Project Annotation Manual

William Tholke
willtholke@berkeley.edu

Andrew Zhang
zhang.andrew@berkeley.edu

Alex Truong
altruong@berkeley.edu

School of Information, University of California, Berkeley[*]

April 24, 2023

Abstract

This annotation manual provides a structured approach for assessing the readability of Python code snippets based on a scoring system. We give an overview of the Python programming language and a description of the data collection process, which involves extracting code snippets from popular Python repositories on GitHub. Moreover, we introduce five main readability features that fall under the following categories: code length and repetition, inline comments and docstrings, naming conventions and case, whitespace, and miscellaneous items.

Contents

[*]Affiliated with Info 159, Natural Langauge Processing, Spring 2023

1 Introduction

This task involves assigning a readability score to snippets of code written in the Python programming language, with the score ranging from 1 to 5 (inclusive) on a scale representing low to high readability.

1.1 Python Overview

Python is a high-level, dynamically typed programming langauage with a large standard library, and is often recognized for its general-purpose use cases. As of July 2022, Python is the fourth most used programming language among developers worldwide, preceded only by languages used for web development and database queries, JavaScript/HTML/CSS and SQL, respectively [1]. This makes it a great candidate for readability evaluation.

It is recommended that the reader possess a fundamental grasp of the Python programming language. If necessary, please refer to the Python 3.11.2 documentation for additional information and clarification. At the time of writing, the most recent release of the language is Python 3.11.2, but the reader may reference documentation from as early as Python 3.6.0.

The data only include snippets with code written in Python 3.6.0 and greater, since release 3.6.0 (2016) introduced string literals, which are encoded as a feature of readability. Note that major additions to the language beyond release 3.6.0, such as data classes and the walrus operator (:=) from release 3.8.0 (2019), are not encoded as features of readability.

We have encoded various requirements of the PEP-8 Guidelines [3] as features, which were created with the philosophy that "code is read more often than it is written" in mind.

1.2 Data Collection and Structure

The data collection process is handled by a script that extracts and exports code snippets from popular Python repositories on GitHub to formatted `tsv` and `txt` files. To access the source code for this process, please refer to the `data-collection/` directory in our GitHub repository. If the repository is private, please contact the authors to request access. The script does the following:

a. Uses the GitHub API to fetch repositories based on user input (minimum number of stars and forks) and prompts the user on whether it should
   i. Use data from previous API request(s), stored in `data-collection/raw-data/`, or
   ii. Make a new API request to collect the data

b. Traverses recursively through each repository up until either running out of data or reaching the user-defined upper bound for the total number of files to consider, executing the following:

i. Searches for the `setup.py` file in the root directory and, if present, analyzes it using the `ast` module to determine the version of Python used in the project. If the version is not 3.6.0 or greater, it will no longer consider the repository for data collection.

ii. If versioned correctly, the program will search for and store all Python files present in the repository until either reaching the user-defined upper bound for the number of files to consider in each repository or hitting the API rate limit.

iii. If the rate limit is hit, the program will wait 60 seconds and then attempt to continue searching for Python files.

c. Traverses through each Python file in each of the cleaned repositories and extracts code snippets by identifying groups of lines of code as either a "Class" or "Function," capturing each of their respective signatures and bodies together as a categorized snippet.

d. Exports the collected data to `data-collection/cleaned-data/` and structures it in files with a unique 4-character identifier in their name:

i. `adjudicated-full-[####].tsv`: the data with columns "UID," "Category," and "Snippet," where "UID" has values with the format `<Github username>|<repository name>|<file name>|<repository-specific snippet ID>`.

ii. `adjudicated-[####].tsv`: the data with no column labels, but with the data in each respective column as **1)** a range of IDs increasing from 0 to 499 by row **2)** the text "adjudicated" in each row **3)** the placeholder for a label, "label-na" **4)** and the code snippet.

e. Prompts the user to review `adjudicated-[####].tsv` and remove any unhelpful data manually, producing a formatted `txt` file when the user indicates that they are finished.

Examples of exported data can be found in `data-collection/cleaned-data/`.

1.3 Acknowledgements

We would like to express our gratitude to David Bamman[1] for his invaluable guidance and support throughout the duration of this project.

The data used in this project was sourced from open-source Python projects publicly available on GitHub. The data is neither private nor under copyright, and can be freely shared with others. The code associated with this paper is licensed under the MIT License.

[1] **dbamman@berkeley.edu**

2 Readability Features

2.1 Code Length and Repetition

- Individual lines should have a max length of 79 characters [6].
- Docstrings and in-line comments should have a max length of 72 characters from the left-margin, specifically if the docstring spans multiple lines each line may only have a max length of 72 characters (see subsection 2.4) [6].

2.2 Inline Comments and Docstrings

- At least one of the following should be present, describing the purpose of the function or class in the snippet:
  - docstring [2]
  - in-line comment(s) [5]
  - descriptive assert statements
  - error handling

2.3 Naming Conventions and Case

- Variables, if present, should be in snake_case[2].
- At least one of the following should be true:
  - the class name is in CamelCase
  - the function name is in snake_case.
- Function or class parameters, if present, should be self-explanatory. For example, parameter names `x`, `y`, `temp`, and `param` are not self-explanatory, whereas `file_path`, `user_id`, and `is_enabled` are clearly self-descriptive.

2.4 Whitespace

- There should be consistency between tabs and spaces for indentation, meaning that the snippet should not contain both tabs and spaces.
- Blank lines should be used, sparingly, to indicate logical sections. Blank lines should be omitted between a bunch of related one-liners [4].

[2]Note that private functions or classes are indicated with an underscore before any alphabetical characters (for example, `_class_name`) and satisfy snake_case convention.

2.5 Miscellaneous

- There should not be extremely dense code blocks or extraneous lines of code or in-line comments.
- There should not be residual to-do statements or lines of non-functional code that were commented out.
- There should not be comparison of boolean values to True or False using == [7].

3 Readability Scoring System

There are five subsections and each subsection contains individual features. A snippet satisfies a subsection if all of the features are met. If any of the features are missed, the subsection is not satisfied. Give the Python snippet one point for each subsection it satisfies, up to a total of 5 points, and replace label-na with the number, from 1 to 5.

4 Annotation Process

4.1 Annotation Workflow

After downloading the .tsv file, open up VSCode and install the extension ”Edit csv”. Now open up the .tsv file, then on the top right corner there will be an ”edit csv” button. Click that, and it will open a tab which has a nicely-formatted table of the snippet number, annotator, your score for the readability of the snippet (**you will edit this column**), and the Python snippet. Now, you will go through the 250 snippets and assign a readability score to each one, based on the readability features and the scoring system detailed in section 3. A quick way to move through entering in scores is to type in a number and hit enter, which moves your cursor to same column in the row below.

4.2 Exporting Annotations

Use the conversions.py file to convert `txt` file data with raw unannotated data back to a `tsv` to easily edit it, and then convert that new `tsv` back to a `txt` file.

5 References

[1] Statista. Most used programming languages among developers worldwide as of 2022, 2022. URL: https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/.

[2] Guido van Rossum, Barry Warsaw, and Nick Coghlan. Pep 257 – docstring conventions, 2001. URL: https://peps.python.org/pep-0257/#what-is-a-docstring.

[3] Guido van Rossum, Barry Warsaw, and Nick Coghlan. Pep 8 – style guide for python code, 2013. URL: https://peps.python.org/pep-0008/.

[4] Guido van Rossum, Barry Warsaw, and Nick Coghlan. Pep 8 – style guide for python code: Blank lines, 2013. URL: https://peps.python.org/pep-0008/#blank-lines.

[5] Guido van Rossum, Barry Warsaw, and Nick Coghlan. Pep 8 – style guide for python code: Inline comments, 2013. URL: https://peps.python.org/pep-0008/#inline-comments.

[6] Guido van Rossum, Barry Warsaw, and Nick Coghlan. Pep 8 – style guide for python code: Maximum line length, 2013. URL: https://peps.python.org/pep-0008/#maximum-line-length.

[7] Guido van Rossum, Barry Warsaw, and Nick Coghlan. Pep 8 – style guide for python code: Programming recommendations, 2013. URL: https://peps.python.org/pep-0008/#programming-recommendations.