education sciences

MDPI

Article

Educational Mobile Apps for Programming in Python: Review and Analysis

Maren Schnieder * and Sheryl Williams

School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University, Loughborough LE11 3TU, UK
* Correspondence: m.schnieder@lboro.ac.uk

Abstract: The interest in educational apps is continuously increasing due to their potential to improve the learning environment of students through the personalisation and interaction of the technology. This paper provides an overview of educational mobile apps that teach programming in Python. Existing apps were reviewed, and suggestions for future development within this field are provided within this paper. A search was performed in the Android Google Play Store. The marketplace for educational apps teaching Python was illustrated based on 78 apps. A framework to categorise the apps based on the interactivity of the user interface was applied. Key revenue streams and features were identified (e.g., interactivity, user interface, cost/adds, reviews, downloads, and country). Their effect on download frequency and user rating was evaluated. The offer of multiple dynamic features, a certificate, and a Python IDE might have a positive influence on the number of downloads or user rating. More than one-third of these apps showed static content like a book, while the remainder had dynamic features such as a Python IDE, community support, competitions, interactive tutorials, and/or quizzes. The recommendation for future app developments is proposed based on these findings.

Keywords: e-learning; distance learning; mobile apps; Python; coding; programming; educational apps

Citation: Schnieder, M.; Williams, S. Educational Mobile Apps for Programming in Python: Review and Analysis. *Educ. Sci.* **2023**, *13*, 66. https://doi.org/10.3390/educsci13010066

Academic Editors: Neil Gordon and Han Reichgelt

Received: 24 November 2022
Revised: 3 January 2023
Accepted: 4 January 2023
Published: 8 January 2023

1. Introduction

Mobile phones and mobile apps have, in the modern era, become an indispensable tool of daily human life [1]. Apps facilitate personal interaction and adapt to the users' needs [1].

Since the introduction of computers into education settings 30 years ago, they have shown true potential to improve the learning experience for students [2]. In the last decade, mobile tablets have also been introduced into educational provision [2] and are now an emergent part of an advanced education system [3].

The evolution of mobile technology has had an impact on education in a way that has changed the fundamental teaching strategies and learning by the inclusion of remote and mobile learning platforms [4]. In higher education, pedagogical technology shows favourable improvement in student learning through the use of mobile apps [5]. Unsurprisingly, educational apps are among the most common category on Google Play [1,6]. Their versatility to encourage personalised learning (i.e., no temporal or location constraint), with readily adaptable formats, allows the use of complementary media and communication (i.e., audio, video, text, and photos) [1,7]. Mobile apps also allow the student to communicate directly, which both improves the guided learning offered by instructors and the peer-to-peer support from fellow students [4]. This level of adaptability empowers learners to choose their preferred learning style and tailor the learning environment to individual needs. Apps have also been proven to help support students with special educational needs [8].

To support the research on educational apps, we mapped the marketplace of apps teaching Python and identified features and user interface types. These features may or may not increase the success of an app (i.e., number of downloads and user review score). The results are especially useful for developers or educators creating new apps. The paper is structured as follows: Following a literature review, the contribution of this paper is illustrated. An overview of the review methodology is presented afterwards. Finally, the results are presented and discussed, and conclusions are drawn.

2. Background

2.1. Educational App Assessments

Researchers have readily tested the effectiveness of educational apps in trial studies with children as the participants. These include an app that improves handwriting [9], one that teaches measurement concepts [10], and another that teaches pupils to tell the time [11]. Reviews of apps used in the higher education setting are becoming frequent [1]. However, researchers have been complaining for years that these reviews provide limited systematic knowledge [5,12]. To overcome this deficiency, researchers have taken advantage of the rapid increase in studies testing apps and have begun to conduct comparisons of multiple apps for various topics including sustainability [13], science education [14], and surgical residency [15]. These reviews can be split into two categories: (i) evaluating apps first hand or (ii) reviewing papers that have evaluated apps.

Crompton and Burke [5] evaluated mobile learning by reviewing the published literature of the time. They gave recommendations for future research such as investigating the reasons for the positive outcome that apps may have. They also recommended testing the apps in an informal setting as opposed to in a classroom, as most researchers have only focused on this scenario so far.

Zydney et al. [14] conduced a similar review with a focus on papers evaluating mobile learning in science education. They concluded that the mobile apps could be categorised into four main groups: (i) place-based data-collection tools (i.e., record the position and data entries), (ii) games/simulations, (iii) learning management systems, and (iv) productivity tools (i.e., note taking). Their recommendations included measures to assess apps based on the students' higher-level cognitive thinking skills, brain activity, and learned skills (e.g., repetition).

In contrast to this, Papadakis et al. [16] determined whether apps had been created following an appropriate design and development standard. They concluded that the reviewed apps focused on drill-and-practice-style learning, and they did not necessarily contribute to a deeper conceptual understanding.

Computation thinking [17] as well as the ability to programme [18] is seen as a perquisite for the children and citizens of tomorrow. Coding is becoming the new literacy for the next generation [19]. In fact, there is a significant increase in Science, Technology, English, and Maths (STEM) jobs globally. However, concerns are raised that not enough school graduates pursue careers in STEM to fulfil this demand [20,21]. To address this situation, reforms have been actioned to implement initiatives such as teaching computer science and increase the focus on STEM subjects in early years [20] as well as to support the acquisition of programming skills [18]. These initiatives range from young 4–5-year-old children learning basic programming concepts to university students writing complex code [18]. To accomplish this, new initiatives have been incorporated to support teachers in developing and selecting appropriate resources to teach coding to children [22]. An example of this is the project TACCLE 3—Coding European Project (Ref. 2015-1-BE02-KA201-012307), which has three primary objectives: (i) equip teachers with the required knowledge and materials to teach coding effectively; (ii) create a website that lists available resources to inspire teachers to create innovative schemes of work; and (iii) provide staff professional development training [22].

While coding may be a common component of engineering degrees, this is not the case for other graduates such as business administration [19]. As said before, coding skills

are essential for the next generation of employees [21]. It is not just university students that show interest in learning to code; it is becoming more widespread in our culture. This has led to the emergence of free online resources to learn coding [18]. These self-paced learning-to-code apps have gained popularity [18], and the number of self-proclaimed education apps has snowballed on Google Play and the App Store [16]. A few apps have been developed through years of extensive research, such as ScratchJr [17], while others may not share this integrity [17]. It therefore makes it challenging for teachers, individuals, and parents to navigate the marketplace. However, it is a critical task to select the best apps with the most effective teaching and learning material [17]. A few educational apps for children may have, in their construction, only limited educational value [16]. While educational apps that teach coding in a gamified way have become popular in recent years [22], the empirical research in the usefulness and effectiveness of these apps is limited [20].

Empowering teachers to create mobile apps may improve the availability and quality of educational apps [23], as they are acknowledged experts in development of curriculum content. However, doing this is challenging for teachers due to the lack of knowledge in app design [23]. In fact, the requirements for learning how to code is generally a barrier that is impossible to overcome for most teachers [23]. Early on, software companies have provided easily used interfaces to develop apps to overcome this constraint for teachers (e.g., visual language–block-based programming languages). An example is App Inventor [24].

There are two different types of apps aimed at supporting pupils and students to gain the necessary skills to programme. The term "coding apps" is commonly defined as "[. . .] apps that are designed to teach students the types of logical thinking, problem solving, sequencing, and planning required for coding computer programs" [25]. These apps do not necessarily teach how to write lines of code in a specific language, but their intention is to introduce pupils to concepts commonly seen in coding [25]. Apps that teach students how to write lines of code are, for example, referred to as self-paced "learn to code" (SPL2C) apps [18]. While papers focusing on coding apps are included in the literature review, the systematic review of apps in this paper focusses solely on apps that teach users to write lines of code in a specific language (i.e., Python).

Research on evaluating coding apps has become even more popular in recent years:

Pila et al. [20] observed a one-week coding course for 28 young children using two different apps (i.e., Daisy the Dinosaur and Kodable). They evaluated, for example, the device and app familiarity, appeal to the students, and student comprehension. They concluded that the children learned foundational coding skills through the two different apps. The result did not give any obvious gender difference. The children's command knowledge was incremental with a growing appeal of the app Daisy the Dinosaur. However, this was not the case for the app Kodable. It is generally accepted that the more a child likes a program, the more frequently they will interact with it. This repetition improved their final learning outcome significantly. However, if a child pays too much attention to the narrative of characters within the game, this decreases the focus of the child on the educational content.

Sheehan et al. [26] conducted a similar evaluation of the app PBS KIDS ScratchJr. They evaluated the interaction between parents and children while playing with the app, the children's ability to select the correct buttons to fulfil a coding task, and the children's ability to determine what a sequence of buttons would do.

Stamatios [17] conducted a systematic literature review of 18 studies to determine whether the app PBS KIDS ScratchJr can teach "computational thinking skills". They concluded that the app (i) can be integrated into an interdisciplinary curriculum, (ii) is freely available, (iii) is age-group-suitable, and (iv) supports students' numerical and literacy skills. The disadvantages included the requirement for a larger screen to effectively use the app as well as limitations of the scope of the activities.

Francisco et al. [22] created a database with 37 apps. Their database includes mostly coding apps but also some SPL2C apps. They compared the apps based on (i) their licence type (more than half of the apps were licenced as "creative commons' or "freeware"),

(ii) categories (e.g. algorithms, using logic, controlling things, creating and debugging), (iii) available languages, and (iv) complementary classification (e.g. app for teaching coding, robotic, programming language, info site, and training course).

The research on SPL2C apps is comparably limited:

A few researchers have published papers about the development of mobile apps teaching Python but without objectively reviewing it. An example is Micah and Bibu [27], who developed a mobile app that teaches Python using the PhoneGap development tool. More advanced tools to teach programming in Python are a "chatbot" that teaches Python [28].

Fabic et al. [29] evaluated their own prototype app using 76 participants who used the app for a 1.5–2 h session. Using a Wilcoxon signed-ranks test, they determined statistically significant learning gains of individual participants between the pre and post test.

Arnedo-Moreno et al. [18] focused on the most effective game design elements as well as comparing these with the wider educational context. They concluded that while leader boards are popular game design elements, they are not prolific in the reviewed apps. On the other hand, progress and feedback is far more common in apps than in general education. They identified the following game elements: progress/feedback, points, badges, prize/rewards, challenges, levels, leader boards, and avatars.

Schnieder et. Al. [30] reviewed the question styles used in apps teaching Python. They also evaluated whether having a more varied question styles influences the number of downloads and user ratings.

2.2. Contribution

The previous literature review has highlighted a few key issues: (i) computing skills are a crucial requirement for the next generation of employees, (iii) insufficient students graduate with the required coding skills to meet the expected demand, (iii) the number of self-proclaimed apps that teach programming or computational thinking is rapidly expanding, (iv) some of the apps have limited educational benefit, and (v) it is difficult for teachers and students to navigate the app market to identify effective apps.

To solve these issues, we adopted a methodology proposed by Tabi et al. [31] to map the marketplace for apps. While Tabi et al. [31] focussed on apps for medication management, this study focusses on apps teaching Python. We focussed on Python because it is considered to be an simple coding language and suitable for first-time programming [27,32,33].

The following research questions are therefore addressed:

RQ1: How is the marketplace for "educational apps" teaching Python, characterised, and what are the key revenue streams and market players?

RQ2: What are the key features of these apps?

RQ3: Which user interface types are commonly used (applying a framework to categorise the apps based on the interactivity of the user interface)?

RQ4: What features of the apps and user interface types result in an increased number of downloads and user rating?

RQ5: Which feature should be selected by developers to create economically viable educational apps that teach programming?

This research clearly differs from the study by Arnedo-Moreno et al. [18], who focussed on game design elements used in twelve selected apps. While they performed a search for apps in the Google Play store, as in this paper, they excluded all apps with a user rating of less than 4.0, with fewer than 1000 ratings, and with fewer than 10,000 downloads. This reduced the number of originally identified apps from 1602 (no duplicates) to 37 apps. As part of a manual evaluation process, this was then further reduced to 9 apps and 3 previously excluded apps being reinstated. It is therefore a fair conclusion that this process is clearly not representative of the app market, as only the **best**-scored apps are included, and no "failures" or less-appealing apps are included.

The information contained in this paper shows a fresh overview of the marketplace of apps that teach Python. While published research has shed light on the educational effectiveness of apps, this study meets the need for developers to identify those features

and common revenue streams as well as their influence on the success of the apps (i.e., downloads user ratings). Obviously, educational effectiveness is a key aspect for educational apps; however, developing commercially viable apps is, for most developers, a prerequisite. The paper also highlights the shear complexity of the market forces users are faced with when selecting an app to learn Python.

3. Methods

3.1. Overview

A common approach for the review of apps was used in this study [34–36]. The search was conducted using apps available on one main mobile app store: Android Google Play Store (from here on referred to as Google Play). Android is the most common mobile operating system, holding a market share of over 70% [37]. The market share of new devices is even higher for Android [38]. Google Play was searched in May 2022, following the search methodology in Tabi et al. [31]. The search engine start page was used to identify relevant apps (see 'Section 3.3. Eliminating Personalisation during the Search' for more details). The query consisted of a search term and the location of the search, for example, Learn Python site://play.google.com/store/apps/. (Accessed on 3 May 2022). The search terms used were Python tutorial, Python course, learn Python, and Python training. As in [31], a cut-off value of 80 apps was used given that the number of apps considered "irrelevant" was increasing beyond that point.

The search terms were selected through a discussion amongst the authors. A preliminary search was conducted to verify the effectiveness of the selected search terms.

3.2. Selection of Eligible Solutions

The following exclusion criteria were applied: (i) the app was not available in English, (ii) the app did not provide learning materials to teach Python, and (iii) the app did not focus on teaching coding but was rather a platform to access a variety of training courses and not necessarily coding-related.

Subsequently, a final list of eligible apps was created, and their website on Google Play for each of these apps was downloaded.

3.3. Eliminating Personalisation during the Search

Searches on Google Play are personalised through, for example, the app store search engine algorithms and cookies. To decrease the personalisation, we applied the same strategies as in Tabi et al. [31]. By using StartPage, we were able to access results from the Google search engine while having an increased anonymity through Secure Sockets Layer encryption and reduced personal data collection. We also used the incognito mode of the browser [39,40].

3.4. Data Extraction and Solution Assessment

Prior to data extraction, an initial framework based on previous literature [31] and discussions amongst the authors was been constructed, which summarises the features of the apps. This framework was updated during the review of the apps. The final design is presented in Table 1.

Thereafter, the apps were moderated using a similar method as Tabi et al. [31]: The website of each app was downloaded, including the description and screenshots of the app. An exception was the country of the developer, as this was not always listed in the app description of all apps. In these cases, the information about the developer's location was searched for through a review of the privacy policy, an online search using the company's name, or the website of the developer. At first, all relevant information about the app was extracted such as rating, number of downloads, etc. The app's features and user interface were assessed as follows: During app assessment, other features were identified that were not part of the initial framework. These were manually entered on an enriched database. Simultaneously, these features were flagged, especially if they were considered novel or

innovative. On completion of the review, the team of authors agreed on a final list, which is included in the study.

Table 1. App characteristics.

App Characteristics	Description
	Features
Quizzes	Quizzes and interactive tutorials that allow users to test their knowledge and skills.
Interview questions	A question bank to prepare for job interviews.
Interactive elements	Some apps include the option to interact with other users like on social media platforms such as Instagram.
Certificate	A certificate is supplied to the user after completion of (parts of) the course.
Python IDE	A compiler that can be used to run code: The code can either be sent to an online compiler or can be run on the device itself.
	Cost and advertisements
Cost	The cost to download the app.
In-app purchases	Some apps could be downloaded for free. However, the app includes options to purchase additional content (referred to as in-app purchases).
Ads	Encoded to determine whether the app generates revenue through advertisements.
	Others
User rating	The rating of the app by users in stars (1 stars to 5 stars).
Downloads	Number of downloads.
Country	Country of the app developer: Sometimes this is not published. Then, it had to be retrieved by Googling the company of the developer or by looking through the privacy policy for information.
	User interface
Static only	Apps that display the same content for everyone: They provide only minimal interactivity, such as the option to change the letter size or background colour, to navigate through the app, and to set bookmarks.
Single dynamic feature	Apps that have one interactive feature: these apps are usually apps providing a Python IDE or a quiz or interactive elements.
Multiple dynamic features	Apps that offer multiple dynamic features solutions such as quizzes, guided coding exercises, competitions against other users, and community support.

One of the authors reviewed and categorised all apps, while 20% (i.e., 15 apps) of these were randomly selected and were categorised by the second author as well. This was performed to ensure that the categorisation method is repeatable. Whether an app offers a certificate, interview questions, quizzes, or a Python IDE had to be determined based on the app description. Consequently, there was a difference between the reviewers as to whether an app has encapsulated these features. The Cohen kappa score of 0.81 interrater reliability was determined as $\kappa = 0.69$. This indicates a substantial agreement according to Landis and Koch statistics guidelines (0.61 to 0.80) [41]. If there was a difference of opinion of the authors reviewing the apps, then a detailed discussion followed until a consensus of opinion was formed.

3.5. Data Analysis

Descriptive statistics were calculated for each attribute once all the apps were reviewed. The statistics and visualisations were computed in Python.

4. Results

4.1. Search and Categorizations

A total of 320 URLs were identified through the search with the four key words. The search term "learn Python" clearly performed best, while the other search terms frequently included apps about different programming languages or non-coding-related courses. The 320 URLs included 189 unique URLs. However, in some cases, the URLs ended on "&hl=en_US&gl=US" and other times on "&hl=en_GB&gl=US" and referred to the same app, only differentiating the English spelling as British as opposed to American. Aft4er these types of duplicates were removed, 123 apps were identified. After excluding all irrelevant apps, 78 apps were included in the final assessment. Figure 1 demonstrates the stages in the review process.

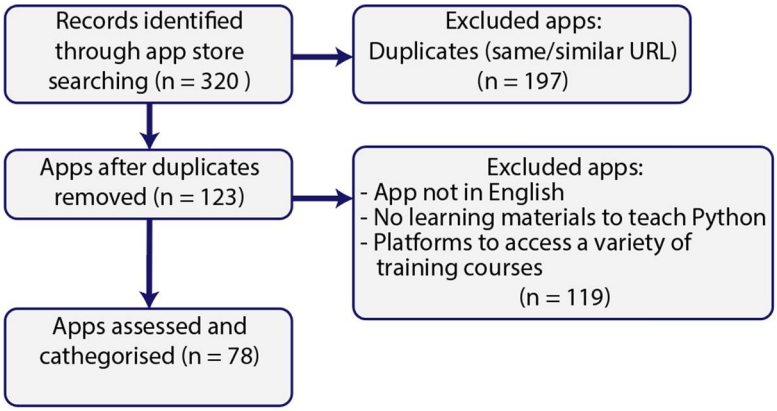

Figure 1. App review process.

4.2. Features

A total of 17.9% (14/78) of the apps award a certificate to the student after completion of the course. These certificates can in most cases be shared on social media or with employees. Some of the apps also offer verifiable certificates (i.e., employees can verify whether the certificate is true and not fake). Certificates are, for example, offered by "Python Advanced" (https://play.google.com/store/apps/details?id=com.python.dev2020.apk20009&hl=en_US&gl=US, accessed on 17 December 2022). Verifiable certificates are, for example, offered by "Learn Python: Ultimate Guide" (https://play.google.com/store/apps/details?id=python.programming.coding.python3.development&hl=en_GB&gl=US, accessed on 17 December 2022). The certificate by "Sololearn: Learn to Code" (https://play.google.com/store/apps/details?id=com.sololearn&hl=en_GB&gl=US, accessed on 17 December 2022) can be posted on LinkedIn for future employers to see.

Another 29.5% (23/78) apps have a question bank of interview questions and answers. A few of these question banks are rather large, with more than 500 questions split across multiple categories. An example of this is the app "Learn Python: Python Tutorial" (https://play.google.com/store/apps/details?id=mcqcom.dhanesha.pythonlanpro&hl=en_US&gl=US, accessed on 25 August 2022), developed by Jayesh Dhanesha.

Only 6.4% (5/78) of the apps include interactive elements. This means that part of the content is generated by users. Common examples of this are social media apps such as Instagram, Twitter, and TikTok. The five reviewed apps have chats or forums users can use to ask questions for community support. A few apps allow users to create lessons for fellow users. One example is "Sololearn: Learn to Code" (https://play.google.com/store/apps/details?id=com.sololearn&hl=en_GB&gl=US, accessed online: 25 August 2022), developed by Sololearn. Like on other social media platforms, users can become "community influencers" with many followers. This app also allows users to communicate, providing 24/7 support to other users.

Overall, 33.3% (26/78) of the apps allow users to test their knowledge with quizzes, MCQs, and similar activities. A detailed overview of the quizzes commonly used in apps

to assess coding skills can be found in Schnieder et al. [30]. These questions can be split into theory questions and coding questions. Theory questions are usually either missing word games (i.e., user needs to type/select a word to complete a sentence) or MCQs where the user selects one or multiple correct answers. Coding questions are, for example, MCQs where the user identifies the correct line of code from a selection of examples so that a specified output is displayed. In the same way, the user could be given a code snippet and be asked to select the correct output. Sometimes, these code snippets have errors. For simple tasks, the user either selects whether the code is working or throws an error. In more complicated questions, the user is asked to select the error message, to select the reason for the error, or to select the line in the code with the mistake. Asking students to identify errors is one of the question types used in the "Python for Engineers Concept Inventory" [42]. These are 250 MCQs that assess the student's coding ability. Similar to the missing word in a sentence, users can also be asked to complete code snippets by selecting or typing the missing part. The question type where the user matches pairs or orders answers is rarely used. A number of the apps have built-in compilers asking the students to type a short code snippet. Given that coding is open-ended [42] and a rather practical skill [43], it is difficult to automatically assess the users' code. One option used by app developers is to simply ask the students to determine whether the code is correct themselves. Other apps offer peer-to-peer support allowing students to evaluate each other's code. Researchers also reported positive experiences with peer code review and stated that it improved the students learning outcomes [43]. Other apps test whether the output is as it should be. Some apps such as "DataCamp: Learn Data Science" (https://play.google.com/store/apps/details?id=com.datacamp&hl=en_GB&gl=US, accessed on 25 August 2022) go even beyond simply offering quizzes. Instead, they offer interactive tutorials where the user first discovers theoretical content and is then guided in steps to writing longer codes with instant and personalised feedback or suggestions on how to fix the code. Quite a few apps offer users to compete against each other in challenges. For example, "Dcoder, Compiler IDE: Code & P" (https://play.google.com/store/apps/details?id=com.paprbit.dcoder&hl=en_US&gl=US, accessed on 12 August 2022) ranks its users on a leader-board.

Slightly less than half of the apps (44.9 %, 35/78) provide a Python IDE, allowing the user to write and run code. Examples are "Learn Python: Ultimate Guide" (https://play.google.com/store/apps/details?id=python.programming.coding.python3.development&hl=en_GB&gl=US, accessed on 17 December 2022), "Learn Python" (https://play.google.com/store/apps/details?id=python.learnpython.learn.pythonx.coding.programming.python3.tutorials.codingx&hl=en&gl=US, accessed on 17 December 2022), and "Learn Python Coding, PythonPad" (https://play.google.com/store/apps/details?id=com.codepoint.learnpython3&hl=en&gl=US, accessed on 17 December 2022). Some of the apps send the code to online compliers, while other apps run the code on the phone itself. Note: A few apps allow users to "run" pre-written code snippets. This is not considered to be a Python IDE, as the user cannot change the code. Some apps go even beyond offering a simple Python IDE. "Dcoder, Compiler IDE: Code & P" (https://play.google.com/store/apps/details?id=com.paprbit.dcoder&hl=en_US&gl=US, accessed on 12 August 2022) offers autocomplete and the option to integrate with Git (Gtihub, bitbucker) to make coding easier.

The user interface categorisation suggested in Tabi et al. [31] was applied in this study. More than one-third of all apps (39.7%, 31/78) only have a static user interface. They provide text or videos but do not allow the user to communicate with other users, test their knowledge with quizzes, or provide a Python IDE. A few apps with a static user interface allow users to change the front size, change the background colour, or set bookmarks. A similar number of apps (37.2%, 29/78) has one dynamic feature such as quizzes or a Python IDE or offers community support. Please note that all the reviewed apps in this category either offer quizzes or a Python IDE. Further, 23.1% of the apps (18/78) offer two or three dynamic features. Most of these apps offer both a Python IDE and quizzes. Rarely, apps also allow users to create lessons for other users, propose or participate in competitions, and

compare their performance with others on leader boards. Examples of a static user interface are "Learn Python—Beginning to Advanced" (https://play.google.com/store/apps/details?id=aga.python&hl=en_US&gl=US, accessed on 17 December 2022), which includes free Python eBooks. Apps with one dynamic feature are, for example, "Python Editor" (https://play.google.com/store/apps/details?id=python.editor&hl=en_US&gl=US, accessed on 17 December 2022) and "Simple Python" (https://play.google.com/store/apps/details?id=com.yernar.python, accessed on 17 December 2022), which are apps offering a Python IDE. "Learn python: python tutorial" (https://play.google.com/store/apps/details?id=com.protectsoft.pythontutorial&hl=en_US&gl=US, accessed on 17 December 2022), "Py Learning Companion" (https://play.google.com/store/apps/details?id=com.tmdstudios.python&hl=en&gl=US, accessed on 17 December 2022), and "Pocket Python" (https://play.google.com/store/apps/details?id=com.edu.app.code&hl=en_US&gl=US, accessed on 17 December 2022) offer quizzes. The following are examples of apps with multiple dynamic features: "learn python programming: pytutor" (https://play.google.com/store/apps/details?id=com.olabode.wilson.pytutor&hl=en_US&gl=US, accessed on 17 December 2022) and "Python Code Play" (https://play.google.com/store/apps/details?id=python.code.play&hl=en_US&gl=US, accessed on 17 December 2022). "Meraki—Learn Python & Typing" (https://play.google.com/store/apps/details?id=org.merakilearn&hl=en_IN&gl=US, accessed on 17 December 2022) even offers live classes run by volunteers.

4.3. Revenue Streams: Ads and Cost

The cost of the app is important, as it could influence the quality of the app [44]. Moreover, the continued exposure to ads in free apps may be counterproductive. In spite of the negative aspects, in-app advertisement remains a key revenue stream [45]. The majority of apps are free to download (89.7%, 70/78). None of the eight apps that charge a fee to download them have in-app purchases or ads. The cost to download these apps ranges from USD 0.99 to USD 6.99 (median: USD 2.50). Of those apps that are free to download, only 11 apps include neither in-app purchases nor ads. Only six free apps offer in-app purchases but do not have ads, while eleven of the free apps offer in-app purchases and have ads. Forty-two of the free apps include ads but do not offer in-app purchases. Overall, ads are the commonly used form of revenue generation for these apps, followed by in-app purchases (Figure 2). Note that the costs reported in this paper are the costs stated on the website of the app in Google Play Store. It is theoretically possible that some in-app purchases (e.g., subscription) give access to additional content outside of the apps (e.g., a desktop version of the app). However, this has been disregarded in this study.

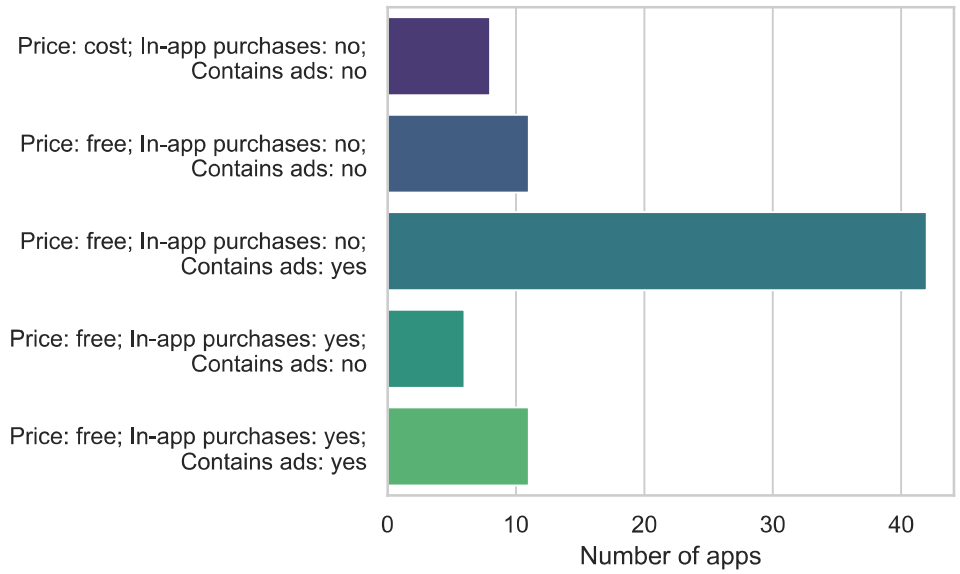

Figure 2. Cost, in-app purchases, and ads.

Most apps have a large range of in-app purchases prices. Some apps offer all items for USD 0.99. Other apps have items on sale for between USD 9.99 and USD 249.99. The maximum cost for in-app purchases ranges from USD 8.99 to USD 400 (median: USD 149.99), while the minimum cost can be as low as USD 0.99 up to USD 9.99 (median: USD 1.99).

4.4. User Rating

Only 52.5% (41/78) of the apps have a user rating score. The user rating score ranges from one to five with an accuracy of 0.1. Five is the highest score. Out of the rated apps, most apps (87.8%, 36/41) are rated at 3.5 or more. The average score is 4.1 (Figure 3).

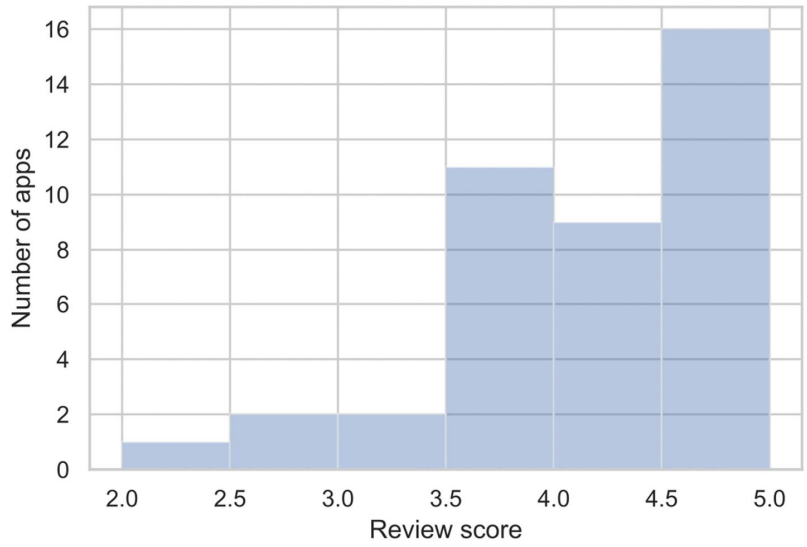

Figure 3. Review score (41 apps).

4.5. Downloads

The number of downloads is reported in the categories shown in Figure 4. For example, category 5 means that the app has been downloaded more than five times, category 50 means that the app has been downloaded more than fifty times, etc. Most of the apps are in the category 10,000+ downloads. Only 14.1% (11/78) of the apps have been downloaded less than 1000 times. Further, 39.7% (31/78) of the apps have been downloaded less than 10,000 times. Eight apps (10.3%) have been downloaded more than 1 million times.

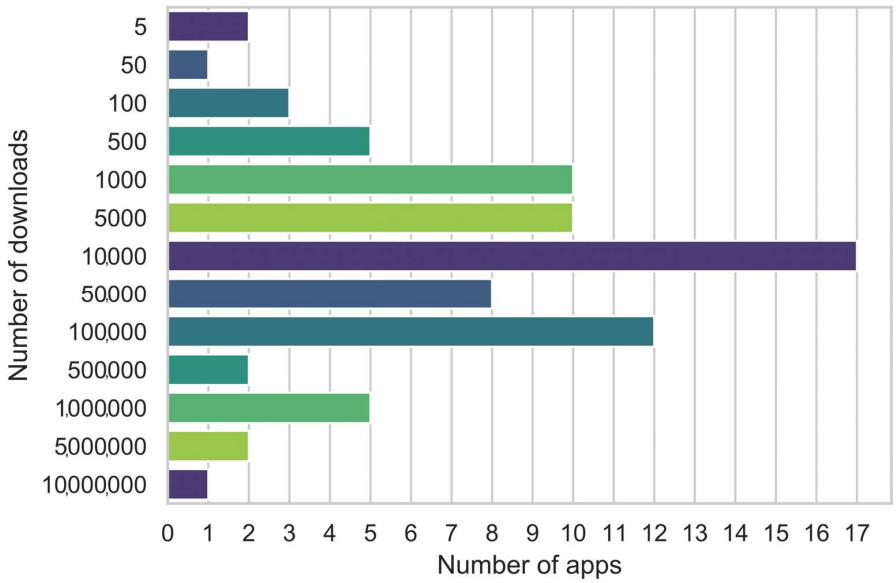

Figure 4. Number of downloads.

4.6. Country and Developer

It was possible to identify the country of the developer for around half of the apps (60.3%, 47/78). Most apps (34.0%, 16/47) were developed in India (Figure 5). The second most common country was the USA with eight developed apps (17.0%). Pakistan and Bangladesh are in the third and fourth place, respectively, with 8.5 % (4/47) and 6.4 % (3/47).

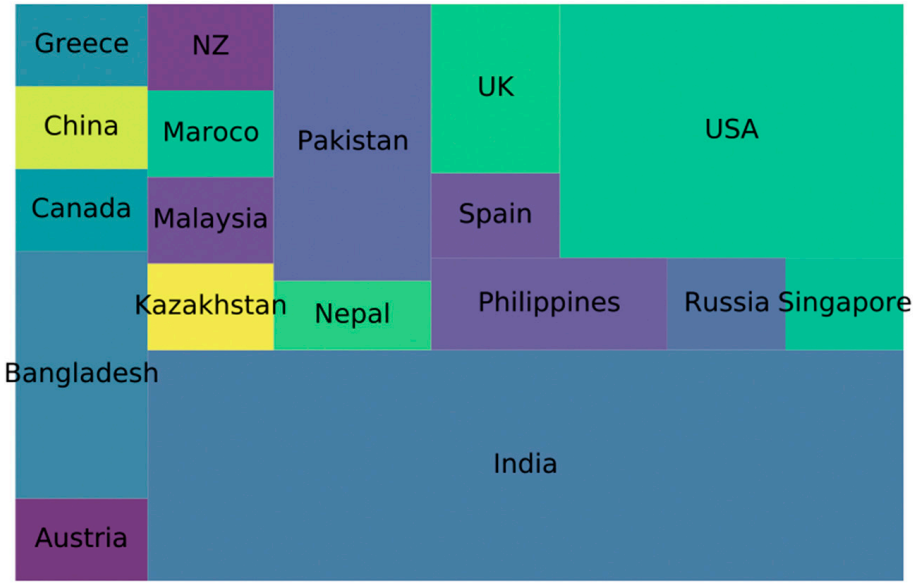

Figure 5. Country of the developer (47 apps).

4.7. Market Overload—Difficult for Users to Navigate

We found 78 unique apps on Google Play. To find the right one, the user must compare many alternative solutions. When someone wishes to use an app to learn Python, they may expect evidence-based pedagogical technologies to optimise their learning experience. This is pivotal to the common standard for apps designed to be user-friendly and promote user engagement [33,34]. Users want to gain skills as easily and with as much motivation as possible.

It is also noted that most of the apps are from rather unknown developers, with some not even mentioning their website or address of the company. Almost no app stated that they are affiliated with a higher education institute (i.e., university). Hence, most of the apps were developed by software companies or developers. Virtually all apps do not state an involvement of companies or institutes experienced in teaching. However, this can easily be explained by the resistance to change. If a university teaches coding a specific way (i.e., "face-to-face" lectures and tutorial), there might be some resistance to change the way they teach (i.e., develop apps). Teachers' resistance to change, information and communications technology (ICT) anxiety, and ITC self-efficacy are listed amongst the main barriers for mobile learning [46]. In the last decade, social, cultural, and organisational factors were commonly cited as reasons why universities are hesitant towards mobile learning and apps [47].

On the other hand, a few of the developers have risen to relatively well-known providers of coding educational online platforms or apps, such as the company Data Camp. They have made a name for themselves. The search for the app developers' credentials and country as performed in this study was rather time consuming, and it cannot be expected that users imitate the same routine [31].

4.8. Interface vs. Review Score

More than one-third of the apps only present texts in formats similar to a book or PDF. This might be a very useful thing to look up specific functions. However, it might not be beneficial for a beginner to learn how to code by reading "a book". Interactive tutorials

that teach users how to code step by step are more beneficial. However, only 60.3% of the apps have one or more dynamic features. As shown in Figure 6, users also rate apps with multiple dynamic features better than apps with a single dynamic feature or static apps. Based on a Kruskal–Wallis test, it can be concluded that the distribution of reviews for apps with multiple dynamic features differs significantly from apps with a single dynamic feature (p = 0.013). While not a statistically significant difference, the mean review score is higher for apps with static interfaces than for apps with a single dynamic feature.

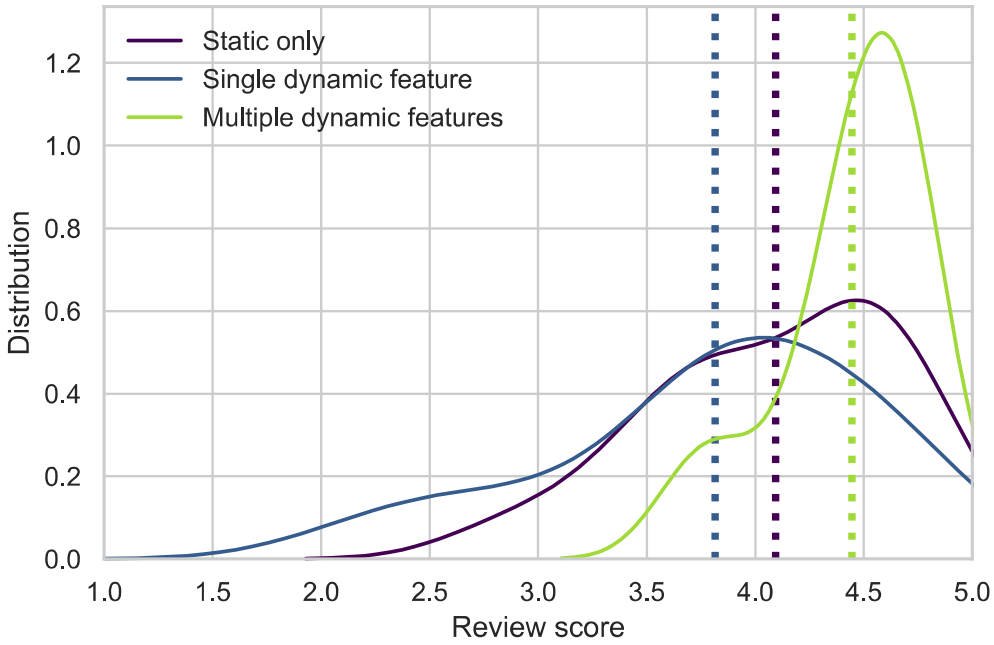

Figure 6. User rating depending on the user interface category.

4.9. In-App Purchases and Ads vs. Review Score

Based on the low cost of the reviewed apps, the willingness to pay seems to be relatively low for these apps. In the published literature, the willingness to pay for an educational apps has been reported to be as low as 13 % (college students, app teaching English [48]). People are willing to pay hundreds or thousands to learn coding as part of a university degree or to take part in accredited programming courses. However, most of the apps reviewed in this paper are free or can be purchased for a few dollars (67.9%, 53/78 are free to download/ no in-app purchases). Based on the number of downloads of these apps, it clearly shows that the focus is more on teaching as many people as possible for a small fee instead of only educating a selection of people for a high price. Hence, these apps could be used to make education more accessible for everyone.

Figure 7 highlights the differences in the user rating depending on the cost of the app (i.e., free, in-app purchases, cost to download) and whether they have ads. The five categories are shown in the figure. Apps that are "free, offer in-app purchases, and don't contain ads" score highest on average (4.60), closely followed by apps that must be purchased (4.48) and apps without ads or in-app purchases (4.30). Apps that have both in-app purchases and ads score the second lowest (4.02), followed by free apps with ads and no in-app purchases, which score the lowest (3.88). A statistical significance between the reviews for each of these groups of apps could only be determined for "Price: free; In-app purchases: no; Contains ads: no" and "Price: free; In-app purchases: yes; Contains ads: yes" (p= 0.034).

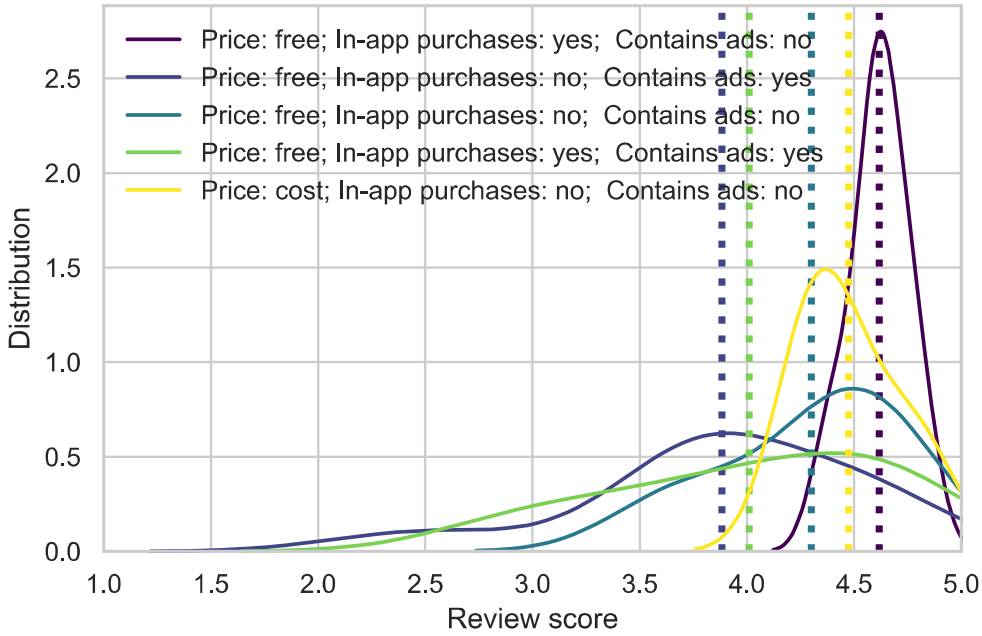

Figure 7. User rating depending on cost and ads.

4.10. Country vs. Review Score

While India is by far the home of the greatest amount of developers of apps when all apps are compared, when only the apps with a user review score of more than 4.0 are compared, USA is in the first place (25%, 5/20), followed by Pakistan (15%, 3/20). All other countries have either one or two apps (Figure 8).

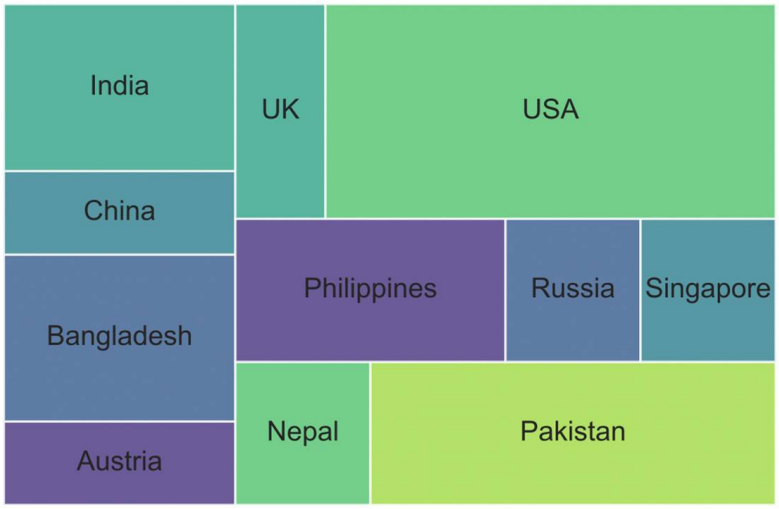

Figure 8. Country of the developer for all apps with a user review of 4.0 or higher (20 apps).

4.11. Others

There is no correlation between the number of installs and the reviews score (Pearson correlation, r = 0.23). This could be explained by new apps that could have a high review score but still have a low number of downloads. There is a positive correlation between purchase costs and reviews (Pearson correlation, r = 0.797).

Based on a Kruskal–Wallis test, there is also no statistically significant difference between the review score of apps with and without interactive elements (p = 0.670), apps with or without interview questions (p= 0.448), apps with or without a Python IDE (p = 0.318), and apps with or without a quiz (p = 0.059). The results can be expected given that some Python IDEs have a rather limited functionality and therefore might not be a

user favourite. Interview questions may not be an important factor for improving the user satisfaction. While there is no statistical significance between the review score of apps with or without quizzes, the *p*-value is very low and close to being significant. The review score of apps with a quiz is higher (4.30) than that of apps without quizzes (4.02).

There is a statistically significant difference between the review scores of apps that do and those that do not offer a certificate ($p = 0.0006$). The mean review score of the apps that offer a certificate is with 4.62 higher than of the apps that do not offer a certificate 3.98. Obviously, the results of the Kruskal–Wallis test should not be seen as a guarantee that apps with a certificate will have a higher review score. It may simply be a corelation but not a causation. Nevertheless, the results can be seen as an indication of the potential benefits (i.e., higher review score) of offering certificates.

For easier comparisons, the results are illustrated in Table 2.

Table 2. Statistically difference (Kruskal–Wallis test) between the review score/number of downloads of apps with or without the following.

	Apps with or without . . .	***p***	**Stat**
Review score	. . . interactive elements	0.670	0.181
Review score	. . . interview questions	0.448	0.577
Review score	. . . a Python IDE	0.318	0.996
Review score	. . . a quiz	0.059	3.578
Review score	. . . a certificate	0.0006 *	11.674
Number of downloads	. . . interactive elements	0.303	1.061
Number of downloads	. . . interview questions	0.500	0.456
Number of downloads	. . . a Python IDE	0.003 *	8.689
Number of downloads	. . . a quiz	0.314	1.013
Number of downloads	. . . a certificate	0.068	3.342

* Statistically significant.

Based on a Kruskal–Wallis test, there is also no statistically significant difference between the number of downloads of apps with and without interactive elements ($p = 0.303$), apps with or without interview questions ($p= 0.500$), apps with or without a certificate ($p = 0.068$), and apps with or without a quiz ($p = 0.314$). There is a statistically significant difference between the number of downloads of apps that offer and those that do not offer a Python IDE ($p = 0.003$). The apps with a python IDE have been downloaded on average 769,406 times, while the apps without a Python IDE have been downloaded on average 21,015 times.

This indicates that users might prefer to download an app with a Python IDE; however, the review score is not statistically higher for apps with a Python IDE. While users seem to rate apps that offer a certificate as better, the number of downloads is not statistically higher for those apps. This indicates that users might be encouraged to download an app based on different criteria than those they use to rate the app.

5. Discussion

5.1. Principal Findings

The review highlights the ample availability of apps supporting users to learn Python. Emphasis is placed on the many functions the current apps offer to inform and allow the development of mobile solutions. Of the reviewed apps, 17.9% award certificates for the completion of the course, 29.5 % offer interview questions, 6.4% have interactive elements, 33.3% have quizzes and MCQs, and 44.9 % have a Python IDE.

While apps with interactive elements may not have a statistically higher reviews score or the number of downloads, they should still be considered by app developers. The

number of downloads is around 10 times higher for apps with these elements then for those without (230,299, vs. 2,204,200).

The *p*-value of 0.059 does not quite warrant a statistically significant difference between the review score of apps with or without quizzes. It gives an initial indication that users may prefer apps where they can test their newly acquired knowledge. The average review score is 4.30 for apps with quizzes compared to 4.02 for apps without.

While there is a statistically significant difference between the review score of apps that do and do not offer certificates, it may not be a causation. However, the result should still highlight to developers that users seem to value the ability to prove their progress on the course. The review score is on average with certificates 4.62 and without 3.98. The number of downloads of apps with certificates is not significantly higher compared to apps without. Users may rate apps as higher when they offer a certificate, but it does not seem to be an incentive to download an app.

Another statistically significant difference between the number of downloads could be found for apps with or without Python IDE. While not necessarily a causation, it appears that users are more inclined to download apps with a Python IDE. However, there is no statistically significant difference between the review score of apps with or without a Python IDE.

Apps with multiple dynamic features have a significantly higher review score compared to apps with a single dynamic feature (p = 0.013). Developers should therefore endeavour to include multiple interactive features in their apps. The average review score for apps with one dynamic feature is 3.8 (multiple dynamic features: 4.4).

Ads, in-app purchases, and purchase price do not seem to have a huge impact on the review score. A statistical significance could only be found between the following two groups of apps: "Price: free; In-app purchases: no; Contains ads: no" (average review score: 4.3) and "Price: free; In-app purchases: yes; Contains ads: yes" (average review score: 4.0) (p= 0.034).

5.2. Initiatives for User Navigation

While there are several initiatives in the market of apps for medication management [31] to help users navigate through the market, there is a limited initiative to do the same for the market of education apps for Python coding (apart from online blogs and reviews). Some websites compare education apps; however, these websites compare the apps based on different characteristics, meaning that an app might score high on one website and low on another [49]. In the past, there have been some attempts at creating a standardised comparison and evaluation metric [49]. However, these are not widely accepted as a common standard.

Prospective students have plenty of rankings at their disposal to compare universities for their education. Schools are assessed by independent moderators to ensure the qualification is of an appropriate standard (i.e., Ofcom, Pearson, AQA, OCR, etc.). Given the benefits educational apps could have on students learning, it would be beneficial to either integrate these apps into accredited courses or create an accreditation of these apps. These initiatives could be an opportunity to create a learning-as-a-service environment offering lifelong learning.

It is difficult for the user to identify whether the app covers all the relevant content for them. Some apps simply show a long list of topics in their description, which might be rather difficult to navigate for a coding beginner. An experienced user obviously knows what they mean; however, they will not need the app to study these topics.

5.3. Limitations

As in Tabi et al. [31], the apps were not downloaded or fully tested. Instead, the review of the apps was based on the description, screenshot on the website of the app, and advertisement videos. It may be that a proportion of apps have functionalities that are not listed in the description, and this may have some bearing on the results of the study.

This method mimics the way users select an app. In doing so, it spans the divide between reality and research. Developers are aware of the way users select apps and therefore are incentivised to list as many features as possible in the descriptions. The study was focused on classifying apps and not on reviewing individual apps in detail through a qualitative evaluation. Hence, there was no need to download the apps. In addition, a study reviewing the way apps assess people's coding abilities and progress throughout the course concluded that based on a review of ten apps, all but one of the apps that state in the description that they include a quiz actually have a quiz in the app [30]. For one of these apps, it was not possible to determine whether the app had a quiz, as these seem to be hidden behind a paywall (i.e., in-app purchases). The review was limited to apps in the English language and might therefore not be representative of the whole market of educational apps to teach Python. Of course, the comparison in this discussion should not be interpreted literally or seen as a causation. For example, having a developer from the USA does not guarantee a high review score. Instead, it should be seen as an illustration to indicate user preferences, for example, which user interface or revenue stream is preferred by the user.

6. Conclusions, Limitations, and Future Research

Educational mobile apps have the potential to empower learners to adapt their learning environment and style to suit their needs and access personalised support. However, finding the best app is challenging given that the market offers many solutions, which makes it difficult for the user to find the best solution. More than one-third of the apps have primarily static features, which means that they do not offer the interactive and engaging learning environment a learner might hope for (RQ3). To find a trustworthy app, it is important for the user to be able to rely on official accreditations of either the developer or the app itself. The first option is rather limited in the description of the app, and the latter is non-existent. This study highlights a lack of reporting on the affiliations of the developers in app store descriptions as well as official accreditations and standardisation (RQ1).

This study identified certain beneficial features in apps, namely sharable certificates, interview questions, Python IDE, options for users to interact (i.e., community support or competitions), as well as quizzes and interactive tutorials (RQ2). It is important to start working on the next generation of educational apps that will improve user engagement (i.e., more interactivity, community support/competitions) and the ability of users to adapt their educational journey to their own needs and preferences. Based on statistical tests, it was determined that the review score of apps with multiple dynamic features is significantly higher compared to apps with a single dynamic feature. When comparing the revenue-generation streams, a statistically significant difference between the reviews scores could only be determined for apps with "Price: free; In-app purchases: no; Contains ads: no" and "Price: free; In-app purchases: yes; Contains ads: yes". There is also a statistically significant difference between the review scores of apps that do and those that do not offer a certificate. Apps that include a Python IDE have been downloaded with significantly more frequency than those that do not (RQ4 and RQ5).

The limitation of this study is that it only focusses on Google Play store apps. Besides investigating apps published in the Apple Store, other webpages that teach programming should be evaluated, too. An example of this is DataCamp (https://www.datacamp.com, accessed on 29 October 2022). An additional review could evaluate the apps and webtools available for other programming languages and education subjects.

Author Contributions: Conceptualization, M.S.; methodology, M.S.; software, M.S.; validation, M.S. and S.W.; formal analysis, M.S. and S.W.; investigation, M.S.; resources, M.S.; data curation, M.S.; writing—original draft preparation, M.S.; writing—review and editing, M.S. and S.W.; visualization, M.S.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Singh, Y.; Suri, P.K. An empirical analysis of mobile learning app usage experience. *Technol. Soc.* **2022**, *68*, 101929. [CrossRef]
2. Falloon, G. Young students using iPads: App design and content influences on their learning pathways. *Comput. Educ.* **2013**, *68*, 505–521. [CrossRef]
3. Mouza, C.; Barrett-Greenly, T. Bridging the app gap: An examination of a professional development initiative on mobile learning in urban schools. *Comput. Educ.* **2015**, *88*, 1–14. [CrossRef]
4. Sarrab, M.; Al-Shihi, H.; Al-Khanjari, Z.; Bourdoucen, H. Development of mobile learning application based on consideration of human factors in Oman. *Technol. Soc.* **2018**, *55*, 183–198. [CrossRef]
5. Crompton, H.; Burke, D. The use of mobile learning in higher education: A systematic review. *Comput. Educ.* **2018**, *123*, 53–64. [CrossRef]
6. Piotrowski, J.T.; Meester, L. Can apps support creativity in middle childhood? *Comput. Hum. Behav.* **2018**, *85*, 23–33. [CrossRef]
7. Shih, Y.E.; Mills, D. Setting the New Standard with Mobile Computing in Online Learning. *Int. Rev. Res. Open Distance Learn.* **2007**, *8*, 1–16. [CrossRef]
8. Rodríguez-Cano, S.; Cuesta-Gómez, J.L.; Delgado-Benito, V.; de la Fuente-Anuncibay, R. Educational Technology as a Support Tool for Students with Specific Learning Difficulties—Future Education Professionals' Perspective. *Sustainability* **2022**, *14*, 6177. [CrossRef]
9. Bonneton-Botté, N.; Fleury, S.; Girard, N.; Le Magadou, M.; Cherbonnier, A.; Renault, M.; Anquetil, E.; Jamet, E. Can tablet apps support the learning of handwriting? An investigation of learning outcomes in kindergarten classroom. *Comput. Educ.* **2020**, *151*, 103831. [CrossRef]
10. Schenke, K.; Redman, E.J.K.H.; Chung, G.K.W.K.; Chang, S.M.; Feng, T.; Parks, C.B.; Roberts, J.D. Does "Measure Up!" measure up? Evaluation of an iPad app to teach preschoolers measurement concepts. *Comput. Educ.* **2020**, *146*, 103749. [CrossRef]
11. Wang, F.; Gao, C.; Kaufman, J.; Tong, Y.; Chen, J. Watching versus touching: The effectiveness of a touchscreen app to teach children to tell time. *Comput. Educ.* **2021**, *160*, 104021. [CrossRef]
12. Pimmer, C.; Mateescu, M.; Gröhbiel, U. Mobile and ubiquitous learning in higher education settings. A systematic review of empirical studies. *Comput. Hum. Behav.* **2016**, *63*, 490–501. [CrossRef]
13. Douglas, B.D.; Brauer, M. Gamification to prevent climate change: A review of games and apps for sustainability. *Curr. Opin. Psychol.* **2021**, *42*, 89–94. [CrossRef]
14. Zydney, J.M.; Warner, Z. Mobile apps for science learning: Review of research. *Comput. Educ.* **2016**, *94*, 1–17. [CrossRef]
15. Dickinson, K.J.; Bass, B.L. A Systematic Review of Educational Mobile-Applications (Apps) for Surgery Residents: Simulation and Beyond. *J. Surg. Educ.* **2020**, *77*, 1244–1256. [CrossRef] [PubMed]
16. Papadakis, S.; Kalogiannakis, M.; Zaranis, N. Educational apps from the Android Google Play for Greek preschoolers: A systematic review. *Comput. Educ.* **2018**, *116*, 139–160. [CrossRef]
17. Stamatios, P. Can Preschoolers Learn Computational Thinking and Coding Skills with ScratchJr? A Systematic Literature Review. *Int. J. Educ. Reform* **2022**, 1–34. [CrossRef]
18. Arnedo-Moreno, J.; Tesconi, S.; Marco, M.J.; García, D.; Fondo, M. A study on the use of gameful approaches in self-paced "learn to code" (SPL2C) apps. In Proceedings of the 3rd International Symposium on Gamification and Games for Learning (GamiLearn'19), Barcelona, Spain, 22 October 2019; Volume 2497.
19. Silic, M.; Silic, D. Novel Approach to Learn to Code Using Gamification. 2020. Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3654313 (accessed on 3 January 2023). [CrossRef]
20. Pila, S.; Aladé, F.; Sheehan, K.J.; Lauricella, A.R.; Wartella, E.A. Learning to code via tablet applications: An evaluation of Daisy the Dinosaur and Kodable as learning tools for young children. *Comput. Educ.* **2019**, *128*, 52–62. [CrossRef]
21. Ball, T.; Zorn, B. Viewpoint: Teach foundational language principles. *Commun. ACM* **2015**, *58*, 30–31. [CrossRef]
22. Francisco, J.G.P.; Rees, A.M.; Hughes, J.; Vermeersch, J.; Jormanainen, I.; Toivonen, T. A survey of resources for introducing coding into schools. In Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing, Salamanca, Spain, 2–4 November 2016; pp. 19–26. [CrossRef]
23. Hsu, Y.C.; Ching, Y.H. Mobile app design for teaching and learning: Educators' experiences in an online graduate course. *Int. Rev. Res. Open Distance Learn.* **2013**, *14*, 117–139. [CrossRef]
24. Wolber, D. App inventor and real-world motivation. In Proceedings of the SIGCSE'11—42nd ACM Technical Symposium on Computer Science Education, Dallas, TX, USA, 9–12 March 2011; pp. 601–606. [CrossRef]
25. Hutchison, A.; Nadolny, L.; Estapa, A. Using Coding Apps to Support Literacy Instruction and Develop Coding Literacy. *Read. Teach.* **2016**, *69*, 493–503. [CrossRef]
26. Sheehan, K.J.; Pila, S.; Lauricella, A.R.; Wartella, E.A. Parent-child interaction and children's learning from a coding application. *Comput. Educ.* **2019**, *140*, 103601. [CrossRef]
27. Micah, L.; Bibu, G.D. Mobile-Based Python Tutor for High School Students. *Int. J. Comput. Sci. Mob. Comput.* **2019**, *8*, 72–78.

28. Okonkwo, C.W.; Ade-Ibijola, A. Python-bot: A chatbot for teaching python programming. *Eng. Lett.* **2021**, *29*, 25–34.
29. Fabic, G.V.F.; Mitrovic, A.; Neshatian, K. *Learning with Engaging Activities via a Mobile Python Tutor*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2017; Volume 10331, pp. 613–616. [CrossRef]
30. Schnieder, M.; Williams, S. How to Assess Programming Skills: Review and Analysis. In Proceedings of the 2022 IEEE German Education Conference, Berlin, Germany, 11–12 August 2022; pp. 1–6.
31. Tabi, K.; Randhawa, A.S.; Choi, F.; Mithani, Z.; Albers, F.; Schnieder, M.; Nikoo, M.; Vigo, D.; Jang, K.; Demlova, R.; et al. Mobile apps for medication management: Review and analysis. *JMIR mHealth uHealth* **2019**, *7*, e13608. [CrossRef]
32. Mannila, L.; Peltomäki, M.; Salakoski, T. What about a simple language? Analyzing the difficulties in learning to program. *Comput. Sci. Educ.* **2006**, *16*, 211–227. [CrossRef]
33. Pears, A.; Seidman, S.; Mannila, L.; Malmi, L.; Adams, E.; Jens, B.; Devlin, M.; Paterson, J. A Survey of Literature on the Teaching of Introductory Programming. *ACM SIGCSE Bull.* **2007**, *39*, 204–223. [CrossRef]
34. Bailey, S.C.; Belter, L.T.; Pandit, A.U.; Carpenter, D.M.; Carlos, E.; Wolf, M.S. The availability, functionality, and quality of mobile applications supporting medication selfmanagement. *J. Am. Med. Inform. Assoc.* **2014**, *21*, 542–546. [CrossRef] [PubMed]
35. Shen, N.; Levitan, M.J.; Johnson, A.; Bender, J.L.; Hamilton-Page, M.; Jadad, A.R.; Wiljer, D. Finding a depression app: A review and content analysis of the depression app marketplace. *JMIR mHealth uHealth* **2015**, *3*, 1–18. [CrossRef]
36. Bender, J.L.; Yue, R.Y.K.; To, M.J.; Deacken, L.; Jadad, A.R. A lot of action, but not in the right direction: Systematic review and content analysis of smartphone applications for the prevention, detection, and management of cancer. *J. Med. Internet Res.* **2013**, *15*, e2661. [CrossRef]
37. StatCounter Global Stats. Mobile Operating System Market Share Worldwide: April 2021–April 2022. 2021. Available online: https://gs.statcounter.com/os-market-share/mobile/worldwide (accessed on 15 May 2022).
38. IDC: The Premier Global Market Intelligence Firm, "Smartphone Market Share". 2021. Available online: https://www.idc.com/promo/smartphone-market-share (accessed on 6 June 2020).
39. Preibusch, S. The value of privacy in web search. In Proceedings of the Twelfth Workshop on the Economics of Information Security (WEIS), Washington, DC, USA, 11–12 June 2013.
40. Winkler, S.; Zeadally, S. An analysis of tools for online anonymity. *Int. J. Pervasive Comput. Commun.* **2015**, *11*, 436–453. [CrossRef]
41. Landis, J.R.; Koch, G.G. The measurement of observer agreement for categorical data. *Biometrics* **1977**, *33*, 159–174. [CrossRef] [PubMed]
42. Cooke, N.; Hawwash, K.; Smith, B. Python for Engineers Concept Inventory (PECI): Contextualized assessment of programming skills for engineering undergraduates. In Proceedings of the 47th SEFI Annual Conference 2019-Varietas Delectat: Complexity Is the New Normality, Budapest, Hungary, 16–19 September 2019; pp. 270–279.
43. Wang, Y.; Li, H.; Feng, Y.; Jiang, Y.; Liu, Y. Assessment of programming language learning based on peer code review model: Implementation and experience report. *Comput. Educ.* **2012**, *59*, 412–422. [CrossRef]
44. Mallawaarachchi, S.R.; Tieppo, A.; Hooley, M.; Horwood, S. Persuasive design-related motivators, ability factors and prompts in early childhood apps: A content analysis. *Comput. Hum. Behav.* **2023**, *139*, 107492. [CrossRef]
45. Huang, H.H.; Lin, C.N. Influencing factors of mobile instant messaging applications between single- and multi- platform use cases. *Comput. Stand. Interfaces* **2023**, *83*, 103658. [CrossRef]
46. Dolawattha, D.D.M.; Salinda Pramadasa, H.K.; Jayaweera, P.M. The Impact Model: Teachers' Mobile Learning Adoption in Higher Education. *Int. J. Educ. Dev. Using Inf. Commun. Technol.* **2019**, *15*, 71–88.
47. Ansari, M.S.; Tripathi, A. An investigation of effectiveness of mobile learning apps in higher education in India. *Int. J. Inf. Stud. Libr.* **2017**, *2*, 33–41.
48. Liu, H. Survey on College Students' Mobile English Learning through APPs. In Proceedings of the 3rd International Conference on Arts, Design and Contemporary Education, Moscow, Russia, 29–30 May 2017; Volume 144, pp. 859–862. [CrossRef]
49. Green, L.S.; Hechter, R.P.; Tysinger, P.D.; Chassereau, K.D. Mobile app selection for 5th through 12th grade science: The development of the MASS rubric. *Comput. Educ.* **2014**, *75*, 65–71. [CrossRef]