: The analysis reveals that Software development textbooks primarily contain Generic scenarios. Textbooks that target Machine learning tend to utilize data and processing scenarios (90.91%). Fittingly, textbooks that target the Computer graphics audience employ such as GUI (66.67%) scenarios.

- **(RQ3) What Python libraries are mentioned in chapters that are identified as learning scenarios?**
  Motivation: Building upon results from RQ1, the goal is to deepen the understanding of how writers practically apply concepts, including identifying which Python libraries are employed for the scenario.
  Results: A list of 19 commonly used Python libraries has been identified across various scenario instances. These libraries are predominantly employed in textbooks targeting the general public ($\mathbb{GE}$). In the dataset, `Django` and `NumPy` emerge as the most frequently utilized ones.

Table 1.1: Collected Dataset of Textbooks

Collected Textbooks from Alexandru et al. [7]	
published period	2001 - 2020
# textbooks	83 textbooks
# textbooks with ToC	76 textbooks
median # per book	11 chapter titles
min. # per book	4 chapter titles
max. # per book	41 chapter titles
# total extracted chapters	1017 chapter titles

Table 1.2: Summary: Textbooks' Target Audience and Title Count

Target Audience	**# Chapter Titles**	**Referenced Textbooks**
Generic (𝔾𝔼)	503	[15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42]
Educational purpose (𝔼𝔻)	152	[43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]
Web development (𝕎𝔻)	132	[55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65]
Data analysis (𝔻𝔸)	104	[66, 67, 68, 69, 70, 71, 72, 73, 74, 75]
Software testing (𝕊𝕋)	32	[76, 77]
Game development (𝔾𝔻)	24	[78, 79]
Others (𝕆𝕋)	21	[80, 81]
Software development (𝕊𝔻)	15	[82, 83]
Machine learning (𝕄𝕃)	13	[84]
Computer graphics (ℂ𝔾)	11	[85]
DevOps (𝔻ℂ)	10	[86]
Total # Chapter Titles	1,017	

As a first step, this study demonstrates that traditional textbooks contain a wealth of information, with learning scenarios being notably diverse. Python programmers can leverage these practical learning scenarios to familiarize themselves with the Python libraries they wish to learn. Additionally, researchers have the opportunity to develop tool support for the large-scale exploitation of this resource. For educators, the study highlights that textbooks offer a wide array of learning scenarios and Python libraries to choose from. Educators can use these scenarios when targeting their audience and creating courses, while adding these scenarios to their existing toolbox of teaching resources.

2 Data Preparation

In this section, the dataset preparation process is described, encompassing the studied textbooks, extraction of chapter titles from the Table of Contents (ToC), and classification of the target audience for each textbook.

2.1 Extracting chapter titles from textbooks.

Table 1.1 provides a concise summary of the datasets collected for this study. To gain insights into the contents of Python textbooks, a dataset obtained from previous research [87] was employed. This dataset included 83 textbooks published between 2001 and 2020. Contact was established with the authors of the aforementioned work, resulting in the acquisition of PDF versions of all these textbooks.

Given that the approach relies on the Table of Contents (ToC), textbooks without a ToC were filtered out. As a result, 76 textbooks met the selection criteria. To extract chapter titles from these textbooks, a semi-automatic approach was employed. Initially, I used the pdfminer tool* for capturing and extracting the chapter titles. However, to address potential issues related to font quality and stylistic variations in textbooks, a manual validation process was subsequently conducted. This validation involved a meticulous comparison of the extracted chapter titles against the originals, as detailed in Table 1.1. The number of chapters in each textbook ranged from 4 to 41, leading to the successful retrieval of 1,017 chapter titles from the 76 textbooks.

*`https://pypi.org/project/pdfminer/`

2.2 Characterizing Textbooks based on Audience

To illustrate the diversity of purposes and the distinct scenarios that may exist for specific textbook audiences, the textbooks were classified based on the intended target audience, as determined by the authors' objectives in creating the textbooks. The methodology for achieving this classification involved two steps:

1. Searches for reviews and descriptions of the textbooks were conducted across various online bookshops, including Amazon reviews[†] and the O'Reilly website[‡].

2. A round-table discussion was adopted, which involved seeking insights from various sources, including an experienced researcher with over ten years of experience in teaching Python.

The information displayed in Table 1.2 provides a summary of eleven distinct categories that represent the intended target audiences for the textbooks under consideration. This table also lists the textbooks that serve as references for each of these identified categories. An observation from the table is that a significant portion of the textbooks were directed at a broad and general readership, constituting a total of 503 chapter titles. In contrast, the category focused on "DevOps" had the fewest number of chapter titles, amounting to 10 titles. The eleven distinct target audience categories are described as follows:

1. Generic ($\mathbb{GE}$): This category pertains to books with content designed for general usage, serving a broad and all-encompassing purpose. For instance, "A Beginner's Guide to Python 3 Programming" [16] falls under this category.

2. Educational purpose ($\mathbb{ED}$): Targeting an audience involved in the realm of education, this category focuses on content where Python is utilized as a tool for both imparting and acquiring knowledge. An example of such a textbook is "Introduction to Computer Science Using Python" [44].

[†] `https://www.amazon.com/b?node=283155`
[‡] `https://www.oreilly.com`

3. Web development ($\mathbb{WD}$): This category is centered around Python web development content. Textbooks falling under this category delve into the intricacies of web development using Python. An example is "Django Design Patterns and Best Practices" [58].

4. Data analysis ($\mathbb{DA}$): Targeting tasks related to data analysis, data structures, and algorithms, this category emphasizes Python's use in these domains. An illustrative textbook is "Learning pandas" [70].

5. Software testing ($\mathbb{ST}$): This category relates to testing within the context of Python software development. Textbooks in this category explore the principles of test-driven development with Python, as seen in "Test-Driven Development with Python" [77].

6. Game development ($\mathbb{GD}$): Focused on the field of game creation, this category encompasses textbooks that guide readers through game development using Python. An example is "Beginning Python Games Development, 2nd Edition" [78].

7. Software development ($\mathbb{SD}$): Tailored for software creation using Python, this category covers various topics such as agile methodologies and building applications with the support of the Python community. An example textbook is "Foundations of Agile Python Development" [82].

8. Machine learning ($\mathbb{ML}$): Centered around machine learning, this category includes textbooks that delve into Python's role in the field of machine learning. An example is "Python Machine Learning" [84].

9. Computer graphics ($\mathbb{CG}$): Focusing on the development of graphical user interfaces (GUIs) using Python, this category includes textbooks that guide readers in creating graphical interfaces with Python. An example is "Python GUI Programming Cookbook" [85].

10. DevOps ($\mathbb{DO}$): This category is related to DevOpspractices and methodologies using Python. Textbooks in this category explore Python's role in practices like DevOps. An example is "Internet of Things with Python" [86].

11. Others ($\mathbb{OT}$): Referring to books that focus on technologies other than Python, this category includes textbooks that may cover technology topics tangential to Python. An example is "Raspberry Pi Cookbook for Python Programmers" [81].

These categories provide a comprehensive framework for classifying textbooks based on their intended target audience and the purpose they serve.

Note that the category names for classification were referenced from a professional Python survey[§].

[§]https://lp.jetbrains.com/python-developers-survey-2021/

Table 2.1: Chapter Contents Distribution Summary by Textbook (Target Audience)

Targeted Audience	**Introduction**	**Fundamentals**	**Software process**	**Scenario**	**Abstract titles**
Generic (GE)	15.5%	32.8%	8.6%	29.0%	14.1%
Educational purpose (ED)	26.3%	15.8%	8.6%	40.8%	8.6%
Web development (WD)	9.9%	12.1%	3.0%	70.5%	4.6%
Data analysis (DA)	23.1%	23.1%	3.9%	49.0%	0.1%
Software testing (ST)	0.0	6.3%	53.1%	40.6%	0.0
Game development (GD)	20.8%	16.7%	0.0	12.5%	50.0%
Others (OT)	19.1%	42.9%	0.0	38.1%	0.0
Software development (SD)	13.3%	26.7%	33.3%	13.3%	13.3%
Machine learning (ML)	15.4%	0.0	0.0	84.6%	0.0
Computer graphics (CG)	9.1%	27.3%	9.1%	54.6%	0.0
DevOps (DO)	10.0%	10.0%	0.0	70.0%	10.0%
Sum # chapter titles	16.7%	24.8%	8.6%	39.5%	10.4%

Note: Scenario Chapters Highlighted in Blue.

3 Findings

In this section, the methodologies employed to answer the research questions are described, along with a discussion of the corresponding results.

3.1 Matching Contents with Target Audience

To effectively understand how Python concepts are applied in textbooks, it was essential to first examine the organizational structure of these textbooks. For this purpose, a qualitative analysis of the contents of 76 Python textbooks was conducted, which involved manually coding 1,017 chapter titles. Specifically, to mitigate subjectivity in this coding, a systematic procedure was followed, drawing on methodologies from previous work [88, 89]. The construction of a taxonomy for Python content categories and the details of the coding process are outlined below.

A random sample of 30% (305) of the 1,017 chapter titles was selected for the pilot construction of the taxonomy, a process informed by the methodology of Wu et al. [89]. The coding team consisted of four individuals: three master's students with over five years of Python programming experience and one professor with more than ten years of experience in Python education and research. The taxonomy construction occurred in two stages. In the first stage, the three master's students engaged in a round-table discussion to familiarize themselves with the content of Python textbooks. They read and re-read the chapter titles and their brief descriptions, taking additional notes where necessary.

In the second stage, these three coders independently assigned initial codes to the chapter titles, annotating them with important information. It was ensured that each title received only one most appropriate label. Subsequently, the professor joined the process as a referee to address and resolve any coding

disagreements. This step was crucial in grouping the initial codes into coherent high-level categories. The team then iteratively reviewed and adjusted these categories, refining the taxonomy through continuous dialogue until unanimous agreement on all categories was reached among the coders and the referee.

3.2 (RQ1) To what extent are chapters that describe learning scenarios prevalent in Python textbooks?

To evaluate and refine the content categories that emerged from the pilot construction, and to ensure a comprehensive understanding, an iterative categorization process was undertaken. This involved categorizing several rounds of random samples, 30 samples per round, until the Kappa agreement score reached a level of 0.80 or higher, indicating almost perfect agreement. In each round where the score fell below 0.80, a thorough review was conducted to resolve any discrepancies.

After three rounds, which included a total of 90 samples, the iterative process concluded successfully with the Kappa score reaching 0.86 in the third iteration. Encouraged by these results, the remaining samples were then classified independently. Through this rigorous process, five distinct types of content were identified:

(a) Introduction The introduction should include any new ideas that the author would like to rely upon the reader. In this example, the book "Python Data Analysis" [72], targets a Generic audience. Specifically, it introduces new specialized topics that are pivotal for beginners, such as Retrieving, Progressing, and Storing Data, and Introduction to the Python's World.

(b) Fundamentals A significant portion of the chapters in the examined textbooks fell into the Fundamentals category. This category is distinct from the Introduction category in that chapter titles typically include terms such as *Fundamental*, *concepts*, *principles*, and *basics*, reflecting a focus on the essential struc-

tures, functions, or foundational facts of Python programming. Such chapters often cover concrete programming examples, including topics like loops, conditional statements (*if*), and general data structures.

For instance, "A Whirlwind Tour of Python" [17] features chapters with titles such as Iteration/Looping and Python Classes, which are indicative of the Fundamentals category. Note that fundamentals do not contain explicit Python libraries in their chapter titles.

(c) Software process The Software process category contains any chapter that describes any software development process like software testing, inspections, and version control. In this example, the book "Becoming a Better Programmer" [90], contains a Generic audience, and includes chapter titles such as Testing Times, Effective Version Control, and Test-Driven Developers.

(d) Scenario Building upon the definition given in the Introduction section, the Scenario category was identified as encompassing chapters that demonstrate practical applications of Python in specific situations. This category is distinct from Fundamentalsin that it focuses more directly on the use of Python in real-world scenarios, rather than on fundamental concepts.

Chapters in this category typically do not discuss software development processes but rather concentrate on actual Python applications. For example, "Data Wrangling with Python" [91] includes chapters on practical tasks like working with Excel files, PDFs, and web scraping. This category also includes more specific functions or libraries in the chapter titles.

(e) Abstract titles The final category encompasses chapter titles that did not align with any of the other established categories. This includes chapter titles that are vague, overly colloquial, or consist of catchy phrases, making it challenging to infer their content. Due to their non-specific nature, these are grouped under this category, which turned out to be one of the most prevalent.

Examples of such chapter titles are 'Divide, Combine, and Conquer,' 'Greed is Good? Prove it,' and 'Hard Problems and (Limited) Sloppiness.' A noteworthy observation is that during a second round of categorization, which involved

a thorough review of the content of each related chapter, it was found that approximately 30.8% of chapters in the Abstract titles category actually aligned more closely with the content of the Scenario category and thus were reclassified accordingly. Despite this, a significant portion, 17.3%, remained uncategorized.

Results Table 2.1 presents an overview of the content distribution within each target audience category in the textbook. The results reveal that scenario content constitutes the largest proportion of chapter content across all the books, accounting for 39.5% of the total chapter content. In contrast, the software process content represents the smallest fraction, making up only 8.6% of the overall content.

A closer examination of the findings indicates that the Machine learning (ML), Web development (WD), and DevOps (DO) books exhibit the highest prevalence of scenario content within their chapters, with percentages of 84.6%, 70.5%, and 70%, respectively. Conversely, the Game development (GD) and Software development (SD) textbooks feature the lowest proportion of scenario content, comprising only 12.5% and 13.3%, respectively.

RQ1 Summary: From the analysis of the 76 textbooks, five categories were emerged: Introduction, Fundamentals, Software process, Scenario, and Abstract titles. It was observed that chapters focusing on practical scenarios, categorized as Scenario, are particularly prevalent, accounting for 39.5% of the content. This trend is especially notable in textbooks focused on Machine learning (ML), Web development (WD), and DevOps (DO).

3.3 (RQ2) What are the characteristics of chapters that contain learning scenarios?

In this section, the focus will be on presenting the approach and results for addressing RQ2.

Approach To answer RQ2, a manual round-table discussion was conducted to classify different scenarios. This approach closely mirrored the methodology

Table 3.1: A Taxonomy of Scenario Instances

Scenario Instances	**Rationale**	**Example of Chapter Titles**
Application Programming Interfaces (𝔸ℙ𝕀)	Focuses on integration and data processing, including web service integration and software communication.	Web APIs with hug, PyMySQL Module, Multiprocessing
Data and Processing (𝔻ℙ)	Focuses on data manipulation, calculation, and modeling.	Curried Functions, Built-In Data Structures, Tracking User Actions
Graphical User Interface (𝔾𝕌𝕀)	Focuses on visual representation and provides visualization tools for data.	Graphical User Interfaces, Computer Generated Art, Graphing with Matplotlib pyplot,
Others/Generic (𝔾𝔼)	Covers scenarios not fitting into the above categories, often for specialized or unique tasks.	Easier Python Packaging with flit, Command-Line Applications, Dates and Times

and classification method that were successfully employed in RQ1, ensuring a consistent and rigorous analysis.

Additionally, the scenarios were categorized based on the specific developer roles identified from the developer survey*. The main focus was on how Python is used by developers. By looking at the developer roles identified in the survey, the analysis aimed to reveal the different ways Python is applied in real-world situations. This thorough approach provided a deeper understanding of Python's roles in various developer communities and made the exploration of RQ2 more insightful.

Results Table 3.1 illustrates the taxonomy of scenario instances. Four distinct types of scenarios have been identified, comprising Application Programming Interfaces (𝔸ℙ𝕀), Graphical User Interface (𝔾𝕌𝕀), Data and Processing (𝔻ℙ), as well as Others/Generic (𝔾𝔼).

- Application Programming Interfaces (𝔸ℙ𝕀) represents the scenario that pertains to the integration or processing of applications. This includes processing tasks such as integrating web services and facilitating data exchange between different software components.

*`https://lp.jetbrains.com/python-developers-survey-2021/`

- Data and Processing ($\mathbb{DP}$) represents the scenario related to the manipulation, calculation, and modeling of data. This also includes tasks like data cleaning, transformation, statistical analysis, and data concurrency.

- Graphical User Interface ($\mathbb{GUI}$) represents the scenario that focuses on the illustration of components or outputs, such as providing visualization tools for data representation and integrating libraries like Matplotlib for graphical displays.

- Others/Generic ($\mathbb{GE}$) represents the category of scenarios that don't neatly fit into the aforementioned classifications, including generic scenarios that encompass a broader scope of content. Through this comprehensive approach, insights are sought into how different scenarios are applied within the various target audiences of the book.

Figure 3.1 presents the distribution of the different scenario categories that are targeted by different types of audiences using a heatmap visualization. The result shows the correlation between the various scenario instances and the target audience of the textbook. The findings reveal that in the context of the Software Development textbook, all scenario instances (100%) are categorized as generic scenarios, which inherently suggests that they encompass broad subject matter.

For example, the chapter 'The IDE: Eclipsing the Command Line' from 'Foundations of Agile Python Development' textbook [82]. Conversely, in the Machine Learning book, the majority of scenarios (90.91%) are focused on data and processing. Another example is the chapter 'Working with Unlabeled Data - Clustering Analysis' from the 'Python Machine Learning' textbook[84].

In the Computer Graphics book, 66.67% of the scenarios pertain to graphic user interface-related content such as the 'Creating the GUI Form and Adding Widgets' from the 'Python GUI Programming Cookbook' textbook [85].

RQ2 Summary: Software development textbooks contain only Generic scenarios. Textbooks that target Machine learning tend to utilize data and processing scenarios (90.91%). Fittingly, textbooks that target the Computer graphics audience employ such as GUI (66.67%) scenarios.

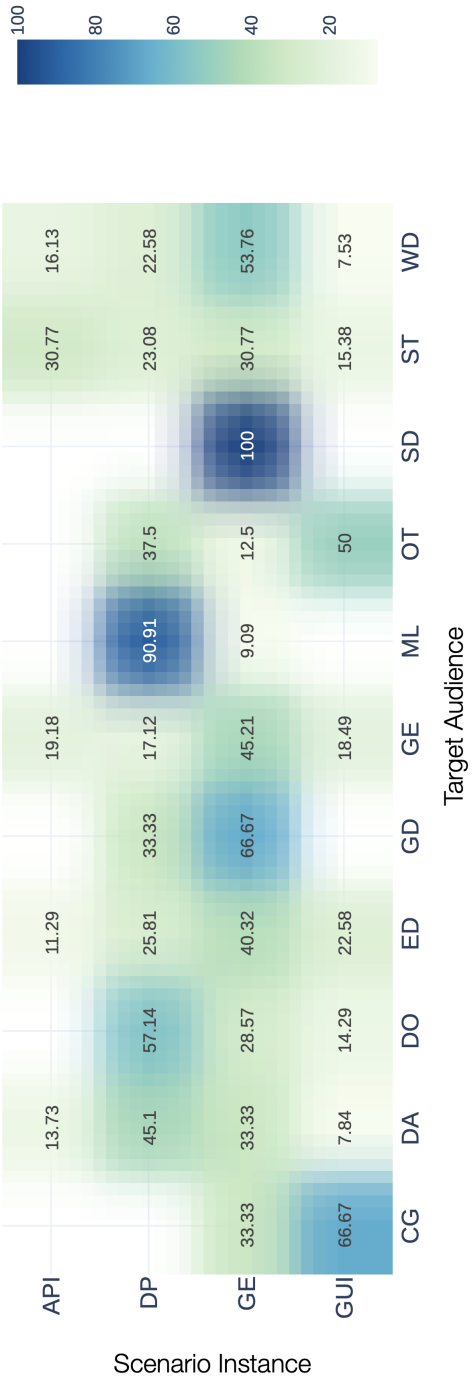

Figure 3.1: The proportion of Scenario instances for each type of textbook (based on target audience)

Table 3.2: List of Python libraries Extracted from Chapter Titles of the Scenario Categories

Target Audience	**Scenario Instance**	**Library**
Generic (𝔾𝔼)	𝔸ℙ𝕀	`PyMySQL` [93], `RxPy` [94]
	𝔾𝕌𝕀	`Matplotlib` [95], `PyDraw` [96], `wxPython`[97]
	𝔻ℙ	`Pygame`[98]
	𝔾𝔼	`wxPython` [97], `Flit` [99], `IronPython` [100], `PyBoxes`[101], `Tkinter` [102]
Educational purpose (𝔼𝔻)	𝔸ℙ𝕀	`Matplotlib` [95], `Scikit-learn` [103]
	𝔾𝔼	`NumPy`[92], `Cython`[104]
Web development (𝕎𝔻)	𝔸ℙ𝕀	`Django` [105]
	𝔾𝔼	`Django` [105]
Data analysis (𝔻𝔸)	𝔸ℙ𝕀	`Scikit-learn` [103]
	𝔻ℙ	`NumPy` [92], `MapReduce` [106], `Pandas`[107]
Software testing (𝕊𝕋)	𝔻ℙ	`Selenium` [108]
Computer graphics (ℂ𝔾)	𝔾𝕌𝕀	`PyOpenGL` [109], `PyGLet` [110]

3.4 (RQ3) What Python libraries are mentioned in chapters that are identified as learning scenarios?

In this section, the focus is on discussing the results for addressing RQ3.

Approach To answer RQ3, an exploration was conducted concerning the utilization of Python libraries as reference points across various scenario instances to differentiate target audiences. This involved a manual round-table discussion process, similar to what was employed in RQ2. Specifically, the Scenario content classification from RQ1 was analyzed. Subsequently, for each title, specific Python libraries were extracted.

For instance, `NumPy`[92] was extracted for the title 'Exploratory and Predictive Data Analysis with NumPy' from the book "NumPy Cookbook, 2nd Edition"[47]. The goal is to understand whether or not certain libraries are more prevalent in different textbooks.

Results Table 3.2 displays the introduction of Python libraries in each textbook for specific target audiences and scenarios. Two key observations can be made from these findings.

First, it was possible to identify 19 distinct Python libraries , as presented in the table. Secondly, the results indicate that the majority of Python libraries were utilized in generic books, totaling 13 libraries. For example, in scenario instances targeting the Generic ($\mathbb{GE}$) audience within the Graphical User Interface ($\mathbb{GUI}$) category, three libraries were introduced: `Matplotlib` [95], `PyDraw` [96], and `wxPython` [97].

It is noteworthy that in the context of the web development book, the only library introduced was `Django` [105]. The most frequently introduced library across all books was `NumPy` [92], which serves as a foundational package for scientific computing in Python, offering an array of mathematical functions, random number generators, and more with accessible high-level syntax. Specifically, `NumPy` appeared in textbooks targeting both the Generic ($\mathbb{GE}$) and Educational ($\mathbb{ED}$) audiences.

A more detailed examination of educational purpose books and generic scenarios revealed that `Cython`[104], a compiler library simplifying the creation of C extensions for Python, and `NumPy` played essential roles. These two libraries are of particular interest because they are not part of the standard Python libraries (i.e., they are PyPI libraries). Their status as third-party libraries suggests the Python community's involvement with other programming languages (e.g., Cython from C programming) and paradigms (e.g., statistical analysis). This further emphasizes the diversity of the Python community and its connections to various programming domains.

In summary, these two Python libraries serve as foundational tools that educators can employ in their subsequent studies. Interestingly, no Python libraries were reported as being used across different target audiences.

RQ3 Summary: A list of 19 commonly used Python libraries has been identified across various scenario instances. These libraries are predominantly employed in textbooks targeting the general public ($\mathbb{GE}$). In the dataset, `Django` and `NumPy` emerge as the most frequently utilized ones.

4 Python Educational Scenario Visualizer: PyEdu

To demonstrate the practical applications of the findings, an interactive application prototype has been developed to showcase knowledge extracted from textbooks.

This prototype, detailed in Figure 4.1, includes a comprehensive overview, features interactive elements, and provides output tools for enhanced user engagement and deeper insight into the subject matter. At its core is a Streamlit-based application*, designed for Scenario Analysis, which connects to the finding database and offers an interactive user interface.

The prototype is accessible at `https://github.com/ploychanok/python-scenario`.

Note that this prototype utilizes only a subset of six textbooks, as referenced in [15], [17], [19], [68], [79], and [111].

4.1 Interface Design

① Sidebar In this section, a radio button is utilized, offering two options: `'By Audience'` and `'By Library'`. These options are followed by two select boxes. When the user selects the `'By Audience'` option, the two select boxes allow for data filtering based on the Target Audience and Chapter selection. Conversely, if the user chooses `'By Library'`, the select boxes enable filtering based on Python libraries and Chapter selection.

*`https://streamlit.io`

② Visualisation of Taxonomy For this section, the utilization of Plotly[†] for data visualization is discussed. A Sankey diagram has been chosen for its effective representation of various entities and their interrelationships. When the `'By Audience'` option is selected, the graph displays the Target Audience on the left, Scenario Instances in the middle, and Libraries on the right. Conversely, when `'By Library'` is chosen, the graph presents the Library on the left, Scenario Instances in the middle, and Target Audiences on the right.

③ Scenario Contents This section will contain several of components which includes:

- Metadata: Textbook title, Chapter title, and Related library or Libraries.
- Chapter Summary: Provides a summary of the selected chapter.
- Code Snippet: Includes code snippets found in the selected chapter.

The chapter summaries, code snippets, and their explanations are extracted using pdfgear[‡] and ChatGPT-4[§], with correctness manually verified by the author.

4.2 Interactive Walk through

Select the viewpoint As illustrated in Figure 4.1, the sidebar (①) incorporates two primary functionalities to assist users in navigating through scenario viewpoints. These are: Firstly, it offers the option to select a viewpoint based on various audience groups, denoted as `'By Audience'`, and secondly, it provides the option to select a viewpoint tailored specifically to libraries, indicated as `'By Library'`. Following the selection, the main content area dynamically updates to display pertinent information, which includes the ② graph and the ③ scenario referenced in Figure 4.1.

[†]`https://plotly.com/python/`
[‡]`https://www.pdfgear.com`
[§]`http://chat.openai.com/`

Searching `'By Audience'` This option presents users with a list of target audiences, as detailed in Table 2.1. Upon selecting a target audience, the second select box dynamically filters the chapters relevant to the selected audience. The filtered results are then displayed in the Graph (②) and Scenario (③) sections.

Searching `'By Library'` This option operates similarly to Searching `'By Audience'` but focuses on Python libraries. Users are provided with a list of Python libraries, similar to the examples in Table 3.2. Selecting a Python library prompts the system to filter chapters associated with the selected library, with the outcomes displayed in the Graph (②) and Scenario (③) sections.

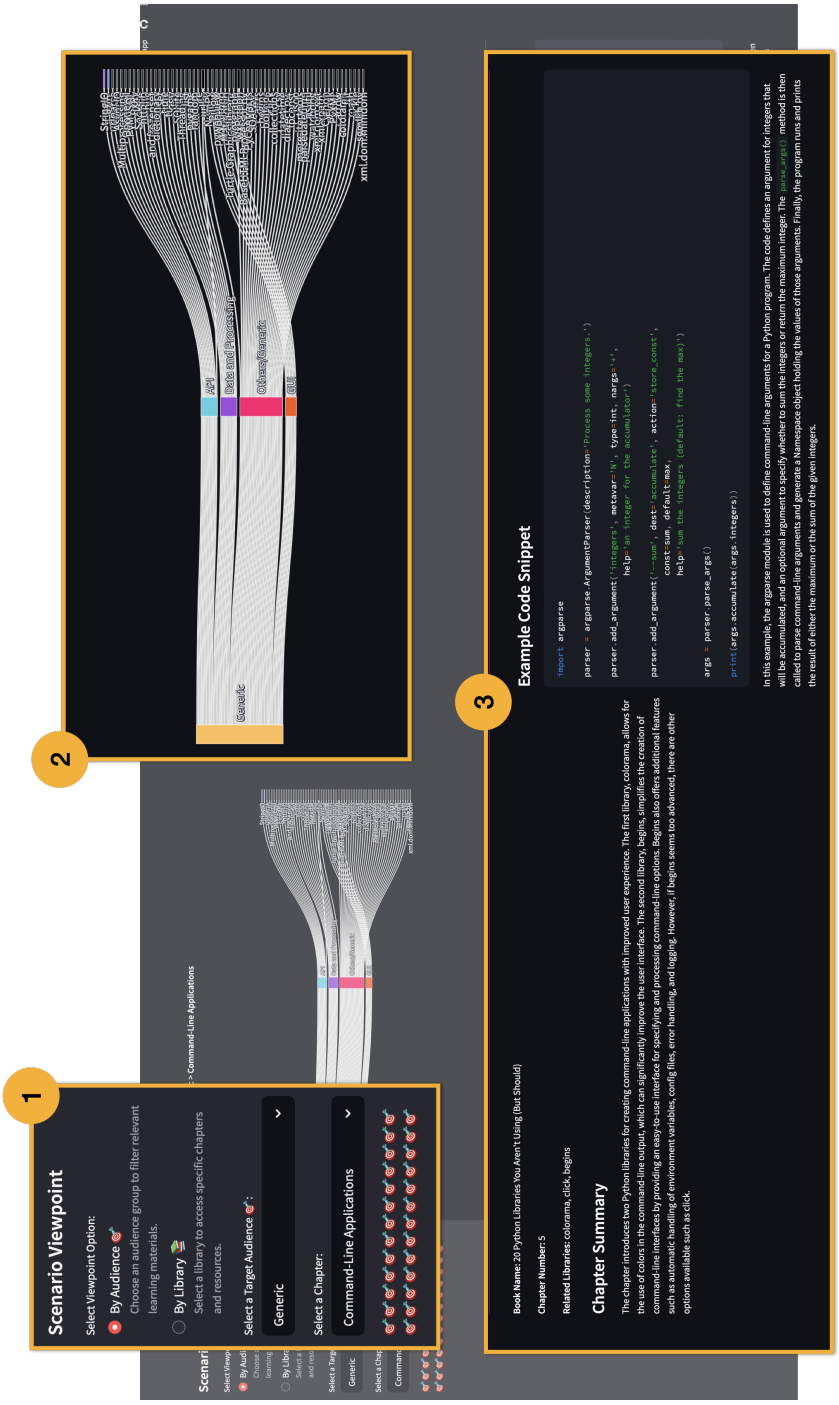

Figure 4.1: Screenshot of the PyEdu Interface

Demo available at: `https://github.com/ploychanok/python-scenario`

5 Lessons Learned

In this section, lessons learned and implications derived from the results are presented. These insights are intended to benefit practitioners, researchers, and educators, not only in the field of Python but also in the broader field of programming.

5.1 For Students of Python

> *Textbooks contain a wealth of information ready to be exploited.*

The results presented in this study are encouraging for programmers, as they indicate that textbooks do include chapters focused on practical applications, specifically real-world cases. As demonstrated in RQ1, the evidence illustrates that through the analysis of chapter titles, successful identification of those chapters primarily designed for scenario-based practice was achieved. This reaffirms the continued value of Python textbooks as a learning resource. Furthermore, as demonstrated in RQ2, it has been shown that the scope of scenarios covered in these textbooks is indeed broad, catering to a wide range of scenarios in generic Python textbooks.

> *It is better to go for the generic textbooks, unless you are trying to find the right learning scenario.*

For RQ3, a compilation of Python libraries has been prepared, offering insights into where developers might want to focus their learning efforts. The next step involves tailoring scenarios to align specifically with the requirements and interests of the audience or students who wish to gain a deeper understanding of the tasks they can accomplish using Python.

Textbooks contain abstract titles, with authors personal opinions

An interesting observation is that textbooks featuring Advanced-Level titles often include personal advice from the author to the reader, a feature not commonly found in other formats, such as online resources or courses. Consequently, an important avenue for future research involves conducting a comparative study that examines chapter titles in textbooks alongside other educational materials, including content created by both educators and software practitioners, as well as software documentation.

5.2 For Researchers and Practitioners

Opportunities for tool support and comparison against other software documentation resources.

Currently, the results presented in this paper (RQ1, RQ2, RQ3) have been obtained through manual analysis. While manual analysis ensures higher quality, it is time-consuming, susceptible to human error, and not scalable for larger datasets. Therefore, the next step involves providing tool support. As such, future work may include the automatic extraction of chapter titles. Such methods could encompass the use of image processing of the Table of Contents. However, it's important to note that this task won't be straightforward, as the Table of Contents can vary in both font size and style. Anticipated challenges in this regard will be addressed as part of future work.

Future work of digging deeper into the contents of each textbook chapters.

All the results obtained in this study (RQ1, RQ2, RQ3) provide only a superficial overview of the contents found within these textbooks. Due to the exclusive focus on analyzing chapter titles, a limitation of this work is the inability to gain a comprehensive understanding of the actual contents of the textbooks. For future research, the aim is to delve deeper and achieve a more qualitative understanding of the actual chapters. This would involve extracting the textual and code example contents to gain insight into how scenarios are organized for the

learners. Particularly in the case of RQ3, it would be interesting to investigate how these code examples differ from or resemble other sources, such as various forms of software documentation, content on Stack Overflow, blogs, and online tutorials.

5.3 For Educators

There exists a wide array of learning scenarios and Python libraries from which to choose.

Apart from illustrating the diverse range of learning scenarios, educators can leverage the findings from RQ1 and RQ2 to identify which textbook scenarios are suitable for teaching Python. The results from RQ3 can also assist educators in integrating the listed Python libraries and examples into their curriculum. As a stepping stone, this extraction of learning scenarios could be helpful for educators, especially if they are teaching a specific target audience. Additionally, it helps them understand that Python indeed caters to a diverse set of programmers.

Identifying essential Python libraries used in multiple scenarios.

The results in RQ3 reveal a set of 19 Python libraries explicitly mentioned in the textbook chapter titles. Among these findings, it becomes apparent that `Matplotlib` and `NumPy` have been identified in multiple textbooks targeting different audiences. The next step involves extracting code examples from these scenario chapters to understand the concepts taught within these scenarios and to gain insight into the motivations of the authors of Python textbooks.

6 Threats To Validity

Threats to the validity of this study have been categorized into three distinct categories: external threats, construct threats, and internal threats.

6.1 External Threats

External threats to the validity of the study center around the ability to generalize the results, particularly due to the use of a curated list of Python textbooks. Two potential threats are summarized. The first threat arises from the fact that the experiment is evaluated solely within the context of the Python language. Therefore, the observations may vary across different programming languages (e.g., Java and C). Nevertheless, it is important to note that Python has gained widespread prominence, emerging as a dominant language in various fields, including education and software development. The significance of Python as a representative language justifies this exploration.

The second potential threat is related to the selection of textbooks. In this study, a total of 76 textbooks were analyzed, which may raise concerns about the generalizability of the findings to all textbooks. However, it's important to note that the sample size, including 1,017 chapter titles across a wide range of topics, provides confidence that the studied textbooks are broadly utilized by educators. Furthermore, all the textbooks have been made open and accessible for readers and for replicating the dataset. In future work, there are plans to extend the framework to encompass additional programming languages. Additionally, the aim is to include a more diverse range of textbooks covering various fields and topics related to Python programming. As an initial study, these insights are considered crucial, laying the foundation for further research into the systematic utilization of information in textbooks to support educators and learners.

6.2 Construct Threats

Construct threats refer to the degree to which measurements accurately capture the intended aspects of the study. A potential threat could arise when extracting the tables of contents from the studied textbooks, as automated methods may not always retrieve them accurately.

To migrate this threat, the author manually inspected all retrieved tables of contents by comparing them against the original textbooks. Therefore, we believe that the dataset we have constructed is valuable for mining and yielding insightful results. In order to create a much larger scale study, an automated method might be required with a larger accuracy. This is seen as potential future work.

6.3 Internal Threats

Internal threats denotes the approximate truth about inferences regarding cause-effect or causal relationships. To address the research questions posed in this study, a series of qualitative analyses were conducted to gain insight into the contents and sequences of textbooks. One potential internal threat arises from the approach employed, which solely involves manual analysis of chapter titles. This presents a threat as chapter titles may not always perfectly capture the content of the chapters. Hence, instances are likely to be mislabeled due to the subjective nature.

To mitigate this threat, the manual coding was systematically conducted with the involvement of multiple individuals, including experienced educators. Additionally, Kappa agreement scores were calculated to ensure coding quality. Once the scores indicated nearly perfect or substantial agreement, coding was independently carried out. For example, in RQ1, the Kappa scores ultimately reached 0.86, indicating strong inter-rater reliability.

7 Related Work

In this section, relevant literature pertaining to Python education, Pythonic idioms, and the study of programming textbooks will be discussed.

7.1 Python Education

To facilitate more efficient learning of Python for novice programmers, several researches have been conducted to propose learning strategies and tools. For example, Wiese et al. [112] conducted research that established a link between code readability, structure, and comprehension among Python novices. They suggested that novices may benefit from lightweight tools that identify common patterns and offer "expert" solution. Hosseini et al. [113] explored the value of engaging features in learning Python programming examples and introduced a tool named PCEX. Their results indicated a positive impact on students' engagement, problem-solving performance, and overall learning experience.

Comparative studies comparing the learning of Python to other programming languages have also been conducted [2, 114]. For instance, Koulouri et al. [115] found that using a syntactically simple language like Python facilitated students' understanding of programming concepts more effectively compared to more complex languages like Java.

> *Although the result of this research shares the overarching aim of offering insights into learning strategies, the approach is distinctive in its exclusive focus on the Python programming language, with no consideration of other languages such as Java.*

7.2 Pythonic Coding

Python is renowned for its pythonic idioms, representing notable programming styles and features [6]. Previous studies on Python idioms have primarily explored them through books or online resources. For example, Alexandru et al. [7] compiled a catalog of 19 pythonic idioms from various books and assessed their performance and readability. Farooq and Zaytsev [8] extended these findings to a broader collection of idioms, reinforcing the observations made by Alexandru et al. Sakulniwat et al. [9] conducted a case study to visualize the usage of open idioms in open-source projects over time, demonstrating that developers tend to adopt idiomatic code. Preliminary experiments by Leelaprute et al. [10] suggested that writing in pythonic idioms could result in memory and time savings.

> *Unlike those studies, the focus of this research extends beyond pythonic styles and encompasses all programming scenarios in Python.*

To support the learning of pythonic idioms, tools like Teddy, developed by Phan-Udom et al. [116], have been introduced for checking idiom usage, demonstrating high precision in detecting idiomatic and non-idiomatic Python code. Zhang et al. [117] designed an automatic refactoring tool to make Python code idiomatic, highlighting its practicality and usefulness.

> *The exploratory study, while conducted manually, lays the groundwork for future efforts to develop tools aimed at assisting learners in various scenarios. This expansion may potentially include a broader range of textbooks and other learning materials, such as online tutorials, blogs, and teaching curricula.*

7.3 Analysis of Programming Textbooks

Textbooks are a crucial component of teaching introductory programming, serving as major sources of example programs and references for solving specific problems [11, 12, 13]. Studies have examined their role in education, focusing on various programming languages. For example, Börstler et al. [14] evaluated the quality of object-oriented example programs from popular Java textbooks,

advising caution in using textbook examples directly. Mazumder et al. [118] reported a lack of explanatory diagrams for variables, arrays, and objects in 15 commonly used introductory Java textbooks. Almansoori et al. [119] analyzed security topics and unsafe function usage in textbooks used in top US universities, finding many did not adequately warn about or teach safe usage of these functions.

Efforts like those by Alpizar-Chacon et al. [120], who integrated textbooks with smart interactive content, and Huang et al. [121], who proposed a framework to construct knowledge graphs from textbooks, are enhancing textbook effectiveness. Wu et al. [122] introduced a programming language learning service, complementing textbook knowledge with Stack Overflow posts. They demonstrated the effectiveness of their weakly supervised link approach in matching posts with textbook chapters.

> *This work extracts valuable information from Python textbooks, with a specific focus on analyzing the Table of Contents and characterizing the learning scenarios in a novel way.*

8 Conclusion and Future Outlook

The approach of this study involved extracting learning scenarios based on the Table of Contents (ToC) from various Python textbooks. By categorizing these textbooks into distinct target audiences, valuable insights into the versatility of Python were provided, encompassing scenarios ranging from education and Web development to Data analysis and Software testing.

The results are promising, indicating that learning scenarios (i.e., Scenario) are prevalent, comprising approximately 39.5% when compared to other title contents. Furthermore, the scenario contents were characterized into four types, including Application Programming Interfaces ($\mathbb{API}$), Data and Processing ($\mathbb{DP}$), Graphical User Interfaces ($\mathbb{GUI}$), and others. Additionally, a list of 19 Python libraries used in these diverse scenarios was identified.

This study represents just the beginning and offers significant potential for Python programmers, researchers, and educators, enabling them to explore practical applications and harness the full capabilities of Python. Future work includes delving deeper into the contents of chapter titles, automating the extraction process, and evaluating the usage of learning scenarios, among other potential avenues for research.

Acknowledgement

I'm truly grateful to some amazing people who've been with me throughout my journey in achieving my Master's degree.

First and foremost, I extend my heartfelt thanks to Professor Kenichi Matsumoto, my thesis advisor. His support has been nothing short of exceptional. Not only did he graciously welcome me into the Software Engineering Laboratory, but he also provided unwavering guidance and encouragement every step of the way throughout my Master's program. His mentorship has been invaluable, and I am deeply appreciative of the opportunities he has opened up for me.

I also want to express my sincere appreciation to my co-supervisor, Associate Professor Raula Gaikovina Kula. His influence on my research journey cannot be overstated. With great patience and expertise, he guided me through the intricacies of conducting research and writing scholarly papers. I am particularly grateful for the incredible opportunity he facilitated, allowing me to participate in the enriching exchange program at the University of Melbourne. This experience truly broadened my perspective and enhanced my academic and personal growth.

A special mention and heartfelt gratitude go to Assistant Professor Dong Wang, whose contributions and guidance during my research were indispensable. He provided much-needed clarity and direction precisely when I needed it most. His insights and expertise significantly elevated the quality of my work, and I am deeply thankful for his unwavering support.

Beyond my advisors, I am profoundly grateful to the members of my thesis committee: Professor Shoji Kasahara, Professor Takashi Ishio, and Assistant Professor Kazumasa Shimari. Their insightful comments and constructive suggestions have been a tremendous help in refining the depth and quality of my thesis. Their collective wisdom has been instrumental in shaping my research journey.

I consider myself exceptionally fortunate to have had the opportunity to work with Associate Professor Christoph Treude during my time at the University of Melbourne. His profound expertise in the field was a major boost to my research endeavors, and I am grateful for the invaluable lessons I learned under his mentorship.

I can't forget to mention my wonderful friends and lab mates, who have been an incredible support system throughout this journey. Together, we've shared countless laughs, overcome challenges, and celebrated successes. Their camaraderie and encouragement have made this journey all the more enjoyable and memorable. I couldn't have asked for better companions on this academic adventure.

Last but most certainly not least, I'm forever indebted to my family. Their unwavering support and belief in my dreams, especially in allowing me to study abroad, have been nothing short of incredible. I owe them a debt of gratitude that words can hardly express. Their constant encouragement has been my driving force, and I am deeply thankful for the love and faith they have bestowed upon me.

Bibliography

[1] J. Hunt, *A Beginners Guide to Python 3 Programming*, 1st ed. Springer, 2019.

[2] S. Khoirom, M. Sonia, B. Laikhuram, J. Laishram, and T. D. Singh, "Comparative analysis of python and java for beginners," *Int. Res. J. Eng. Technol*, vol. 7, no. 8, pp. 4384–4407, 2020.

[3] A. Bogdanchikov, M. Zhaparov, and R. Suliyev, "Python to learn programming," in *Journal of Physics: Conference Series*, vol. 423, no. 1. IOP Publishing, 2013, p. 012027.

[4] K. Srinath, "Python–the fastest growing programming language," *International Research Journal of Engineering and Technology*, vol. 4, no. 12, pp. 354–357, 2017.

[5] A. Nagpal and G. Gabrani, "Python for data analytics, scientific and technical applications," in *2019 Amity international conference on artificial intelligence (AICAI)*. IEEE, 2019, pp. 140–145.

[6] J. J. Merchante and G. Robles, "From python to pythonic: Searching for python idioms in github," in *Seminar Series on Advanced Techniques and Tools for Software Evolution (SATToSE)*, 2017.

[7] C. V. Alexandru, J. J. Merchante, S. Panichella, S. Proksch, H. C. Gall, and G. Robles, "On the usage of pythonic idioms," in *Proceedings of the 2018 ACM SIGPLAN international symposium on new ideas, new paradigms, and reflections on programming and software*, 2018, pp. 1–11.

[8] A. Farooq and V. Zaytsev, "There is more than one way to zen your python," in *Proceedings of the 14th ACM SIGPLAN International Conference on Software Language Engineering*, 2021, pp. 68–82.

[9] T. Sakulniwat, R. G. Kula, C. Ragkhitwetsagul, M. Choetkiertikul, T. Sunetnanta, D. Wang, T. Ishio, and K. Matsumoto, "Visualizing the usage of pythonic idioms over time: A case study of the with open idiom," in *2019 10th International Workshop on Empirical Software Engineering in Practice (IWESEP)*. IEEE, 2019, pp. 43–435.

[10] P. Leelaprute, B. Chinthanet, S. Wattanakriengkrai, R. G. Kula, P. Jaisri, and T. Ishio, "Does coding in pythonic zen peak performance? preliminary experiments of nine pythonic idioms at scale," in *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*, 2022, pp. 575–579.

[11] J. Börstler, M. S. Hall, M. Nordström, J. H. Paterson, K. Sanders, C. Schulte, and L. Thomas, "An evaluation of object oriented example programs in introductory programming textbooks," *ACM SIGCSE Bulletin*, vol. 41, no. 4, pp. 126–143, 2010.

[12] M. de Raadt, R. Watson, and M. Toleman, "Textbooks: under inspection," University of Southern Queensland, Tech. Rep., 2005.

[13] W. J. Fitzgerald, J. Elmore, M. Kung, and A. J. Stenner, "The conceptual complexity of vocabulary in elementary-grades core science program textbooks," *Reading Research Quarterly*, vol. 52, no. 4, pp. 417–442, 2017.

[14] J. Börstler, M. Nordström, and J. H. Paterson, "On the quality of examples in introductory java textbooks," *ACM Trans. Comput. Educ.*, vol. 11, no. 1, feb 2011.

[15] C. Hattingh, *20 Python Libraries You Aren't Using (But Should)*. O'Reilly Media, Inc., 2016.

[16] J. Hunt, *2019 Book A Beginner's Guide To Python 3 Program*. Springer Cham, 2019.

[17] J. VanderPlas, *A Whirlwind Tour of Python.* O'Reilly Media, Inc., 2016.

[18] P. Goodliffe, *Becoming a Better Programmer: A Handbook for People Who Care About Code*, 1st ed. O'Reilly Media, 2014.

[19] M. Pilgrim, *Dive Into Python 3.* Apress Berkeley, CA, 2010.

[20] B. Slatkin, *Effective Python: 59 Specific Ways to Write Better Python.* Addison-Wesley Professional, 2015.

[21] M. Jaworski and T. Ziade, *Expert Python Programming.* Packt Publishing, 2016, 536 pages.

[22] L. Ramalho, *Fluent Python: Clear, Concise, and Effective Programming.* O'Reilly Media, 2015.

[23] D. Mertz, *Picking a Python Version: A Manifesto.* O'Reilly Media, Inc., 2015.

[24] P. Barry, *Head First Python: A Brain-Friendly Guide.* O'Reilly Media, 1st edition, 2010.

[25] M. Pirnat, *How to Make Mistakes in Python.* O'Reilly Media, Inc., 2015.

[26] B. Lubanovic, *Introducing Python: Modern Computing in Simple Packages.* O'Reilly Media, 1st edition, 2014.

[27] M. Lutz, *Learning Python, 5th Edition.* O'Reilly Media, 5th edition, 2013.

[28] R. Gupta, *Making Use of Python 1st.* Wiley, 1st edition, 2002.

[29] R. van Hattem, *Mastering Python: Master the art of writing beautiful and powerful Python by using all of the features that Python 3.5 offers.* Packt Publishing, 2016.

[30] M. Alchin and J. B. Browning, *Pro Python.* Apress, 2nd edition, 2014.

[31] A. Harris, *Pro IronPython (Expert's Voice in .NET) 1st ed. Edition.* Apress, 1st ed. edition, 2009.

[32] M. Alchin, *Pro Python (Expert's Voice in Open Source) 1st ed. Edition.* Apress, 1st ed. edition, 2010.

[33] L. Sneeringer, *Professional Python 1st Edition.* Wrox, 1st edition, 2015.

[34] A. Martelli, A. Ravenscroft, and D. Ascher, *Python Cookbook Second Edition.* O'Reilly Media, Second edition, 2005.

[35] E. Matthes, *Python Crash Course: A Hands-On, Project-Based Introduction to Programming 1st Edition.* No Starch Press, 1st edition, 2015.

[36] J. R. Briggs, *Python for Kids: A Playful Introduction To Programming.* No Starch Press, 1st edition, 2012.

[37] M. Venkitachalam, *Python Playground: Geeky Projects for the Curious Programmer 1st Edition.* No Starch Press, 1st edition, 2015.

[38] M. Lutz, *Python Pocket Reference: Python In Your Pocket 5th Edition.* O'Reilly Media, 5th edition, 2014.

[39] J. Ingrassellino, *Python Projects for Kids.* Packt Publishing, 2016.

[40] A. Tigeraniya, *Python Unlocked.* Packt Publishing, 2015.

[41] T. Hall and J.-P. Stacey, *Python 3 for Absolute Beginners (Expert's Voice in Open Source) 1st ed. Edition.* Apress, 1st ed. edition, 2009.

[42] A. B. Downey, *Think Python: How to Think Like a Computer Scientist 2nd Edition.* O'Reilly Media, 2nd edition, 2015.

[43] T. Antao, *Bioinformatics with Python Cookbook First Published Edition.* Packt Publishing; First Published edition, 2015.

[44] C. Dierbach, *Introduction to Computer Science Using Python: A Computational Problem-Solving Focus 1st Edition.* Wiley; 1st edition, 2012.

[45] L. F. Martins, *IPython Notebook Essentials.* Packt Publishing, 2014.

[46] R. Johansson, *Numerical Python: Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib 2nd ed. Edition.* Apress; 2nd ed. edition, 2018.

[47] I. Idris, *NumPy Cookbook, 2nd Edition.* Packt Publishing, 2015.

[48] ——, *NumPy Cookbook.* Packt Publishing, 2012.

[49] M. L. Hetland, *Python Algorithms: Mastering Basic Algorithms in the Python Language 2nd ed. Edition.* Apress; 2nd ed. edition, 2014.

[50] G. Zaccone, *Python Parallel Programming Cookbook.* Packt Publishing, 2015.

[51] M. L. Hetland, *Python Algorithms: Mastering Basic Algorithms in the Python Language (Expert's Voice in Open Source) 1st ed. Edition.* Apress; 1st ed. edition, 2010.

[52] N. H. Tollervey, *Python in Education.* O'Reilly Media, Inc., 2015.

[53] E. Bressert, *SciPy and NumPy 1st Edition.* O'Reilly Media; 1st edition, 2012.

[54] H. P. Langtangen, *Python Scripting for Computational Science (Texts in Computational Science and Engineering, 3) 3rd Edition.* Springer; 3rd edition, 2007.

[55] N. George, *Beginning Django CMS 1st ed. Edition.* Apress; 1st ed. edition, 2015.

[56] S. Newman, *Django 1.0 Template Development.* Packt Publishing, 2008.

[57] A. Mele, *Django By Example.* Packt Publishing, 2015.

[58] A. Ravindran, *Django Design Patterns and Best Practices.* Packt Publishing, 2015.

[59] J. Solorzano and M. Lavin, *Lightweight Django: Using REST, WebSockets, and Backbone 1st Edition.* O'Reilly Media; 1st edition, 2014.

[60] M. A. Russell, *Mining the Social Web.* O'Reilly Media; First Edition, 2011.

[61] ——, *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More Second Edition.* O'Reilly Media; Second edition, 2013.

[62] M. Alchin, *Pro Django (Expert's Voice in Web Development) 2nd Edition.* Apress; 2nd edition, 2013.

[63] M. Pippi, *Python for Google App Engine.* Packt Publishing, 2015.

[64] C. A. Jones, F. L. Drake, and L. Lewin, *Python & XML: XML Processing with Python 1st Edition.* O'Reilly Media; 1st edition, 2002.

[65] R. Lawson, *Web Scraping with Python: Successfully scrape data from any website with the power of Python (Community Experience Distilled).* Packt Publishing, 2015.

[66] M. T. Goodrich, M. H. Goldwasser, and R. Tamassia, *Data Structures and Algorithms in Python 1st Edition.* Wiley; 1st edition, 2013.

[67] K. D. Lee and S. Hubbard, *Data Structures and Algorithms with Python (Undergraduate Topics in Computer Science).* Springer International Publishing, 2015.

[68] J. Kazil and K. Jarmul, *Data Wrangling with Python: Tips and Tools to Make Your Life Easier 1st Edition.* O'Reilly Media; 1st edition, 2016.

[69] Z. Radtka and D. Miner, *Hadoop with Python.* O'Reilly Media, Inc., 2015.

[70] M. Heydt, *Learning pandas.* Packt Publishing, 2015.

[71] D. McCreary and A. Kelly, *Making Sense of NoSQL: A guide for managers and the rest of us First Edition.* Manning; First Edition, 2013.

[72] I. Idris, *Python Data Analysis.* Packt Publishing, 2014.

[73] F. Nelli, *Python Data Analytics: Data Analysis and Science using pandas, matplotlib and the Python Programming Language 1st Edition.* Apress; 1st edition, 2015.

[74] W. McKinney, *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython 1st Edition.* O'Reilly Media; 1st edition, 2012.

[75] A. Nandi, *Spark for Python Developers.* Packt Publishing Limited, 2015.

[76] U. Gundecha, *Learning Selenium Testing Tools with Python.* Packt Publishing, 2014.

[77] H. Percival, *Test-Driven Development with Python: Obey the Testing Goat: Using Django, Selenium, and JavaScript 1st Edition.* O'Reilly Media; 1st edition, 2014.

[78] W. McGugan and H. Kinsley, *Beginning Python Games Development, Second Edition: With PyGame 2nd ed. Edition.* Apress; 2nd ed. edition, 2015.

[79] W. McGugan, *Beginning Game Development with Python and Pygame: From Novice to Professional (Beginning From Novice to Professional) 1st ed. Edition.* Apress; 1st ed. edition, 2007.

[80] wrobstory, *PythonToScala.* GitBook, 2014.

[81] T. Cox, *Raspberry Pi cookbook for Python programmers.* Birmingham, UK: Packt Pub., 2014.

[82] J. Younker, *Foundations of Agile Python Development*, 1st ed. Berkeley, CA: Apress, Jun 2008.

[83] D. Mertz, *Functional Programming in Python*, first edition ed. Shroff/O,ÄôReilly, Jun 2019.

[84] S. Raschka and R. S. Olson, *Python Machine Learning: Unlock Deeper Insights into Machine Learning with This Vital Guide to Cutting-Edge Predictive Analytics.* Birmingham, UK: Packt Publishing, 2015.

[85] B. A. Meier, *Python GUI Programming Cookbook - Second Edition.* Birmingham: Packt Publishing, 2017.

[86] G. C. Hillar, *Internet of Things with Python: Interact with the World and Rapidly Prototype IoT Applications Using Python.* Birmingham (U.K.): Packt Publishing, 2016.

[87] G. Robles, R. G. Kula, C. Ragkhitwetsagul, T. Sakulniwat, K. Matsumoto, and J. M. Gonzalez-Barahona, "Pycefr: Python competency level through code analysis," in *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*, ser. ICPC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 173,Äì177. [Online]. Available: https://doi.org/10.1145/3524610.3527878

[88] H. Hata, C. Treude, R. G. Kula, and T. Ishio, "9.6 million links in source code comments: Purpose, evolution, and decay," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2019.

[89] J. Wu, H. He, W. Xiao, K. Gao, and M. Zhou, "Demystifying software release note issues on github," *arXiv preprint arXiv:2203.15592*, 2022.

[90] P. Goodliffe, *Becoming a better programmer*, first edition ed. O'Reilly Media Inc, 2015.

[91] J. Kazil and K. Jarmul, *Data Wrangling with Python: Tips and Tools to Make Your Life Easier*, 1st ed. O'Reilly Media, 2016.

[92] (2023) Numpy. Accessed on October 13, 2023. [Online]. Available: https://numpy.org/

[93] (2023) Pymysql. Accessed on October 13, 2023. [Online]. Available: https://github.com/PyMySQL/PyMySQL

[94] (2023) Rxpy. Accessed on October 13, 2023. [Online]. Available: https://github.com/ReactiveX/RxPY

[95] (2023) Matplotlib. Accessed on October 13, 2023. [Online]. Available: https://matplotlib.org/

[96] (2023) Pydraw. Accessed on October 13, 2023. [Online]. Available: https://pypi.org/project/pydraw/

[97] (2023) wxpython. Accessed on October 13, 2023. [Online]. Available: https://wxpython.org/index.html

[98] (2023) Pygame. Accessed on October 13, 2023. [Online]. Available: https://www.pygame.org/news

[99] (2023) Flit. Accessed on October 13, 2023. [Online]. Available: https://pypi.org/project/flit/

[100] (2023) Ironpython. Accessed on October 13, 2023. [Online]. Available: https://ironpython.net/

[101] (2023) Pyboxes. Accessed on October 13, 2023. [Online]. Available: https://pypi.org/project/pyboxes/

[102] (2023) Tkinter. Accessed on October 13, 2023. [Online]. Available: https://docs.python.org/3/library/tkinter.html

[103] (2023) Scikit-learn. Accessed on October 13, 2023. [Online]. Available: https://scikit-learn.org/stable/

[104] (2023) Cython. Accessed on October 13, 2023. [Online]. Available: https://cython.org/

[105] (2023) Django. Accessed on October 13, 2023. [Online]. Available: https://www.djangoproject.com/

[106] (2023) Mapreduce. Accessed on October 13, 2023. [Online]. Available: https://www.talend.com/resources/what-is-mapreduce/

[107] (2023) Pandas. Accessed on October 13, 2023. [Online]. Available: https://pandas.pydata.org/

[108] (2023) Selenium. Accessed on October 13, 2023. [Online]. Available: https://selenium-python.readthedocs.io/

[109] (2023) Pyopengl. Accessed on October 13, 2023. [Online]. Available: https://pyopengl.sourceforge.net/

[110] (2023) Pyglet. Accessed on October 13, 2023. [Online]. Available: https://pyglet.org/

[111] T. Antao, *Bioinformatics with Python Cookbook*. Packt Publishing, 2015.

[112] E. S. Wiese, A. N. Rafferty, and A. Fox, "Linking code readability, structure, and comprehension among novices: it's complicated," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 2019, pp. 84–94.

[113] R. Hosseini, K. Akhuseyinoglu, P. Brusilovsky, L. Malmi, K. Pollari-Malmi, C. Schunn, and T. Sirkiä, "Improving engagement in program construction examples for learning python programming," *International Journal of Artificial Intelligence in Education*, vol. 30, no. 2, pp. 299–336, 2020.

[114] K. McMaster, S. Sambasivam, B. W. Rague, and S. Wolthuis, "Java vs. python coverage of introductory programming concepts: A textbook analysis," *Information Systems Education Journal*, vol. 15, pp. 4–13, 2017.

[115] T. Koulouri, S. Lauria, and R. D. Macredie, "Teaching introductory programming: A quantitative evaluation of different approaches," *ACM Transactions on Computing Education (TOCE)*, vol. 14, no. 4, pp. 1–28, 2014.

[116] P. Phan-Udom, N. Wattanakul, T. Sakulniwat, C. Ragkhitwetsagul, T. Sunetnanta, M. Choetkiertikul, and R. G. Kula, "Teddy: automatic recommendation of pythonic idiom usage for pull-based software projects," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2020, pp. 806–809.

[117] Z. Zhang, Z. Xing, X. Xia, X. Xu, and L. Zhu, "Making python code idiomatic by automatic refactoring non-idiomatic python code with pythonic idioms," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022, pp. 696–708.

[118] S. F. Mazumder, C. Latulipe, and M. A. Pérez-Quiñones, "Are variable, array and object diagrams in java textbooks explanative?" in *Proceedings*

of the 2020 ACM conference on innovation and technology in computer science education, 2020, pp. 425–431.

[119] M. Almansoori, J. Lam, E. Fang, A. G. Soosai Raj, and R. Chatterjee, "Textbook underflow: Insufficient security discussions in textbooks used for computer systems courses," in *Proceedings of the 52nd ACM technical symposium on computer science education*, 2021, pp. 1212–1218.

[120] I. Alpizar-Chacon, J. Barria-Pineda, K. Akhuseyinoglu, S. Sosnovsky, P. Brusilovsky *et al.*, "Integrating textbooks with smart interactive content for learning programming," in *CEUR Workshop Proceedings*, vol. 2895. CEUR WS, 2021, pp. 4–18.

[121] X. Huang, Q. Liu, C. Wang, H. Han, J. Ma, E. Chen, Y. Su, and S. Wang, "Constructing educational concept maps with multiple relationships from multi-source data," in *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 1108–1113.

[122] J. Wu, Y. Sun, J. Zhang, Y. Zhou, and G. Huang, "A programming language learning service by linking stack overflow with textbooks," in *2023 IEEE International Conference on Web Services (ICWS)*. IEEE, 2023, pp. 234–245.