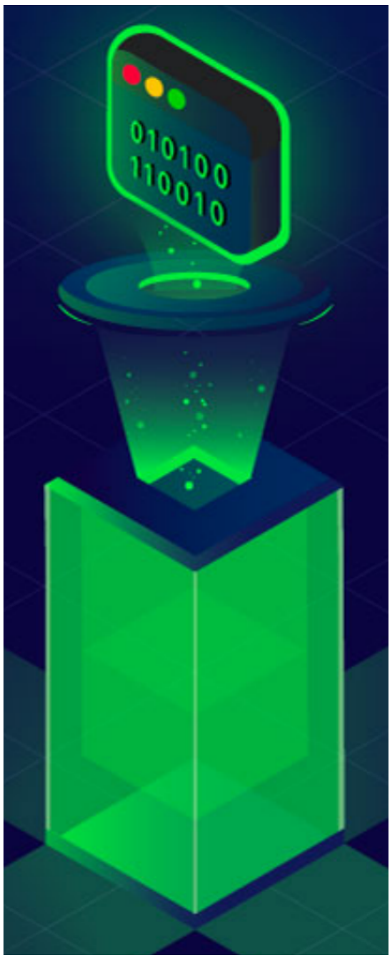
Leapfrog

Guide to best practices for developer

Version 2.0

Developer's Handbook



Author(s):
Anish Manandhar
Bala Maharjan
Nischal Khadka
Robus Gauli
Sawan Vaidya
Saugat Gautam

Editor(s):
Suja Manandhar
Designer(s):
Ishan Manandhar

April 12, 2020

Introduction

Software development is as much of an art as it is science. It is not only the application of one's logic but also the expression of their imagination and craftsmanship. Though it might not seem like a big deal for a new developer to create working applications, gaining mastery of this craft requires passion, dedication, pragmatism and continuous learning. This handbook has been designed to provide you with a concise introduction to the best practices that one should follow in their road to mastering the art of software development. Though it may not be possible to precisely quantify the value of these best practices, they are observed to be commonly used by the best of the developers and successful organizations.

While it is not feasible to cover every topic in depth, the handbook intends to provide you with a starting point. You should dive deeper on your own to gain an in-depth knowledge of the best practices and to implement them in real time scenarios. You might find dozens of books for many of the topics covered in the handbook. From a beginner's perspective, it might be overwhelming to digest everything covered in the handbook at once. Hence, multiple reads at certain intervals is encouraged.

Also, this excerpt from Josh Bloch's seminal book Effective Java is equally applicable to this handbook:

"While the rules in this handbook do not apply 100 percent of the time, they do characterize best programming practices in the great majority of cases. You should not slavishly follow these rules, but violate them only occasionally and with good reason. Learning the art of programming, like most other disciplines, consists of first learning the rules and then learning when to break them."

Finally, you're bound to make mistakes or errors in judgement no matter how much you try to adhere to the best practices. Learn from your mistakes and take the responsibility to rectify them. Be open to admit ignorance or error. Or to even admit that you need help.

Who is this handbook for?

This handbook is primarily written for software developers who are also referred to as software engineers or programmers. These terms mean different things but this handbook uses these terms interchangeably. This book can also be used by people other than developers. For example a Quality Assurance Engineer can use this to know about core weaknesses of a project which leads to bugs and other problems. Similarly, a Project Manager can read this to know what developers should do to improve the health of the project

Contents

Part I

Content

1. Readability or Understandability

Readable code is one in which the components of the code are easy to grasp and the goal of the overall code is well defined. Building a codebase that is readable is not just a single goal for a team but a commitment that lasts the entire duration of the project. Developers should think of themselves as authors of their code instead of configuration of a complex piece of software.

Here are some cultural habits that leads to a readable code:

- A good code review practice is in place and rules are set from the onset of the project.
- Developers are mindful of how someone new will perceive their code.
- The Boy Scout Rule - "Leave the campground cleaner than you found it" is followed which iteratively improves the readability of code.

1.1 Well formatted and follows standards

Every popular language/framework has guidelines over how code should be formatted and a list of dos and don'ts. It is best to start with these standards at the onset of the project. Sometimes the team may have custom stylistic preferences. These should be well documented and people from outside the team may not follow these standards. Also, it is important to not overstretch the standards as this makes documenting and enforcing standards a burden in itself. As it could be a tedious process to monitor, evaluate and enforce the standards, most teams use linters and prettifiers. Additionally, standard code quality analyzer tools like Sonarqube and Codeclimate can dramatically improve the code review process through automation. It is highly recommended that projects mandate usage of these tools.

1.2 Right documentation

It is popularly said that good code is self documented. While code should definitely have inherent readable qualities, it is wishful thinking to rely on this alone. Just like coding skills, documentation skills can be developed through learning and practice. Developers are recommended to learn about documentation syntaxes and standards for their respective frameworks. Expertise in Docblocks, knowledge of Markdown, RST or alike, and of diagram tools such as draw.io, lucidcharts etc help in producing crisp documentation. A project can also start using automatic documentation generation tools of their liking.

1.3 Convention over configuration

Conventions such as naming and placement of artifacts increase the readability of code in two ways. First, they enforce a high degree of consistency, and second they eliminate the burden of thinking about what parameters needs to be set in a configuration file.

To take this a step further is to create structures such as Abstract classes, Enumerations and choose a logic organization strategy such as MVC (Model View Controller)

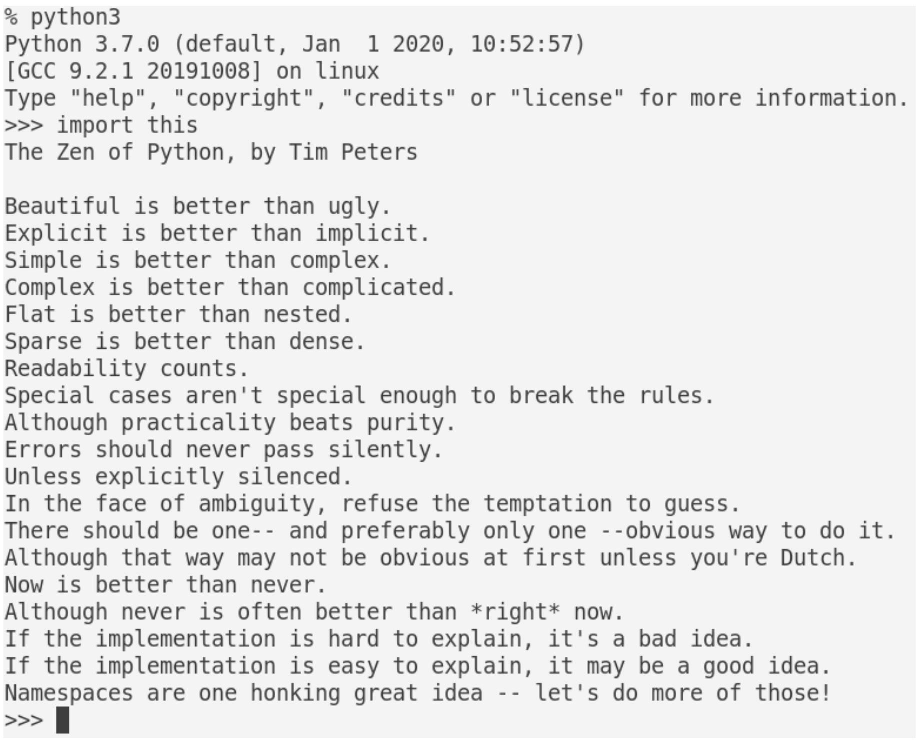
1.4 Libraries

It is advised to practice restraint on picking libraries. We can run into libraries that fully solve our problems but are obscure, and there are ones that partially solve our problems that are very popular.

Writing one's own library can sometimes seem tempting and can be beneficial as the end product is tailored to the project's needs and is fully under the developer's control. However, with the power also comes the necessity to maintain and document the workings of the library. Libraries which are publicly available do not necessitate this as the documentation and maintenance is performed by the owner or the community.

1.5 Zen of Python

Although this topic might look like it's specific to Python, you will find it useful across all languages and platforms. Like "Zen Religion", you will find that many things in this guideline is not absolute and is open to interpretation



```
% python3
Python 3.7.0 (default, Jan  1 2020, 10:52:57)
[GCC 9.2.1 20191008] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>> 
```

1.6 Useful Patterns

Some other ways to increase readability of code are as follows:

- Use Objects, Maps or Dictionaries instead of too many conditionals.
- Use Programmable objects in representing APIs such as the ones provided by ORMs.
- Naming should be consistent across variables, functions, files, database artifacts etc. For example, having a consistent style of prefixing such as "is_" prefix for booleans, "on_" prefix for callbacks, "view_" prefix for views etc can enhance readability
- Write functions that have a single responsibility. Avoid writing functions with side-effects.
- Focus on tests as tests such as unit tests to improve code quality as well as readability

2. Math

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam a orci ornare nibh tincidunt molestie sed nec tellus. Morbi non sapien id lorem posuere pretium. Vestibulum commodo cursus purus, a elementum sem imperdiet sit amet.

$$\begin{matrix} -2 & 1 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 \\ 0 & 0 & 1 & -2 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & 1 \\ 0 & 0 & 0 & \cdots & 1 & -2 \end{matrix}$$

Phasellus posuere dolor dignissim aliquam tempus. Morbi egestas felis in lorem varius, ac egestas ante lacinia. Nulla sed ultrices dui. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras rutrum nisl ut ligula tristique gravida. Fusce augue enim, elementum in euismod a, mollis et nunc. Duis vel ipsum eros. Pellentesque odio nisl, bibendum non ex in, sodales laoreet enim. Curabitur eu leo ac urna scelerisque dictum at non tellus.

$$\begin{aligned} codeword &= m(x) \,.\, g(x) \\ &= (1 + x + x3).(1 + x + x3) \\ &= 1 + x + x3 + x + x2 + x4 + x3 + x4 + x6 \\ &= 1 + 2x + x2 + 2x3 + 2x4 + x6 \\ &= (1, 0, 1, 0, 0, 0, 1)\, mod\, 2 \end{aligned}$$

3. Code

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam a orci ornare nibh tincidunt molestie sed nec tellus. Morbi non sapien id lorem posuere pretium. Vestibulum commodo cursus purus, a elementum sem imperdiet sit amet. Phasellus posuere dolor dignissim aliquam tempus. Morbi egestas felis in lorem varius, ac egestas ante lacinia. Nulla sed ultrices dui. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras rutrum nisl ut ligula tristique gravida. Fusce augue enim, elementum in euismod a, mollis et nunc. Duis vel ipsum eros. Pellentesque odio nisl, bibendum non ex in, sodales laoreet enim. Curabitur eu leo ac urna scelerisque dictum at non tellus.

Listing 3.1: A sample Python listing

```python
import requests

class Scanner:
    """The Scanner scans items on sandbox escaping."""

    def __init__(self, item):
        """Initialize class for the given item."""

        self.__queue_item = item

        item.mount("http://", requests.HTTPAdapter)
        item.mount("https://", requests.HTTPAdapter)

    @staticmethod
    def hello():
        """Print an example message."""

        print("Hello␣World!")
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit (see 3.1).

4. Citation

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam a orci ornare nibh tincidunt molestie sed nec tellus. Morbi non sapien id lorem posuere pretium. Vestibulum commodo cursus purus, a elementum sem imperdiet sit amet. Phasellus posuere dolor dignissim aliquam tempus. Morbi egestas felis in lorem varius, ac egestas ante lacinia. Nulla sed ultrices dui. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras rutrum nisl ut ligula tristique gravida. Fusce augue enim, elementum in euismod a, mollis et nunc. Duis vel ipsum eros. Pellentesque odio nisl, bibendum non ex in, sodales laoreet enim. Curabitur eu leo ac urna scelerisque dictum at non tellus[1].

[1]Gommers, T. (2017). This is a test citation that links to my homepage. https://finnwea.com.

5. Table

The schedule 5.1 may be subject to changes on short notice. These activities are the *do* stage of the Deming Wheel.

Table 5.1: List of audit activities to perform

Date	**Activity**
01-01-2018	On-site audit activities:
09:00	**Opening Meeting** Establish personal contact with the auditee, confirm the plan for carrying out the audit, explain and confirm the activities, roles and responsibilities of those involved in the audit, confirm communication arrangements and reporting requirements and provide an opportunity for the auditee to clarify issues and ask any questions.
10:00	**Documentation** Documentation about scoping, planning, implementing, monitoring and reviewing.
12:00	**Information security management system (ISMS)** This meeting will be held with security officers aswell as the process manager.
21:00	**Closing Remarks (Audit Follow-Up)** The primary purpose of this meeting is to present the audit findings and conclusions, ensure a clear understanding of the results, and agree on the timeframe for corrective actions. This meeting can be held at the end of the audit.

Part II

Extra

Bibliography

Gommers, T. (2017). This is a test citation that links to my homepage. https://finnwea.com.

Appendices