The Journal of Open Source Education

FDS Python: Fundamentals and Data Structures with Python

Ram B. Basnet[1] **and Tenzin Doleck**[2]

1 Colorado Mesa University **2** McGill University

DOI: 10.21105/jose.00047

Software

- Review
- Repository
- Archive

Submitted: 05 February 2019
Published: 26 April 2019

Summary

FDS Python is a set of Jupyter notebooks that covers fundamental concepts of programming and data structures using Python. These interactive notebooks are based on the open-source textbook: Think Python: How to Think Like a Computer Scientist (http://openbookproject.net/thinkcs/python/english3e/index.html#) (Wentworth, Elkner, Downey, & Meyers, 2012). The notebooks consist of a summary of the theory/lecture notes and syntax of concepts along with interactive code examples. The notebooks provide coverage from the basic programming concepts to data structures and popular libraries such as turtle, pygame, sqlite3, etc. The mapping of notebook chapters with the textbook chapters is provided in the README.md file.

These notebooks are designed to accommodate beginners as well as experienced programmers who are interested in learning to code in Python. More importantly, the interactive nature of these notebooks force students to code to learn and not learn to code. Pythontutor (http://pythontutor.com) (Guo, 2013) is used to help students visualize and understand the concepts better by working on problems from open.kattis.com (https://open.kattis.com/) (Basnet, Doleck, Lemay, & Bazelais, 2018) where appropriate.

Statement of need

It is well acknowledged that programming is a challenging endeavor (Robins, Rountree, & Rountree, 2013). As such, considerations of the suitability of programming languages, such as Python, for teaching and learning programming have received growing interest (Mannila, Peltomäki, & Salakoski, 2006). In fact, among programming languages, Python is the most popular language for introductory computer science courses among US universities (Guo, 2014).

Textbooks are commonly used for delivering course content in programming courses; however, "a majority of university students do not regularly read course textbooks" (Rockinson-Szapkiw, Wendt, & Lunde, 2013). To ameliorate this situation, educators have suggested the use of computational notebooks, such as the Jupyter computational notebook project, for offering both new ways of instruction and making interactive curricula (O'Hara, Blank, & Marshall, 2015; Shen, 2014).

Although Python provides a way to quickly write and test code and concepts in the interactive ">>>, chevron prompt" mode, the codes and results vanish and cannot be reproduced once the session is terminated. Moreover, writing multiple line of codes or functions is not intuitive because of the language's strict syntactic rules on enforcing whitespaces to represent block of codes. Jupyter notebooks can easily overcome these

drawbacks as one can quickly and interactively write single to many lines of codes and at the same time keep track of the output results as part of the notes. Furthermore, students can refer to what they have done and manage all their notes in one convenient location.

The notebooks are complete and can be easily adopted by other instructors who wish to teach Python using a practical, follow-along live coding approach. These notebooks are being used in university courses (Beginning and Advanced Programming courses) since 2017 at Colorado Mesa University. Depending on the need and the audience skill level, instructors in introductory programming course can spend more time demonstrating the concepts with many examples supplementing them with visualization; in contrast, in advanced or intensive boot camp like courses, instructors can skip the theoretical portions and instead focus on the syntax and spend more time on advanced topics and concepts.

References

Basnet, R., Doleck, T., Lemay, D., & Bazelais, P. (2018). Exploring computer science students' continuance intentions to use kattis. Education and information technologies, *23(3)*, 1145–1158. doi:10.1007/s10639-017-9658-2

Guo, P. (2013). Online python tutor: Embeddable web-based program visualization for cs education. *Proceeding of the 44th ACM technical symposium on Computer science education*, 579–584.

Guo, P. (2014). Python is now the most popular introductory teaching language at top u.s. Universities [blog]. https://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext.

Mannila, L., Peltomäki, M., & Salakoski, T. (2006). What about a simple language? Analyzing the difficulties in learning to program. *Computer Science Education*, *16(3)*, 211–227. doi:10.1080/08993400600912384

O'Hara, K., Blank, D., & Marshall, J. (2015). Computational notebooks for ai education. *Proceedings of the International Florida Artificial Intelligence Research Society Conference (FLAIRS), Hollywood, FL*.

Robins, A., Rountree, J., & Rountree, N. (2013). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13(2)*, 137–172. doi:10.1076/csed.13.2.137.14200

Rockinson-Szapkiw, A., Wendt, J., & Lunde, R. (2013). Electronic versus print textbooks: The influence of textbook format on university students' self-regulated learning strategies, motivation, and text anxiety. *American Journal of Distance Education*. doi:10.1080/08923647.2013.796230

Shen, H. (2014). Interactive notebooks: Sharing the code. *Nature*. doi:10.1038/515151a

Wentworth, P., Elkner, J., Downey, A. B., & Meyers, C. (2012). *How to think like a computer scientist: Learning with python 3 (rle)* (2nd ed.). http://openbookproject.net/thinkcs/python/