Chapter 07 - Batteries Included Creating a Python SDK

- Software Development Kit (SDK): Custom software library that acts as a wrapper for your API → consumers interact with your API directly in their programming language without requiring extra code to handle API communication
  - Code examples
  - Debuggers
  - Documentation

> **Zan Markan**
>
> What is SDK? is a programming language-native means of exposing an API that would otherwise only be accessible as a REST endpoint $\xrightarrow{\text{how?}}$ usually bundle the common REST API calls in functions or classes that you can use directly in code by importing a library
> If your users are coders it is common to provide them an SDK

> **Joey Greco**
>
> A well-built SDK takes care of all the nitty-gritty for you, it gives me a few lines of code I can copy/paste on my machine along with a few examples of how to access and manipulate various data. It tells me what I need to do to authenticate, It's a great way to bridge the gap from an external service to the code that you're writing.

> **People just want the Data! Your API don't matter to them!**

> **Simon Yu**
>
> All the boilerplate code that API consumers needed to write before is already taken care of by the SDK library itself. Instead of every consumer re-inventing the wheel themselves, they simply import the SDK call the correct method, and go!
> An API consumer often won't want to pay for a service until they are up and running in production. For many API providers therefore, unblocking API consumers also

unblocks revenue.
Since SDKs provide a ready-made way to integrate with the API and eliminate the need to write custom integration code from scratch, they dramatically reduce the support burden required of API producers.
Without SDKs, if an integration doesn't work, API producers often get pulled into 1:1 support, which is extremely costly.

- When you use SDK, you get auto-complete and type hints + reduce TTHW metric + you can enforce user to use your API responsibly by conforming to call limits + sending correctly formatter requests. → doing the right thing, is the easy thing with SDK

> **Francisco Goitia**
>
> What motivated StatsBomb to create SDKs for your APIs?
> Our philosophy is to understand the end user of our products. Our user might be a football analyst who doesn't know how to query APIs. they just want to do a plot of the data to get their job done. So, if they can install a Python library with `pip install` and start using the data, it makes their life easier.
> For the users of your APIs, why do you tink SDKs make their life easier?
> If I was writing client code to use our APIs, I would need to write something like the SDK to decouple the API from my software. That's using good software practices. So, by using the SDK, they don't have to do that extra work.

Picking a Language for Your SDK

- It doesn't need to be written in the same language as your API, it needs to be written that is most used by your consumers → They use SDK to interact with your API instead of calling the API directly $\xrightarrow{\text{Zan Markan}}$ "Go where the users are"
- Python SDK → Open Source Tool = OpenAPI Python Client
- Generating SDK from OpenAPI Specifications (OAS) → Speakeasy, APIMatic, Fern
  - Why use this tools? because getting the SDK design and Implementation right is difficult + Maintaining and supporting them is even harder + If the developer that was responsible for this task leave what you do?

Minimum Viable SDK

Making Your SDK Easy to Install

- npm for Node.js, Maven for Java, for Python it is Python Package Index (PyPI) $\xrightarrow{\text{Enable}}$ install packages by `pip` $\xrightarrow{\text{if}}$ project structured correctly even install directly from GitHub repo by `pip`
- Create a `pyproject.toml`
- Build-Backend $\xrightarrow{\text{We use}}$ Setuptools $\xrightarrow{\text{Other Options}}$ Hatchling, Flit, PDM, UV

Making the SDK Consistent and Idiomatic

> **Simon Yu**
>
> The SDK should be consistent and predictable. Naming conventions, error handling, and response format should be the same throughout the SDK to avoid unnecessary confusion to users.

- Your SDK should be idiomatic $\xrightarrow{\text{means}}$ following the norms used by other programmers of that language $\xrightarrow{\text{for Python means}}$ Your code most be Pythonic $\xrightarrow{\text{means}}$ Follow the Python Conventions = Python Enhancement Proposals (PEPs)
- Style of SDK = PEP 8 - Style Guide for Python Code, Summary:
  - 4-space indentations
  - Keep lines to 79 characters or less
  - Use `UpperCamelCase` for classes
  - `lowercase_with_underscores` for functions and methods
- Other Additional Convention that we will be using in SDK:
  - PEP 202 - List Comprehensions
  - PEP 343 - Context Mangers
  - PEP 257 - Docstrings
  - PEP 518 - Build System Requirements
  - PEP 484 - Type Hints
- Create Directory Structure:
  - `mkdir src/swcpy`
    `touch src/swcpy/__init__.py` → SDK will be a Python Package $\xrightarrow{\text{so}}$ each directory that contain code, must contain `__init__.py`
    `touch src/swcpy/swc_client.py`

- `swc_client.py`: primary client that will interact with API
- Install your package locally, you need to run this command in a place that your `pyproject.toml` is
  `pip install -e .`
- look at the file tree of your project
  `tree --prune -I 'build|*.egg-info|__pycache__'`

Building a Feature-Rich SDK

Using Sane Defaults

- To hide complicated details implement sane defaults
  - SDK should know base URL of the API without being told
    - If SDK is a read-only wrapper for a public API = default production API address
    - If API requires authentication or is a read/write API = default to sandbox environment $\xrightarrow{\text{so}}$ prevent accidents
- `swc_config.py`

Providing Rich Functionality

- Handling versions
- Handling Pagination
- Client-side caching
- Authentication
- Data type validation
  - It is easier for API provider to do this $\xrightarrow{\text{why?}}$ because of it's knowledge + We have access to Pydantic
- Retry/backoff logic
  When you make an API call from your SDK, sometimes it may fail due to:
  Temporary network hiccup
  Slowdown in the API
  Due to a load-balancing
  Service is in the middle of bringing more servers online to handle increased load
  Solution: make API resilient by retrying a few times before giving up
  Problems of the Solution: Accidental Distributed Denial-of-Service (DDOS) attack from your own user by just simply continuously retrying
  Solution: Exponential backoff instead of simple retries:

The time between each try will get exponentially longer with each failed attempt → more failure happens = more break needs
Problem: people would cluster over the same time
AWS blog - Exponential Backoff and Jitter
* Solution: Jitter ≈ random element around backoff $\xrightarrow{\text{How to Implement it?}}$ Backoff Python Library

> **Best Solution = Exponential Backoff with Jitter**

Performing Logging

- Don't be black box → Show user what is happening under the hood → 2 ways:
  - Opensource
  - Implement meaningful logging
    - Python built-in logging library

Logging Levels

- `logging.DEBUG`: Detailed information, typically only for interest to a developer trying to diagnose a problem.
- `logging.INFO`: Confirmation that things are working as expected.
- `logging.WARNING`: Indication that something unexpected happened or that a problem might occur in the near future But the software is still working as expected.
- `logging.ERROR`: Indicates that, due to a more serious problem, the software has not been able to perform some function.
- `logging.CRITICAL`: A serious error, indicating that the program itself may be unable to continue running.

> **Logging messages**
>
> The logs with level of `WARNING` or more will always be displayed by default
> But to see `INFO` or `DEBUG` → `--log-level`

Hiding Your API's Complicated Details

> **Joey Greco**
>
> As a data consumer, I don't want to have to worry about API versioning, headers, authentication, rate-limiting, or hunting down the correct endpoints to use, I just want to call some function and be able to do things that are meaningful to me.

Supporting Bulk Downloads

> **Robin Linacre**
>
> As a data scientist, I find the complexity of many open data services frustrating. I don't want to have to learn how to query an endpoint, think about data types, or read through API documentation. Just give me the data!

- Can be implemented by use of FastAPI's Static Files
  - SDK will access the files from their web-hosted location + it will build url for each file
- Parquet files: Open source column-oriented data file format designed for efficient data storage and retrieval.
  - Provides high performance compression
  - Encoding schemes to handle complex data in bulk
  - Supported in many programming language + analytic tools
- Pydantic → data validation
- backoff → retry functionality
- PyArrow → handling parquet file

Documenting Your SDK

- SDK get documented by a adding a comprehensive docstrings to the methods that will be used by programmers → `help()`
- `tree --prune -I 'build|*.egg-info|__pycache__'`
- Update your SDK with pip: `pip install --upgrade .`

Testing Your SDK

- How to implement our tests? you need to choose a test layout, we are using a "test outside application" $\xrightarrow{\text{means}}$ testing against the installed module instead of code in your local path

Support Every Task the API Supports

- User should be able to accomplish any task with SDK that they could accomplish by directly using API $\xrightarrow{\text{means}}$ Every API endpoint and parameter should be supported by SDK

> **Implement these functions**
>
> - `get_health_check` (completed)
> - `list_leagues` (completed)
> - `get_league_by_id`
> - `get_counts`
> - `list_teams`
> - `list_players`
> - `get_player_by_id`
> - `list_performances`
> - Bulk functions
> - `get_bulk_player_file` (completed)
> - `get_bulk_league_file`
> - `get_bulk_performance_file`
> - `get_bulk_team_file`
> - `get_bulk_team_player_file`

Additional Resources:

- Test Python Package Publishing with PyPI Test
- Packaging Python Projects - Python Packaging User Guide
- Why parquet files are my preferred API for bulk open data
- Code Style — The Hitchhiker's Guide to Python
- How To Build A Best In Class Python SDK | Speakeasy
- Welcome to SDKs.io | SDKs.io

- Good Integration Practices — pytest documentation