**STEELEYE ASSIGNMENT**

**Introduction**

Fast API

FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.6+ based on standard Python type hints.

The key features are:

Fast: Very high performance, on par with NodeJS and Go (thanks to Starlette and Pydantic). One of the fastest Python frameworks available.

Fast to code: Increase the speed to develop features by about 200% to 300%.

Fewer bugs: Reduce about 40% of human (developer) induced errors.

Intuitive: Great editor support. Completion everywhere. Less time debugging.

Easy: Designed to be easy to use and learn. Less time reading docs.

Short: Minimize code duplication. Multiple features from each parameter declaration. Fewer bugs.

Robust: Get production-ready code. With automatic interactive documentation.

**Constraints**

An API, or Application Programming Interface, is a server that you can use to retrieve and send data to using code. APIs are most commonly used to retrieve data, and that will be the focus of this beginner tutorial.

When we want to receive data from an API, we need to make a request. Requests are used all over the web. For instance, when you visited this blog post, your web browser made a request to the Dataquest web server, which responded with the content of this web page.

**Making API Requests in Python**

In order to work with APIs in Python, we need tools that will make those requests. In Python, the most common library for making requests and working with APIs is the requests library. The requests library isn’t part of the standard Python library, so you’ll need to install it to get started.

If you use pip to manage your Python packages, you can install requests using the following command:

*pip install requests*

If you use conda, the command you’ll need is:

*conda install requests*

Once you’ve installed the library, you’ll need to import it. Let’s start with that important step:

*import requests*

Now that we’ve installed and imported the requests library, let’s start using it.

**Database**

The Python Database API (DB-API) defines a standard interface for Python database access modules. It’s documented in PEP 249. Nearly all Python database modules such as sqlite3, psycopg, and mysql-python conform to this interface.

**SQLAlchemy**

[SQLAlchemy](http://www.sqlalchemy.org/) is a commonly used database toolkit. Unlike many database libraries it not only provides an ORM layer but also a generalized API for writing database-agnostic code without SQL.

*$ pip install sqlalchemy*

**Records**

[Records](https://github.com/kennethreitz/records) is minimalist SQL library, designed for sending raw SQL queries to various databases. Data can be used programmatically or exported to a number of useful data formats. Also included is a command-line tool for exporting SQL data.

*$ pip install records*

**PugSQL**

[PugSQL](https://pugsql.org/) is a simple Python interface for organizing and using parameterized, handwritten SQL. It is an anti-ORM that is philosophically lo-fi, but it still presents a clean interface in Python.

*$ pip install pugsql*

**Django ORM**

The Django ORM is the interface used by [Django](https://www.djangoproject.com/) to provide database access. It’s based on the idea of [models](https://docs.djangoproject.com/en/dev/#the-model-layer), an abstraction that makes it easier to manipulate data in Python.

The basics:

* Each model is a Python class that subclasses django.db.models.Model.
* Each attribute of the model represents a database field.
* Django gives you an automatically-generated database-access API; see [Making queries](https://docs.djangoproject.com/en/dev/topics/db/queries/).

**peewee**

[peewee](http://docs.peewee-orm.com/en/latest/) is another ORM with a focus on being lightweight with support for Python 2.6+ and 3.2+ which supports SQLite, MySQL, and PostgreSQL by default. The [model layer](https://peewee.readthedocs.io/en/latest/peewee/quickstart.html#model-definition) is similar to that of the Django ORM and it has [SQL-like methods](https://peewee.readthedocs.io/en/latest/peewee/quickstart.html#retrieving-data) to query data. While SQLite, MySQL, and PostgreSQL are supported out-of-the-box, there is a [collection of add-ons](https://peewee.readthedocs.io/en/latest/peewee/playhouse.html#playhouse) available.