




# PLNQ: Simplifying Code Kata Preparation

Zachary Kurmas <sup>1</sup>

<sup>1</sup> Grand Valley State University

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Submitted: 13 June 2025

Published: unpublished

## License

Authors of papers retain copyright  
and release the work under a  
Creative Commons Attribution 4.0  
International License ([CC BY 4.0](#))

## High-Level Description

PLNQ allows instructors to define small, auto-graded coding exercises (henceforth called *code katas*) using a single Jupyter notebook. In general, a code kata requires three parts:

1. A textual description of the problem to be solved
2. Test cases used by the auto-grader
3. Metadata used by the assessment environment (CodingBat, PrairieLearn, zyBooks, etc.)

A [Jupyter](#) notebook is a single document that contains both text blocks and code blocks. \* The problem description can be placed in a text block \* Test cases can be placed in a code block \* Metadata can also be placed in a code block.

The use of a Jupyter notebook significantly simplifies the process of creating and managing code katas as compared to other kata platforms which typically either (a) place each component in a separate file, or (b) require users to enter all the components into a single file or web form that only supports one type of formatting.

The current version of PLNQ supports the preparation of Python exercises for [PrairieLearn](#); however, \* It would not be difficult to add support for other platforms such as [CodingBat](#), [Donda](#), etc. \* Adding support for additional programming languages is trivial for languages Jupyter supports directly. Adding support for other languages is possible, but might not be practical.

The PLNQ source is available on GitHub (<https://github.com/kurmasz/plnq>) and is licensed under the [GNU GPL v3](#). In addition, PLNQ is available as a PyPi package: [plnq-gvsu](#).

The GitHub repository contains both \* A set of [examples](#) illustrating each major feature, and \* [Complete documentation](#).

In addition, \* I have shared the PrairieLearn assignment I created using PLNQ for my Python course. You can browse them [here](#). \* You can also watch a [video](#) that explains my motivation for creating PLNQ, then walks through the different components of a template notebook as well as the execution of PLNQ to produce a working PrairieLearn question.

## Statement of Need:

Writing code is a skill that requires significant practice ([Spacco et al., 2015](#)). One popular method of practice is to complete many *code katas*: short, focused coding exercises — preferably in an environment that provides immediate feedback ([Edwards et al., 2019](#)). There are many platforms<sup>1</sup> that provide this type of practice; however, preparing katas presents some logistical challenges.

<sup>1</sup>Just a few of the many platforms that provide code katas: [CodingBat](#), [CloudCoder](#), [CodeWorkout](#), [Donda](#), [zyBooks](#).

37 A typical kata requires, at minimum \* a problem description \* a set of test cases, and \*  
38 some metadata.

39 Each of these components is best edited and managed using a different format. \* The  
40 problem description is often a plain text file, HTML, Markdown, or Latex. \* The set of  
41 test cases is often source code (although it could also be a data serialization format like  
42 json, yaml, or XML). \* The metadata is often a data serialization format (although it could  
43 also be source code).

44 The use of different formats leads to a tension between ease of preparation/editing, and  
45 ease of management: It is easiest enter a code kata when each component is placed in a  
46 separate file where authors can use different support tools (syntax highlighters, formatters,  
47 linters etc.). However, it is easiest to manage and distribute a code kata when the entire  
48 kata is contained in a single file. Placing the entire kata in a single file can also simplify  
49 editing by reducing the need to switch between tabs/files.

50 The large number of code kata platforms demonstrates that either approach can work;  
51 however, the fact that there is such a large number of platforms suggests that many users  
52 aren't completely satisfied with currently available options.

53 In addition, because different platforms take different approaches to managing katas, it  
54 is difficult for instructors at different institutions to share katas unless they happen to  
55 use the same platform. (Sharing is especially difficult in the case when the platforms use  
56 differing sets of files.)

57 PLNQ provides the best of both options by using a single Jupyter notebook to define a code  
58 kata. Jupyter notebooks contain both text and code blocks, allowing authors to place  
59 all of a kata's components into a single file, yet still manage each component using the  
60 most appropriate highlighting and formatting tools. Jupyter notebook platforms provide  
61 text-focused highlighting and formatting tools for text blocks, as well as code-focused  
62 syntax highlighting and formatting tools for code blocks.

63 Edwards, S. H., Murali, K. P., & Kazerouni, A. M. (2019). The relationship between  
64 voluntary practice of short programming exercises and exam performance. *Proceedings*  
65 *of the ACM Conference on Global Computing Education*, 113–119. <https://doi.org/10.1145/3300115.3309525>  
66

67 Spacco, J., Denny, P., Richards, B., Babcock, D., Hovemeyer, D., Moscola, J., & Duvall, R.  
68 (2015). Analyzing student work patterns using programming exercise data. *Proceedings*  
69 *of the 46th ACM Technical Symposium on Computer Science Education*, 18–23. <https://doi.org/10.1145/2676723.2677297>  
70