Packaging as Publishing

Python Code Formalizations

Presentation Outline

1. ✍️ Publishing
2. 📦 Python Packaging
3. 🧰 Tools

Publishing

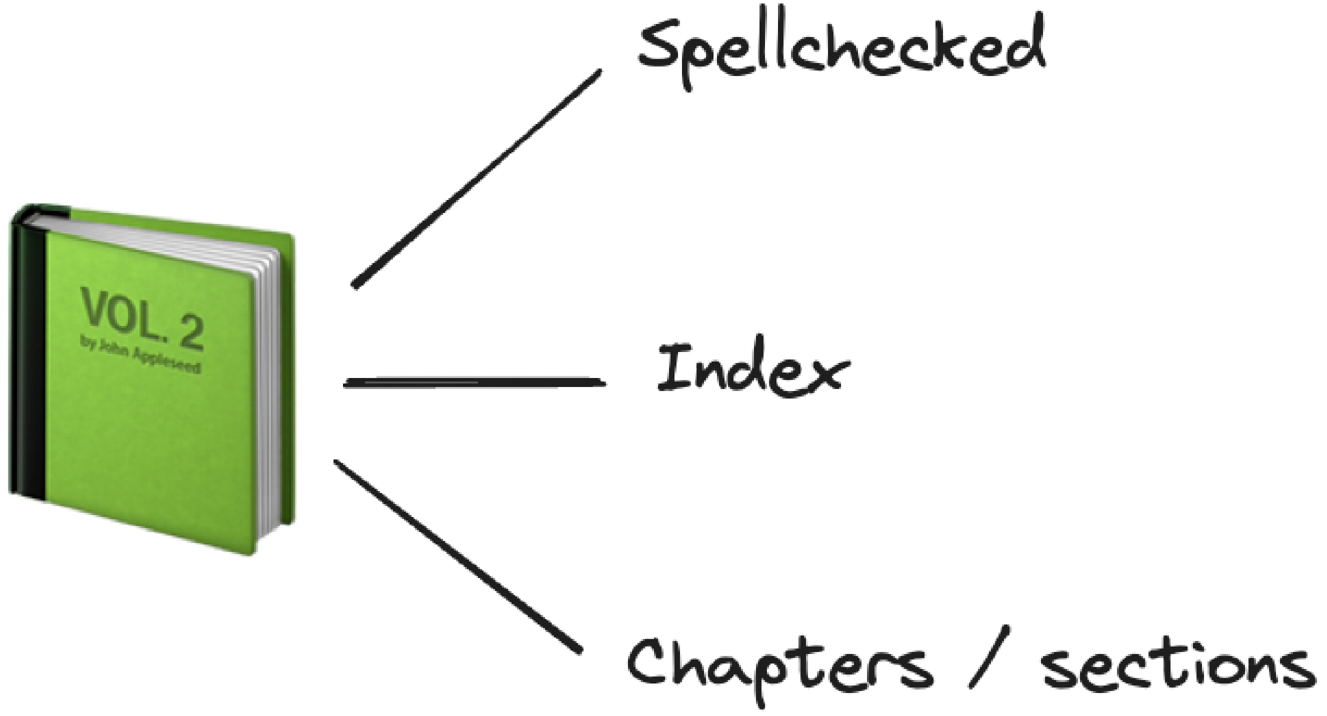
Why does this matter?

> Following publishing conventions increases:
>
> - Understanding
> - Trust
> - Reach

Publishing - Text vs Book

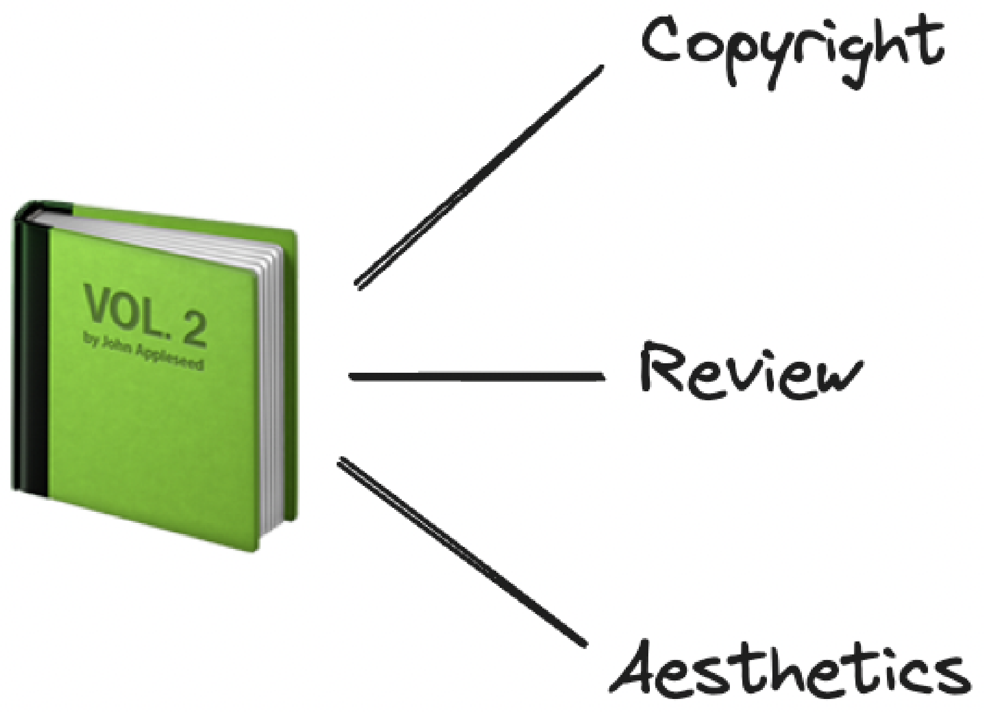
How are these two different?

Publishing - Understanding



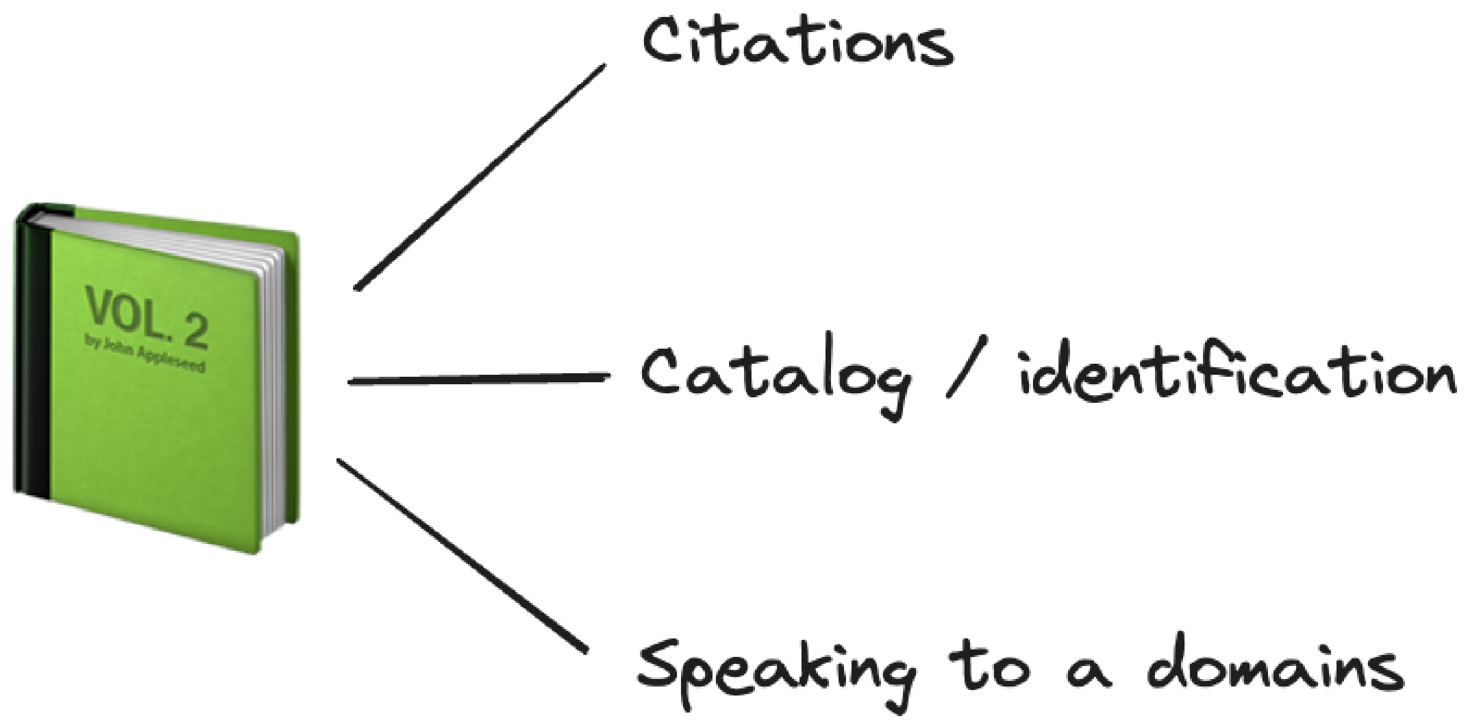
- Unsurprising formatting for **understanding** (sections, cadence, spelling, etc.)

Publishing - Trust



- Sense of **trust** from familiarity, connection, and authority (location, review, or style)

Publishing - Connection



- **Connection** to a wider audience (citations, domains, cataloging)

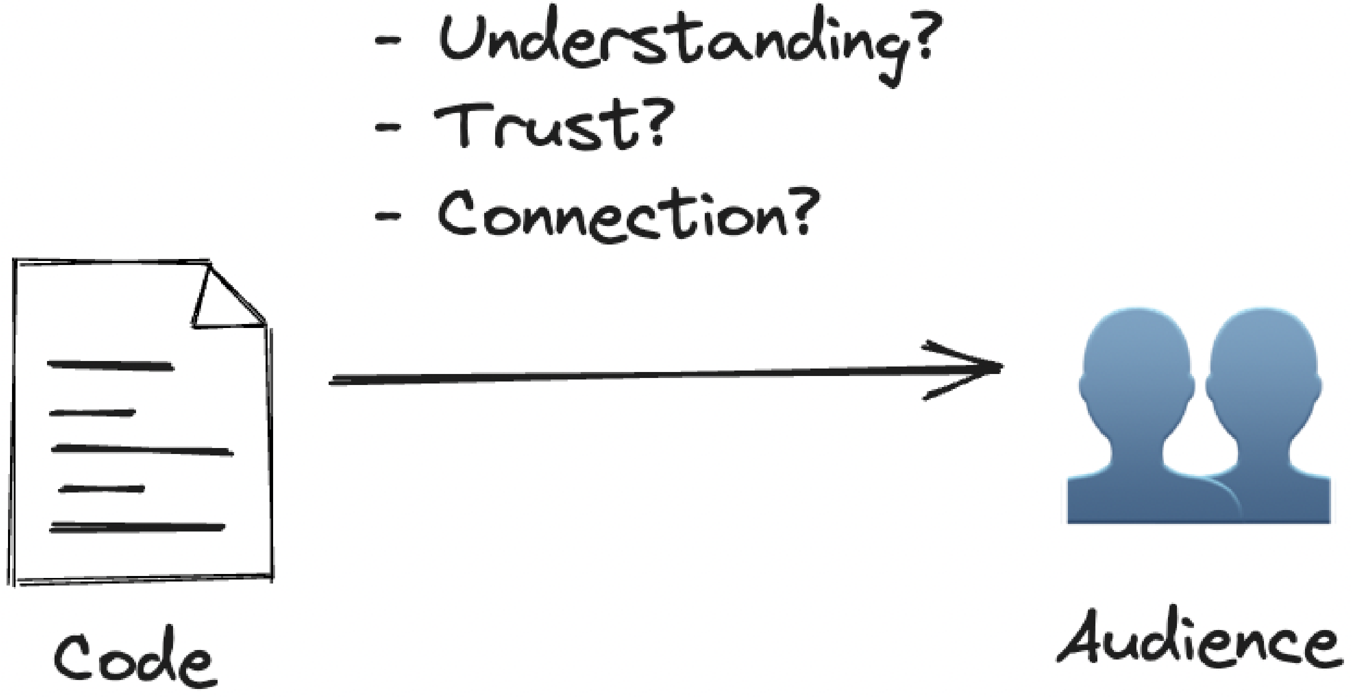
Publishing - Code as Language

Code is another kind of written language.

Ignoring language conventions can often result in poor grammar, or ***code smell***.

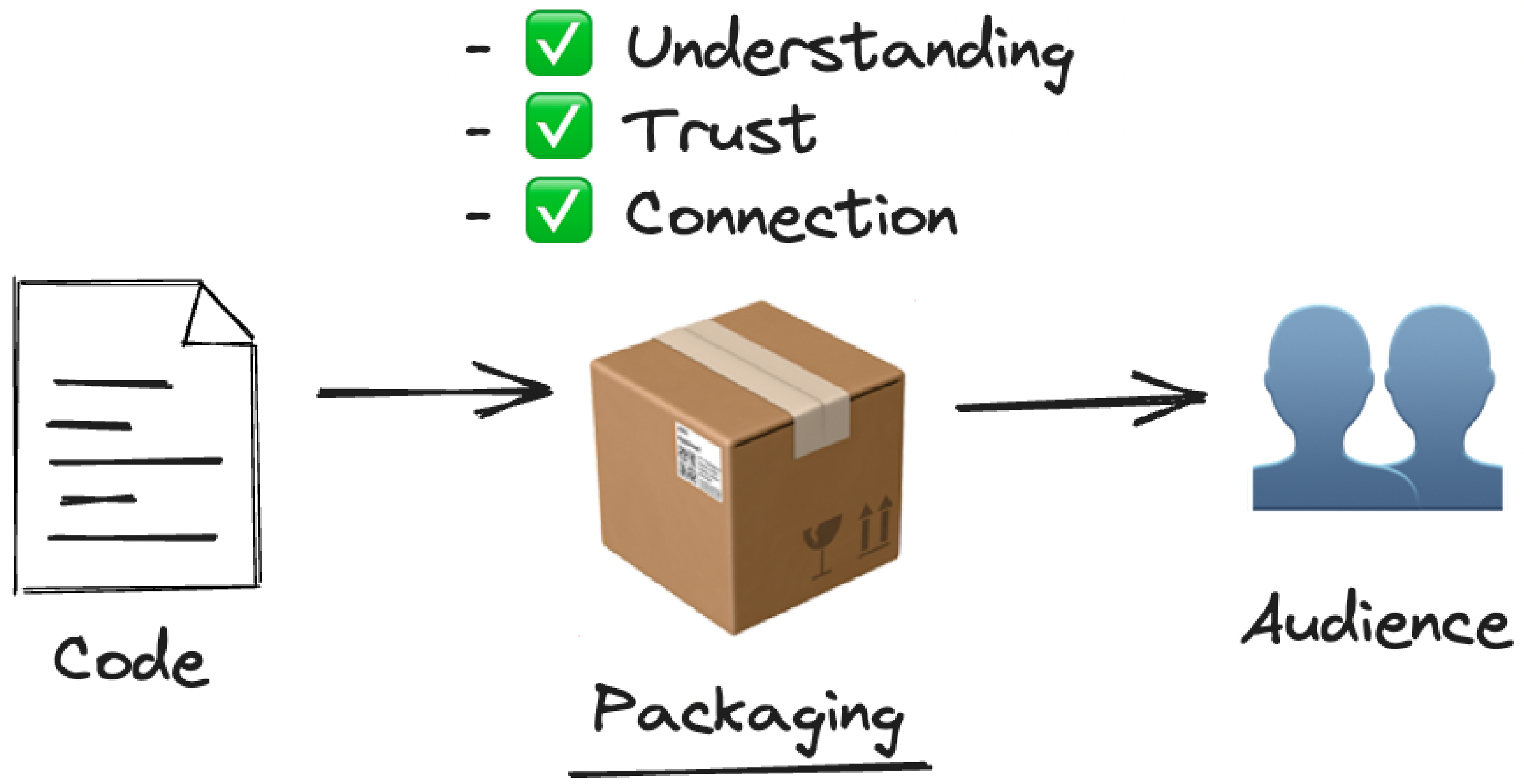
Code smells are indications that something might be going wrong. They generally reduce the understanding, trust, and connection for your code.

Publishing - Code as Language



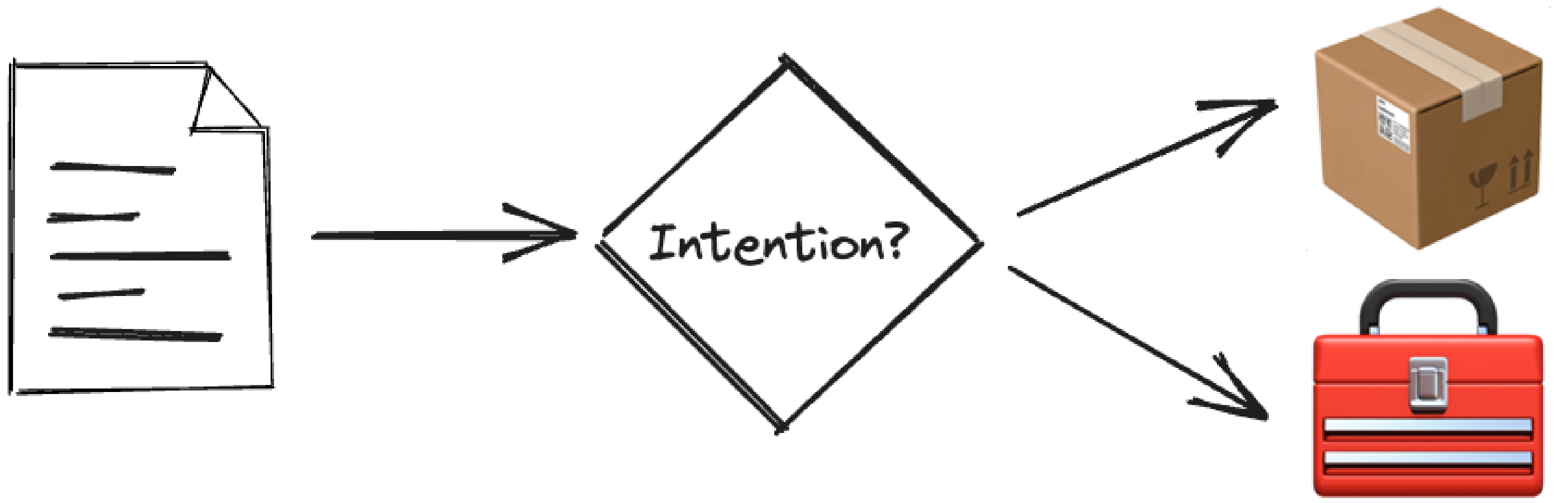
Who are you writing for? Do they understand, trust, and connect with your code?

Publishing - Python



Publishing in Python involves the act of "**packaging**".

Publishing - Python



Python packaging is a loose practice which requires adjustment based on intention (there's no one-size fits all forever solution here).

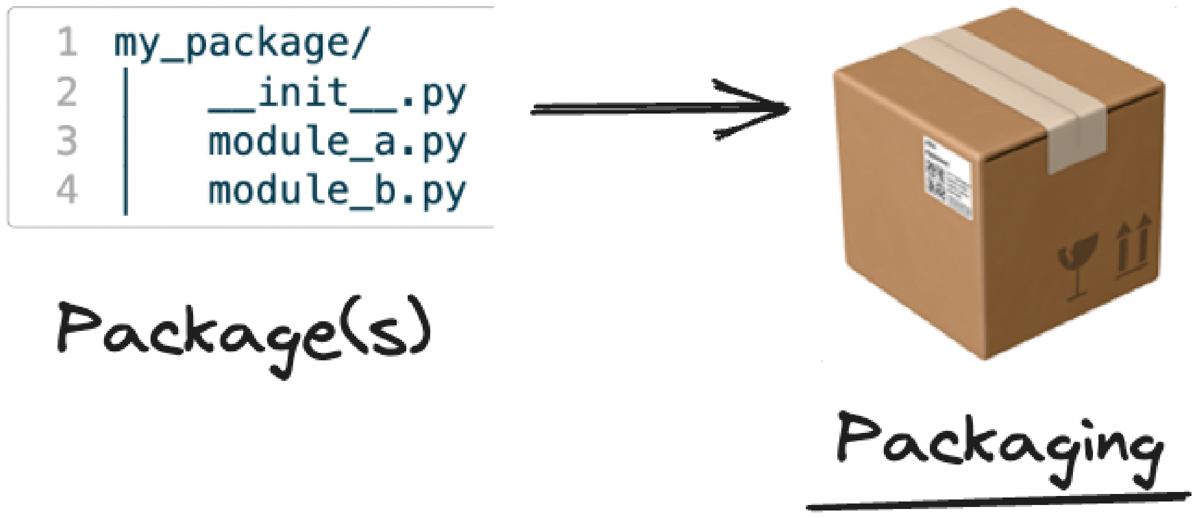
- For example: we'd package Python code differently for a patient bedside medical device use vs a freeware desktop videogame.

Python Packaging - Definitions

```
my_package/
│    __init__.py
│    module_a.py
│    module_b.py
```

- A Python **package** is a collection of modules (`.py` files) that usually include an “initialization file” `__init__.py`.

Python Packaging - Definitions



```
my_package/
│    __init__.py
│    module_a.py
│    module_b.py
```

- Python ***"packaging"*** is a broader term indicating formalization of code with publishing intent.

Python Packaging - Definitions

- Python packages are commonly installed from **PyPI** (Python Package Index, https://pypi.org).

> For example: `pip install pandas` references PyPI by default to install for the `pandas` package.

Python Packaging - Understanding

```
project_directory
├── README.md
├── LICENSE.txt
├── pyproject.toml
├── src
│   └── package_name
│       └── __init__.py
│       └── module_a.py
└── tests
    └── __init__.py
    └── test_module_a.py
```

Python Packaging today generally assumes a specific directory design. Following this convention generally improves the **understanding** of your code.

Python Packaging - README.md

```
project_directory
├── README.md # used for documentation
...
```

The **`README.md`** file is a markdown file with documentation including project goals and other short notes about installation, development, or usage.

- The `README.md` file is akin to a book jacket blurb which quickly tells the audience what the book will be about.

Python Packaging - LICENSE.txt

```
project_directory
├── README.md
├── LICENSE.txt # indicates usage persmissions and protections
...
```

The **`LICENSE.txt`** file is a text file which indicates licensing details for the project. It often includes information about how it may be used and protects the authors in disputes.

- The `LICENSE.txt` file is akin to a book’s copyright page.
- See https://choosealicense.com/ for more details on selecting an open source license.

Python Packaging - pyproject.toml

```
project_directory
├── README.md
├── LICENSE.txt
├── pyproject.toml # outlines the project organization (and much more)
...
```

The **`pyproject.toml`** file is a Python-specific TOML file which helps organize how the project is used and built for wider distribution. More here later!

- The `pyproject.toml` is akin to a book's table of contents, index, and printing or production specification.

Python Packaging - Source Code

```
project_directory
├── README.md
├── LICENSE.txt
├── pyproject.toml
├── src # isolates source code for use in project
│   └── package_name
│       └── __init__.py
│       └── module_a.py
...
```

The `src` directory includes primary source code for use in the project. Python projects generally use a nested package directory with modules and sub-packages.

- The `src` directory is akin to a book's body or general content (perhaps thinking of modules as chapters or sections of related ideas).

▭	▭
1	2